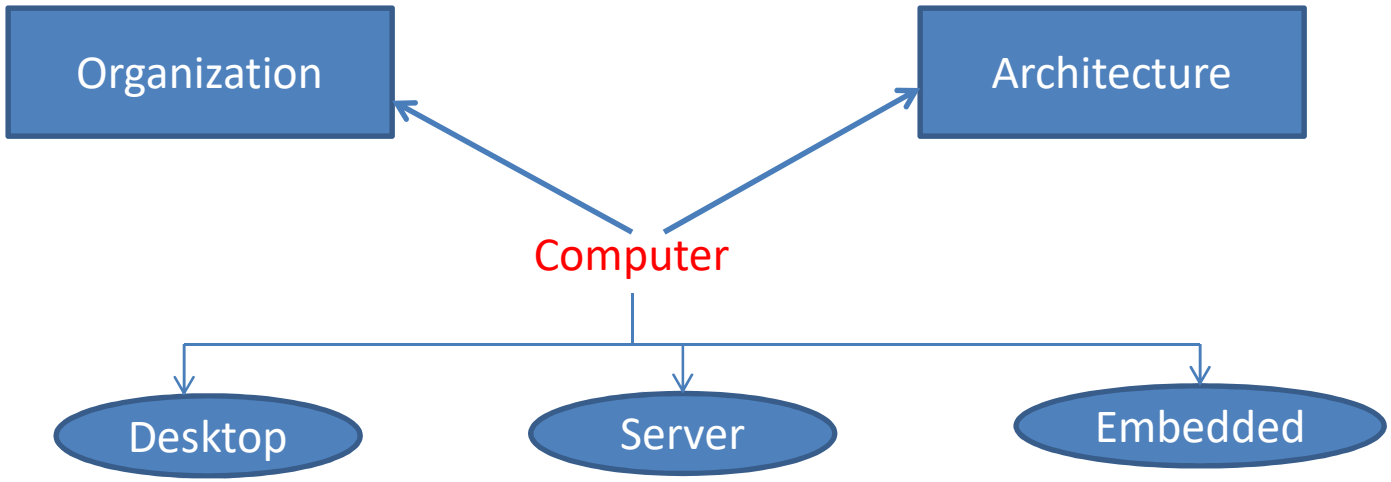
Introductory Lecture - COA

S.Venkatesan
Network Security and Cryptography Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
venkat@iiita.ac.in

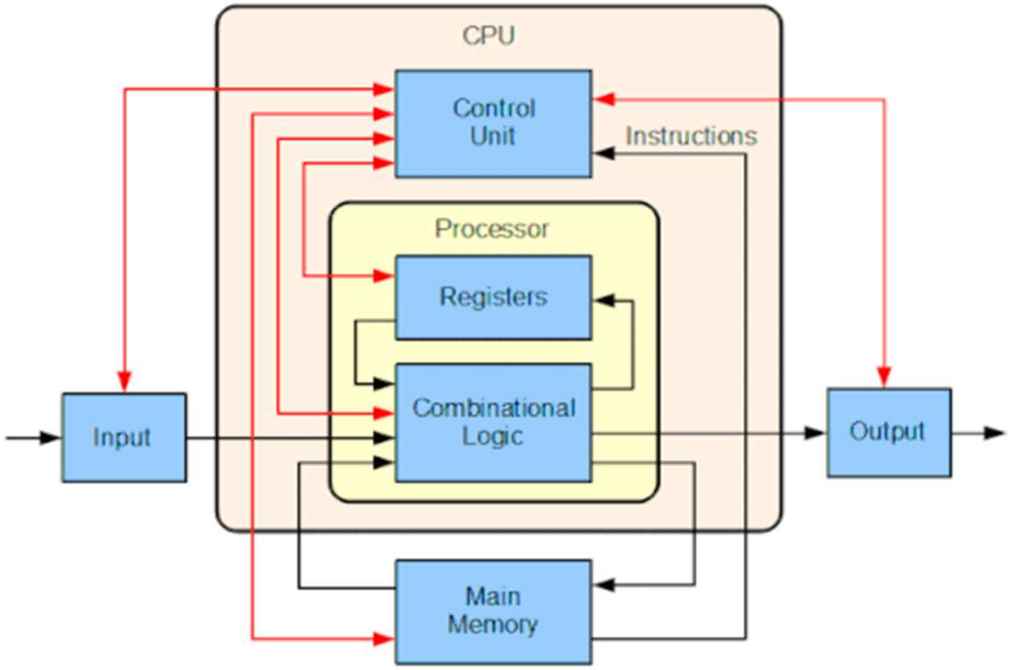
Acknowledgement: The contents and figures are copied from various sources. Thanks to all authors and sources made those contents public and usable for educational purpose

COA?



- Architecture: High Level, What a computer should do
  - Instruction set,
  - Number of bits used for data representation,
  - I/O Mechanisms
  - Addressing Techniques
- Organization: Low Level, How a computer do, Interconnection among units
  - Control signals,
  - Interfaces,
  - Memory Technology

Functional Block of Computer



Performance

- In FLOPs
- Multi-Core
- RAM

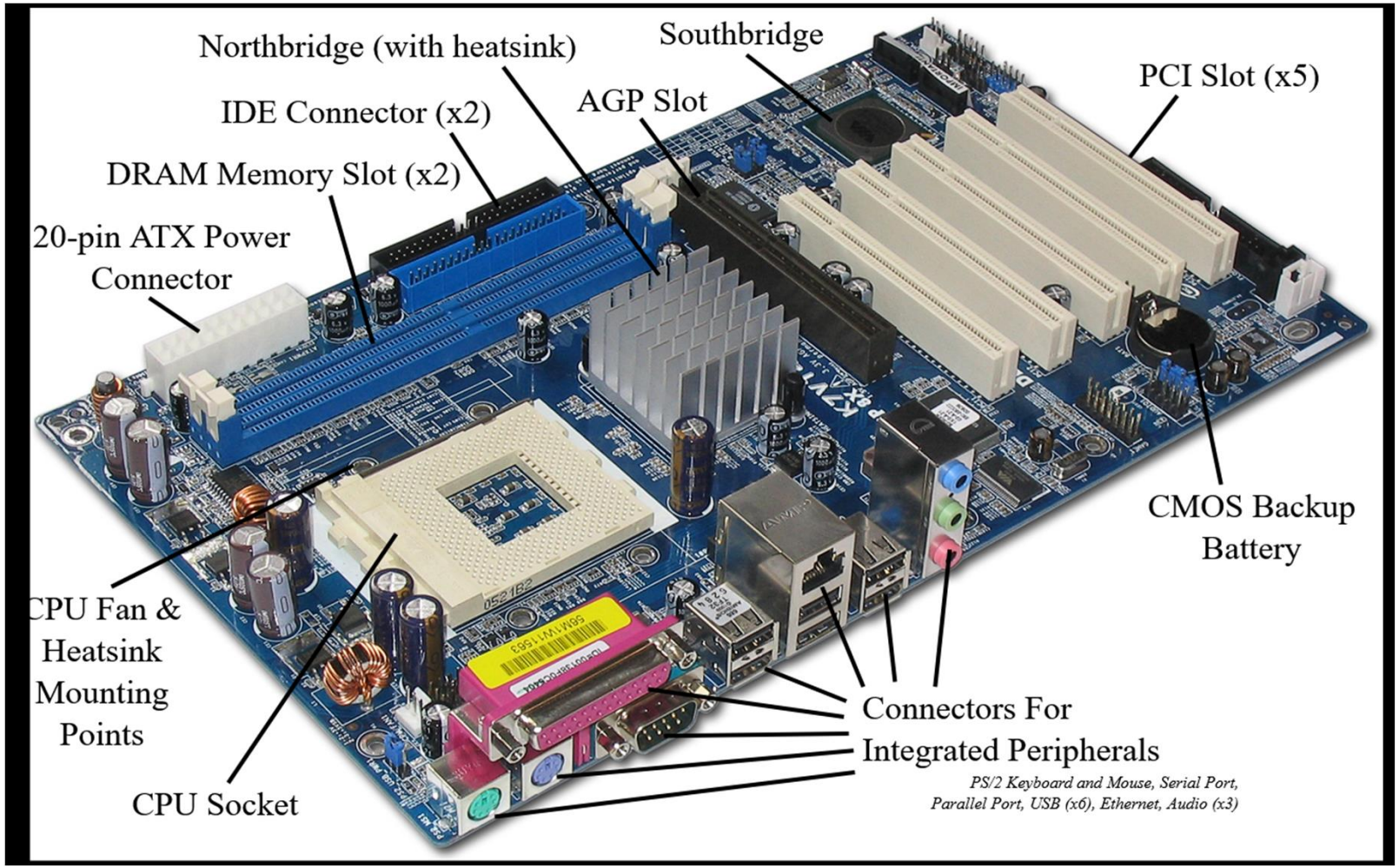
Year	Technology	Performance
1951	Vacuum Tube	1
1965	Transistor	35
1975	IC	900
1995	VLSI Circuit	2400000
2005	ULSI Circuit	6200000000

CPU Execution Time – User CPU Time and System CPU Time

Generation

Generations of computers	Generations timeline	Evolving hardware
First generation	1940s-1950s	Vacuum tube based
Second generation	1950s-1960s	Transistor based
Third generation	1960s-1970s	Integrated circuit based
Fourth generation	1970s-present	Microprocessor based
Fifth generation	The present and the future	Artificial intelligence based

Motherboard

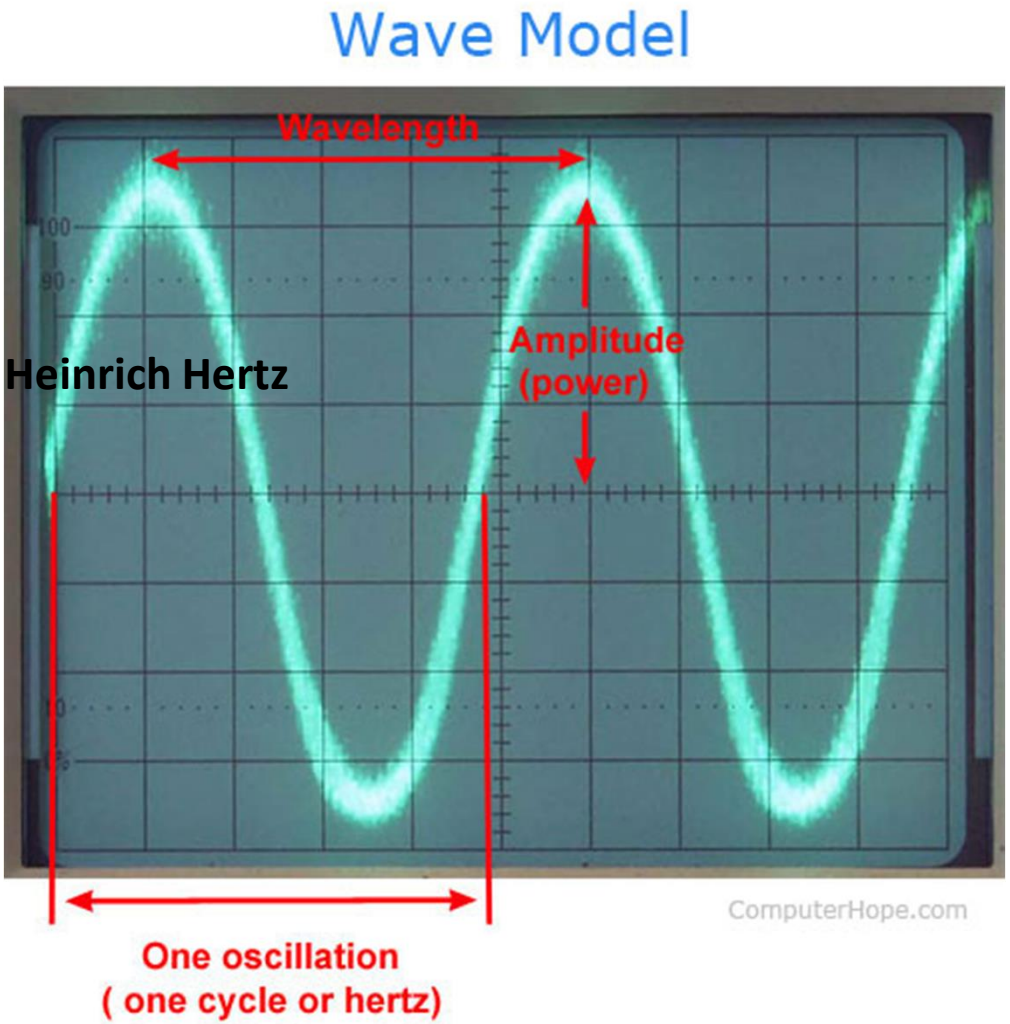


Source: https://turbofuture.com/computers/the-motherboard-components

Computer processor's speed

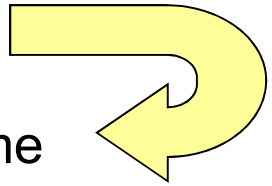
- Named after **Heinrich Hertz** and abbreviated as **Hz** or illustrated as the *f* symbol, **hertz** equals one cycle per second, measuring the waves or frequencies of electric changes each second.

- Hertz is commonly used to measure a computer monitor's refresh rate and a computer processor's speed.



Concepts of Performance and Speedup

Performance = 1 / Execution time



is simplified to

Performance = 1 / CPU execution time

(Performance of M_1) / (Performance of M_2) = Speedup of M_1 over M_2
= (Execution time of M_2) / (Execution time M_1)

Terminology: M_1 is *x* times **as fast as** M_2 (e.g., 1.5 times as fast)
M_1 is 100(*x* – 1)% **faster than** M_2 (e.g., 50% faster)

CPU time = Instructions × (Cycles per instruction) × (Secs per cycle)
= Instructions × CPI / (Clock rate)

Instruction count, CPI, and clock rate are not completely independent, so improving one by a given factor may not lead to overall execution time improvement by the same factor.

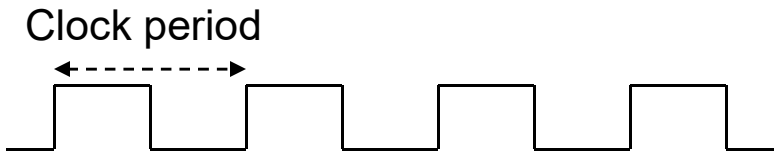
Elaboration on the CPU Time Formula

CPU time = Instructions × (Cycles per instruction) × (Secs per cycle)
= Instructions × Average CPI / (Clock rate)

Instructions: Number of instructions executed, not number of instructions in our program (dynamic count)

Average CPI: Is calculated based on the dynamic instruction mix and knowledge of how many clock cycles are needed to execute various instructions (or instruction classes)

Clock rate: 1 GHz = 10^9 cycles / s (cycle time 10^{-9} s = 1 ns)
200 MHz = 200 × 10^6 cycles / s (cycle time = 5 ns)



Dynamic Instruction Count

How many instructions are executed in this program fragment?

Each "for" consists of two instructions: increment index, check exit condition

12,422,450 Instructions

250 instructions
for i = 1, 100 **do**
20 instructions
  for j = 1, 100 **do**
  40 instructions
    for k = 1, 100 **do**
    10 instructions
    endfor
  endfor
endfor

Static count = 326

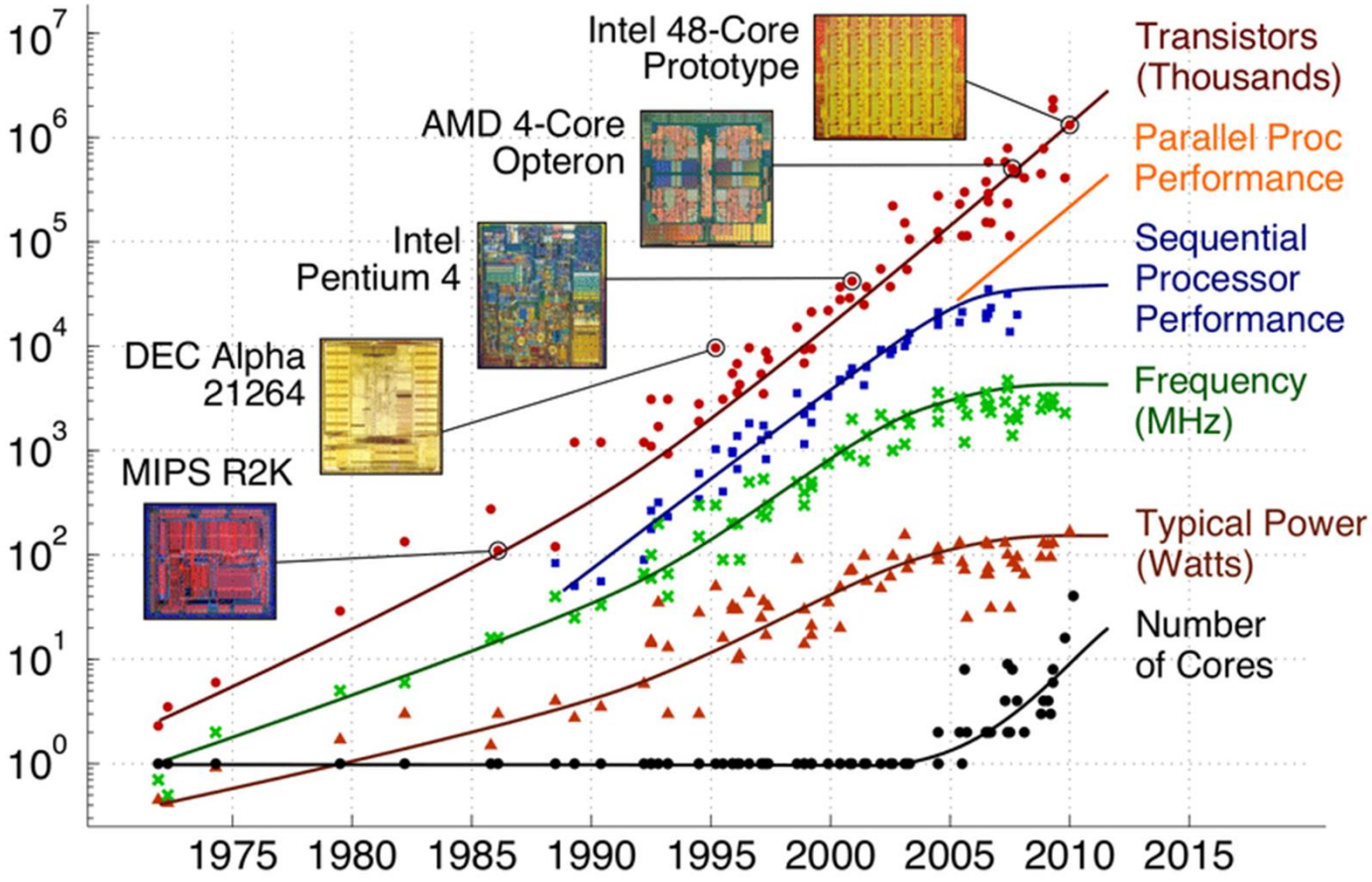
2 + 20 + 124,200 instructions
100 iterations
12,422,200 instructions in all

2 + 40 + 1200 instructions
100 iterations
124,200 instructions in all

2 + 10 instructions
100 iterations
1200 instructions in all

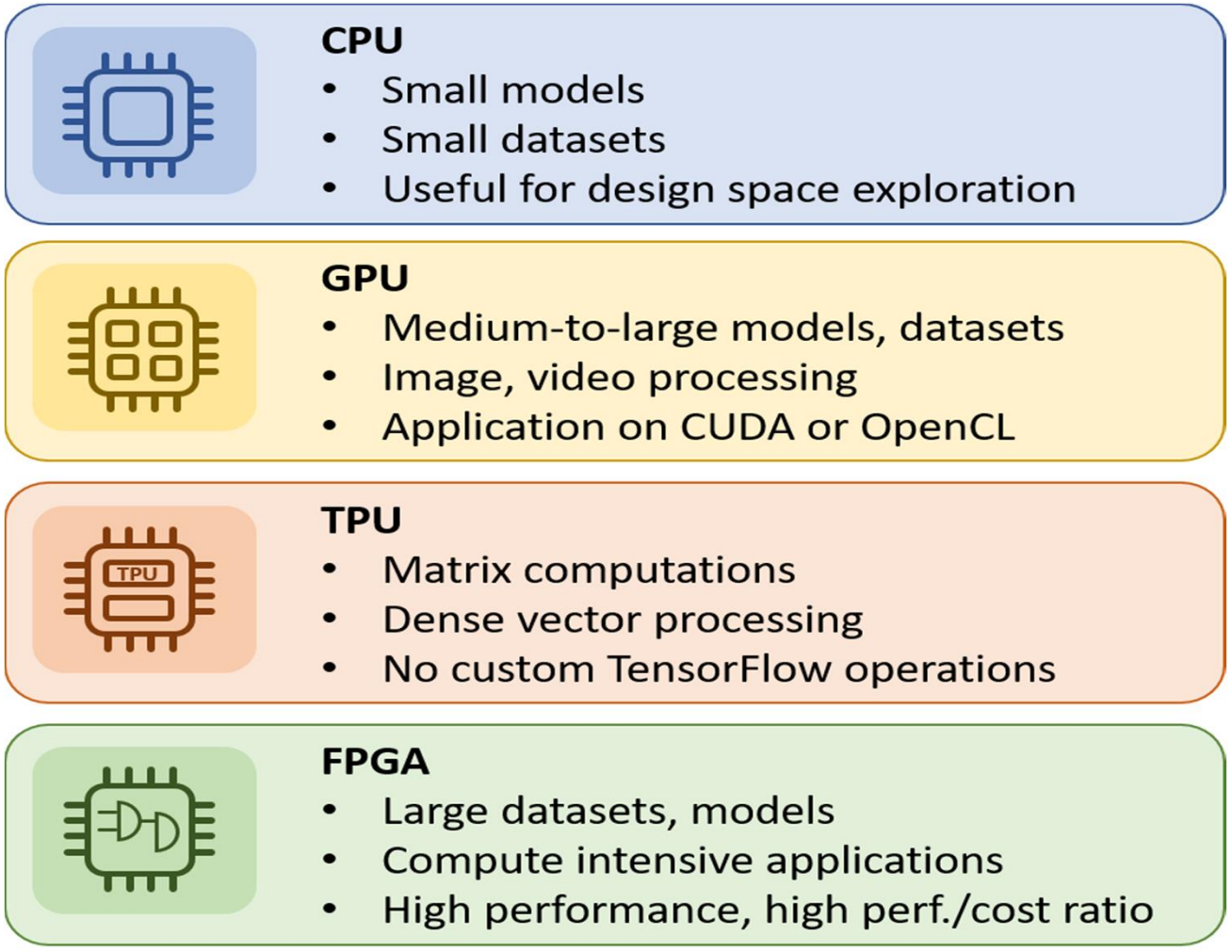
for i = 1, n
while x > 0

Processors

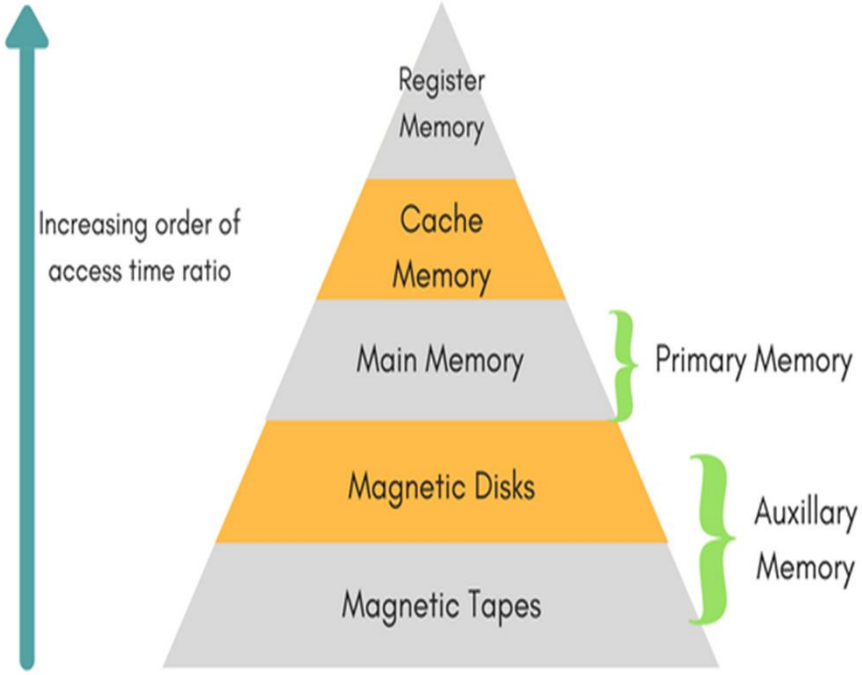


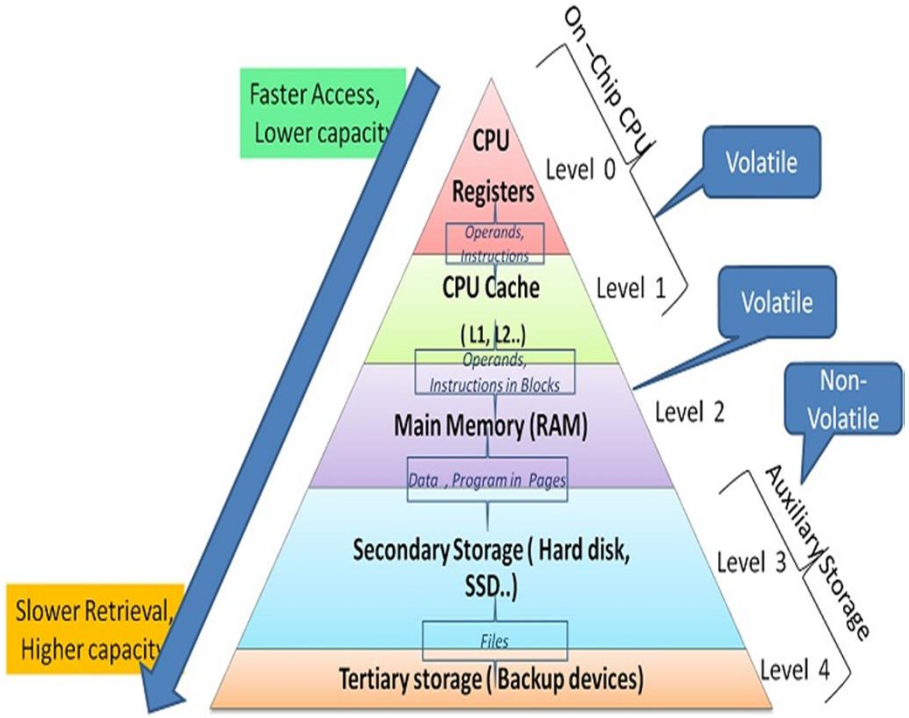
Ref: Pranav Tendulkar
Mapping and Scheduling on Multi-core Processors using SMT Solvers

Processing Units



Memory Hierarchy





What happens to your program

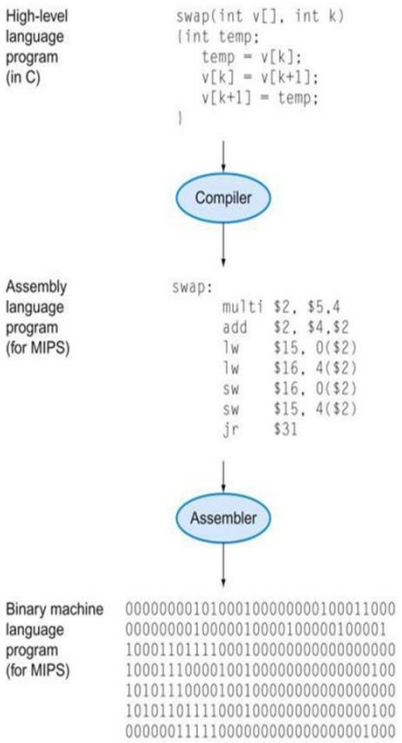


FIGURE 1.4 C program compiled into assembly language and then assembled into binary machine language. Although the translation from high-level language to binary machine language is shown in two steps, some compilers cut out the middleman and produce binary machine language directly. These languages and this program are examined in more detail in Chapter 2.

Thank You