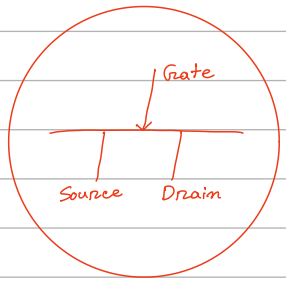
Chapter - 1

Moore's Law: The number of transistors on a microchip would double approximately every 18 - 24 months, leading to a corresponding increase in computer power.

** However, in recent years, it has become increasingly challenging to sustain the pace of doubling transistor counts on a chip at the same rate as before.

→ physical limitations.
→ Quantum effects.
→ Quantum Tunneling effect



⇒ transistor

Understanding Performance of a program...

- Algorithm / original program
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation — Chapter 2,3
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

Software Based

Hardware based

software used to translate the program into machine instructions.

— Chapter 4,5,6

Eight Great Ideas

- Design for ***Moore's Law***

- Use ***abstraction*** to simplify design

Hide the lower level details to keep things simple.

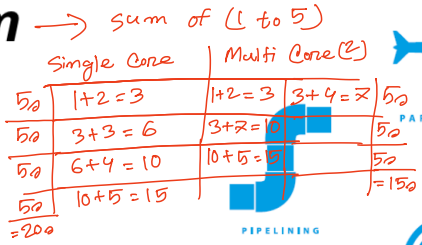
- Make the ***common case fast***

You should know the common case first via experimentation

- Performance *via* ***parallelism***

- Performance *via* ***pipelining***

will be explained in chap-4



- Performance *via* ***prediction***

- ***Hierarchy*** of memories

will be explained in chap-5

- ***Dependability*** *via* redundancy

computers need to be both fast and reliable. Since any hardware can break, we make systems reliable by adding extra components that can take over if something breaks and to help spot any problem.

Why do we carry a spare tire in the trunk??

backup / redundancy

Imagine you're at a crosswalk, and the traffic light is about to change. Instead of waiting for the "Walk" signal to definitely light up, you start crossing when you see the cars slowing down and the opposite light turning red. You predict that it's safe to cross based on these cues. Most of the time, your prediction is accurate, and you save a bit of time by starting to cross earlier.

Now, if you mispredict and the "Walk" signal doesn't light up, you quickly step back to the curb. The cost of stepping back is low (you just lose a few seconds), and the mechanism to recover (stepping back) is simple. Because your predictions are usually correct, you cross the street faster on average compared to waiting every time for the "Walk" signal to appear.

In computing, this concept is similar to speculative execution in CPUs. The processor guesses the likely path of a program and starts executing instructions ahead of time. If the guess is correct, the program runs faster. If the guess is wrong, the processor discards the work done on the wrong path and starts over from the correct point. As long as the prediction is usually right and the cost of recovering from a wrong guess is low, this approach speeds up the overall performance.

Statistically, the guesses are often correct and the cost of recovering from a wrong guess is relatively low.

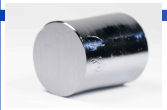
Datapath: Imagine your kitchen as a computer's datapath. The datapath is the part of the computer where data is processed, just like your kitchen is where you prepare food

1. Ingredients (Data):
   - The ingredients you use for cooking (vegetables, spices, etc.) represent the data the computer needs to process.
2. Chef (Processor):
   - The chef, is like the computer's processor. You take the ingredients and transform them into a meal. Similarly, the processor takes data and performs operations on it.
3. Kitchen Tools (Functional Units):
   - The tools you use (knives, pots, pans) are like the functional units in a datapath (ALU, multipliers, etc.). These tools help you process the ingredients. For example, a knife is used to chop vegetables, and a pan is used to cook them. In a datapath, the ALU might add or subtract numbers, while other units handle multiplication or data storage.
4. Recipe (Instructions):
   - The recipe you follow to make a dish is like the set of instructions the computer follows to process data. Each step in the recipe tells you what to do with the ingredients. In a computer, instructions tell the processor what operations to perform on the data.
5. Kitchen Workflow (Data Flow):
   - The way you move ingredients through your kitchen, from the fridge to the cutting board to the stove, represents the flow of data through the datapath. The datapath moves data from memory to the processor, through the functional units, and back to memory if needed.

ISA => Different ISAs define how a processor understands and executes instruction.

ABI => ABI defines how software interacts with the system's hardware and OS.

Manufacturing ICs

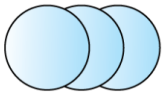


cylindrical shaped silicon

Silicon ingot

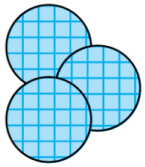
Slicer

Blank wafers — round disc of silicon



20 to 40 processing steps → photo lithography

Patterned wafers

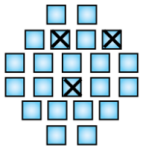


Wafer tester

Tested wafer

Dicer

Tested dies

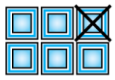


Bond die to package — a protective layer

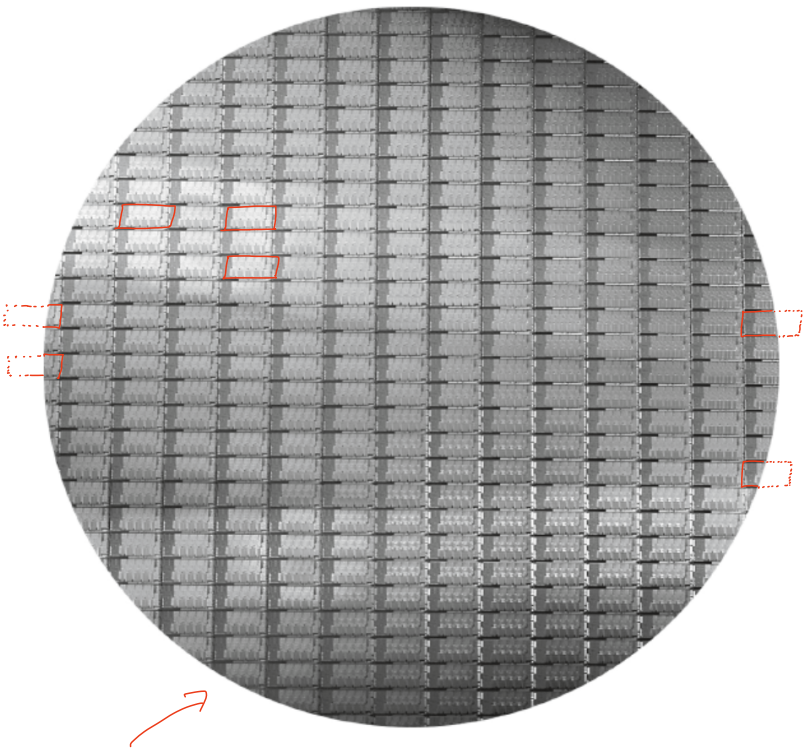
Packaged dies

Part tester

Tested packaged dies



Ship to customers



this is a wafer. As the shape is round,

So, Wafer Area = πr^2

and, Die can be in rectangular or square shape.

Area = Height × Width

$$\text{Dies per Wafer} \approx \frac{\text{Wafer Area}}{\text{Die Area}}$$

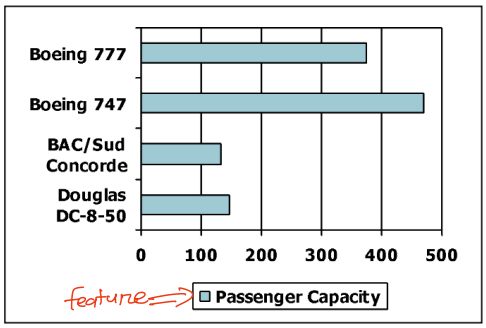
$$\text{Cost per die} = \frac{\text{Cost per Wafer}}{\text{Dies per Wafer} \times \text{Yield}}$$

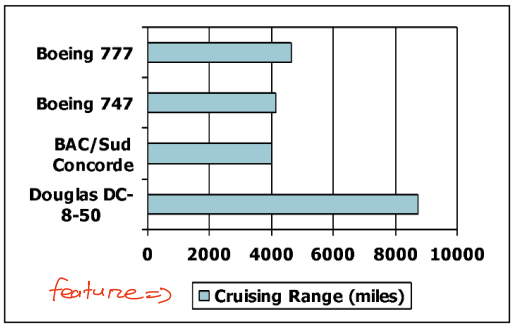
Yield → amount of working dies per wafer.

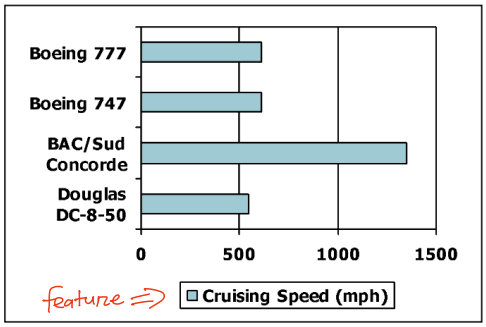
Defining Performance

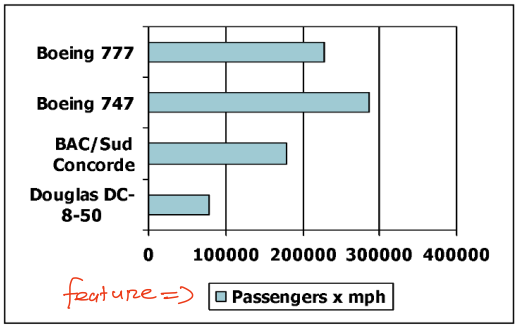
whenever we speak of performance, we must mention the feature based on which we calculated the performance.

- Which airplane has the best performance?









lesser response time ⟶ better performance

$$\Rightarrow \text{Performance} = \frac{1}{\text{Execution time}}$$

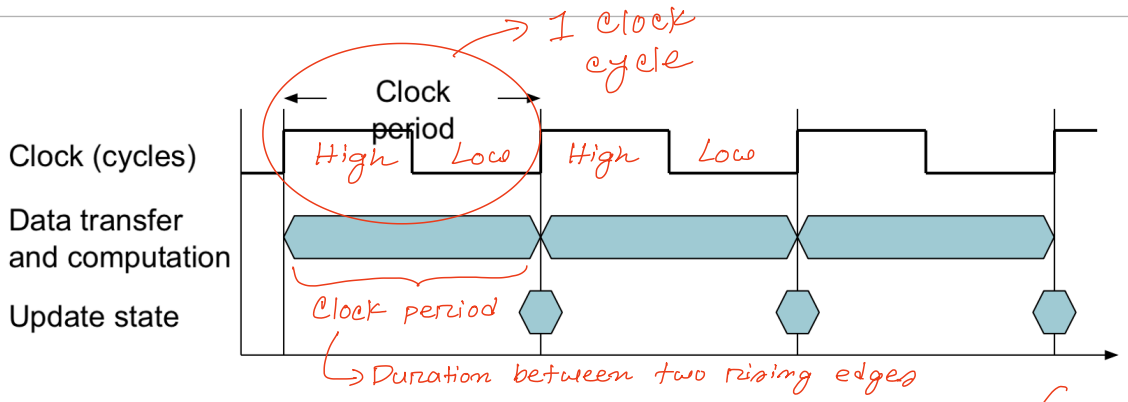
X is n times faster than Y

$$P_X = n \times P_Y$$

$$\Rightarrow \frac{P_X}{P_Y} = n$$

$$\Rightarrow \frac{\frac{1}{E_X}}{\frac{1}{E_Y}} = n$$

$$\Rightarrow \frac{E_Y}{E_X} = n$$



- Clock period: duration of a clock cycle (clock cycle time)
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second (clock Rate)
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time ✓
  - Can do faster clock, but causes 1.2 × clock cycles$_A$
- How fast must Computer B clock be? → Find the new frequency of B? clock Rate

$\text{CPU Time}_B = 6s$

$\text{Clock Cycle}_B = 1.2 \times 20\times10^9$

$= 24\times10^9$

$\text{CPU Time}_B = \dfrac{\text{CPU Clock Cycles}_B}{\text{Clock Rate}_B}$

$\Rightarrow \text{Clock Rate}_B = \dfrac{\text{CPU Clock Cycles}_B}{\text{CPU Time}_B}$

$= \dfrac{24\times10^9}{6}\ \dfrac{1}{s}$

$= 4\times10^9\ s^{-1}$

$= 4\times10^9$ Hz $= 4$ GHz

$\text{Clock Rate}_A = 2$ GHz

$\text{CPU Time}_A = 10s$

$\text{CPU Time}_A = \dfrac{\text{CPU Clock Cycles}_A}{\text{Clock Rate}_A}$

$\text{CPU Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$

$= 10s \times 2$ GHz

$= 10\times2\times10^9 \times s \times$ Hz

$= 20\times10^9 \times s \times \dfrac{1}{s}$

$= 20\times10^9$

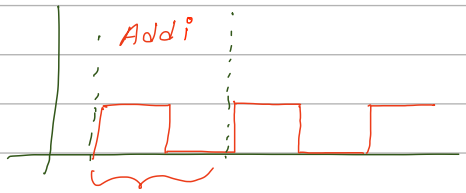
this is the total number of clock cycles for a single task/ program

CPU Time = CPU Clock Cycle × Clock Cycle Time

CPI = Clock Cycle per instruction ??

assembly program to add 1 to 4

```
addi x20, x0, 1   → instruction
addi x20, x20, 2
addi x20, x20, 3
addi x20, x20, 4
```



1 clock cycle

CPU Clock Cycle = 4 × 1

4 → Instruction count

1 → Clock cycle per instruction (CPI)

CPU Time = CPU Clock Cycle × Clock Cycle Time

= Instruction count × CPI × Clock cycle Time

Instruction count for a program, determined by program, ISA, compiler

=> Suppose two programs were compiled using same ISA, does not mean their instruction count is same. It means rules, followed to compile these programs were same.

AVG CPI => Determined by CPU Hardware

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

→ to figure out which computer is faster, we run the same program in both computers

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster...

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

...by this much

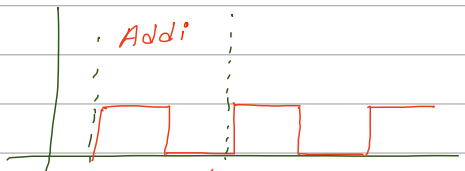
=> Same program and a ISA

So, Instruction count will be same.

Lesser time means faster.

Average CPI

addi
addi
addi
addi
Sub
Sub



1 clock cycle

2 clock cycle

4 add ins.
1 clock cycle per add ins.

2 Sub ins.
2 clock cycles per Sub ins.

We need to find the Average CPI

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

- Weighted average CPI

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

Relative frequency

type of instructions

Class	Add	Sub B	Mul C
CPI for class	1	2	3
IC in sequence 1	2 ✓	1 ✓	2 ✓
IC in sequence 2	4 ✓	1 ✓	1 ✓

Total IC = 5

Clock cycles = $(2\times1 + 1\times2 + 2\times3)$

$= 10$

$\therefore$ Avg CPI = $\frac{10}{5} = 2$

Total IC = 6

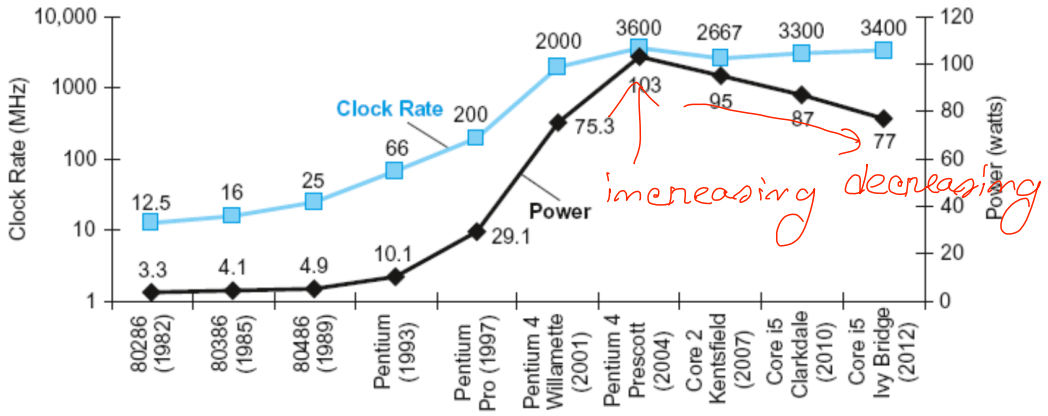
Clock cycles = $(4\times1 + 1\times2 + 1\times3)$

$= 9$

$\therefore$ Avg CPI = $\frac{9}{6} = 1.5$

CPU Time = Instruction count of a program X clock cycles per instruction x Duration of a clock cycle.

Power Trends



- In CMOS IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- The power wall
  - We can't reduce voltage further
  - We can't remove more heat
- How else can we improve performance?

Microarchitecture Enhancement:

↳ Cache optimization – increase the size & efficiency of cache

↳ Specialized execution units – GPUs for specific task.

Software optimization :

Energy efficient Computing:

↳ power gating – turn off the unused component of CPU

Heterogeneous Computing:

↳ Combine different types of cores. (high perf. + energy eff.)

Advanced cooling and packaging.

SPEC — Corporation

Standard | Performance Evaluation

SPEC CPU Benchmark is a standardized set of tests used to measure and compare the perform. of computer processors.

$$\text{Spec ratio} = \frac{\text{Referenece time}}{\text{Execution time}}$$

$$\text{Geometric mean} = \underbrace{(SR1 \times SR2 \times \dots \times SRn)^{\frac{1}{n}}}$$

this value tells that, on average the CPU is about n times faster than the referenece system across the tested workloads.

Amdahl's Law

this law helps us to understand the overall performance improvement gained by optimizing a single part of a system.

$$T_{improved} = \frac{T_{affected}}{\text{improvement factor}} + T_{unaffected}$$

Mips — second
per
Millions of Instructions

1 Mips

= 1 million instructions executed per second.

12 ins. executed in 24 s

$\Rightarrow$ 24 s ——— 12 ins

1 s ——— $\frac{12}{24}$ ins.

$= 0.5$ "

10,00000 ins = 1 Million ins

$\therefore$ 1 " $= \frac{1}{1000000}$ "

$\therefore 0.5$ " $= \frac{0.5}{1000000} = \frac{0.5}{10^6}$