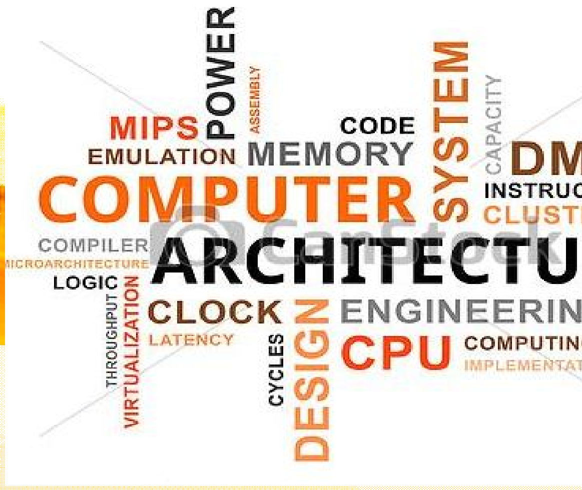
UNIT V
I/O ORGANIZATION AND PARALLELISM

Accessing I/O devices – Interrupts – Direct Memory Access – Buses–Interface circuits – Standard I/O Interfaces (PCI, SCSI, USB) –Instruction Level Parallelism : Concepts and Challenges – Introduction to multicore processor – Graphics Processing Unit



Recap the previous Class

Introduction

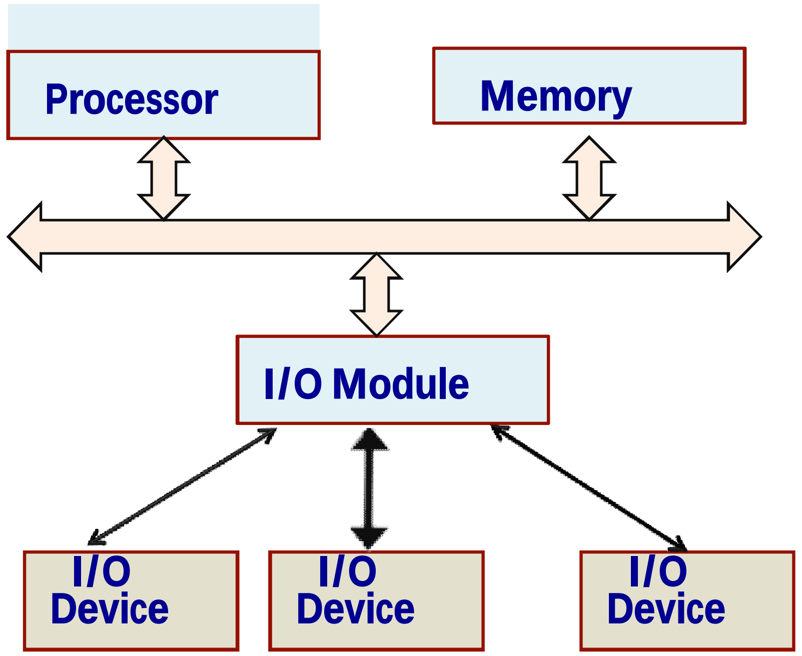
- Interfacing input/output devices is **more complex** as compared to interfacing memory systems.
- **Why?**
  - Wide variety of peripherals (keyboard, mouse, disk, camera, printer, scanner, etc.).
  - Widely varying speeds.
  - Data transfer rate can be regular or irregular.
  - Sizes of data blocks transferred at a time varies widely (few bytes to Kbytes).
- Slower than processor and memory.

Input / Output Interface

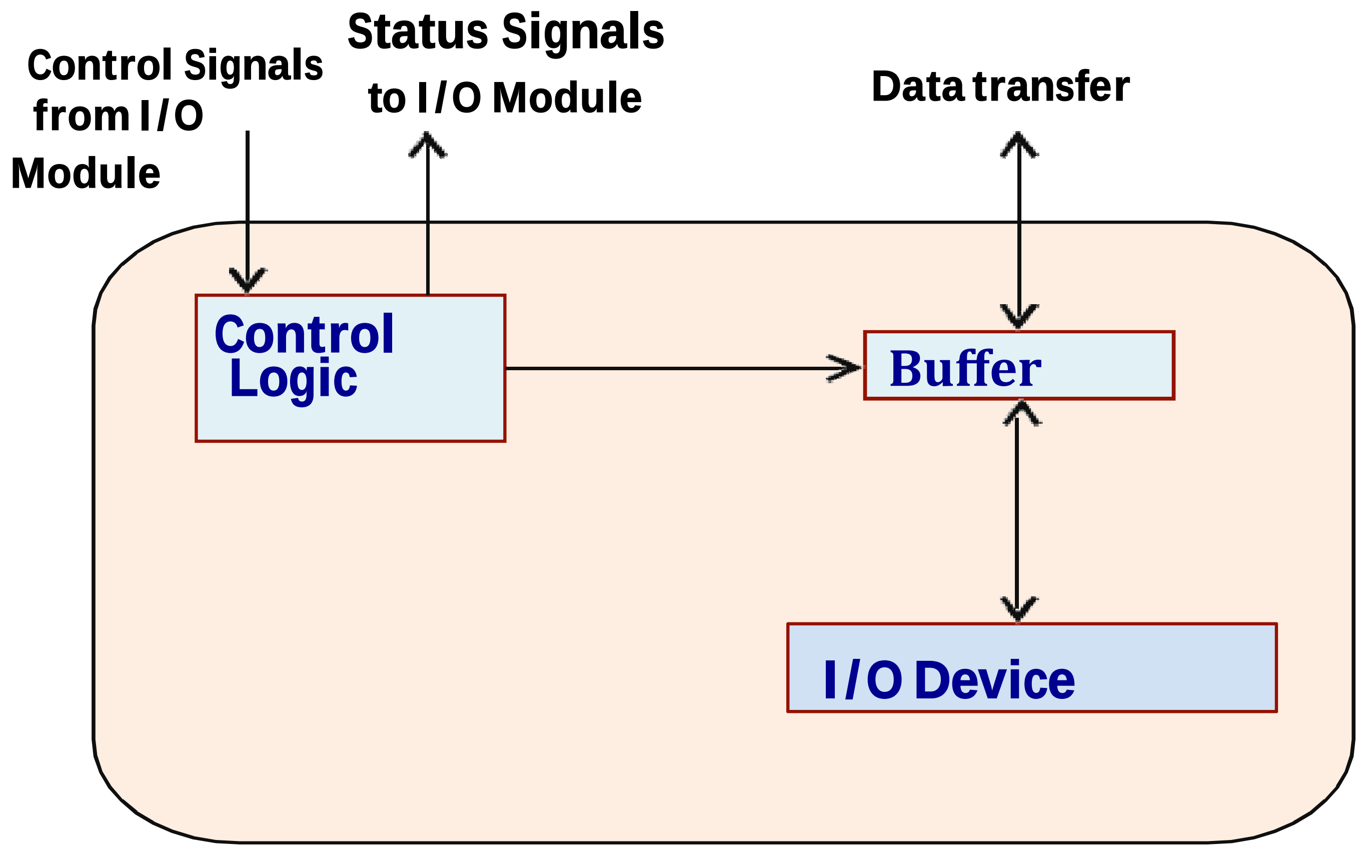
- To handle widely different types of I/O devices, we need a programmable I/O interface or *I/O module*.

– Interfaces to processor and memory on one side.

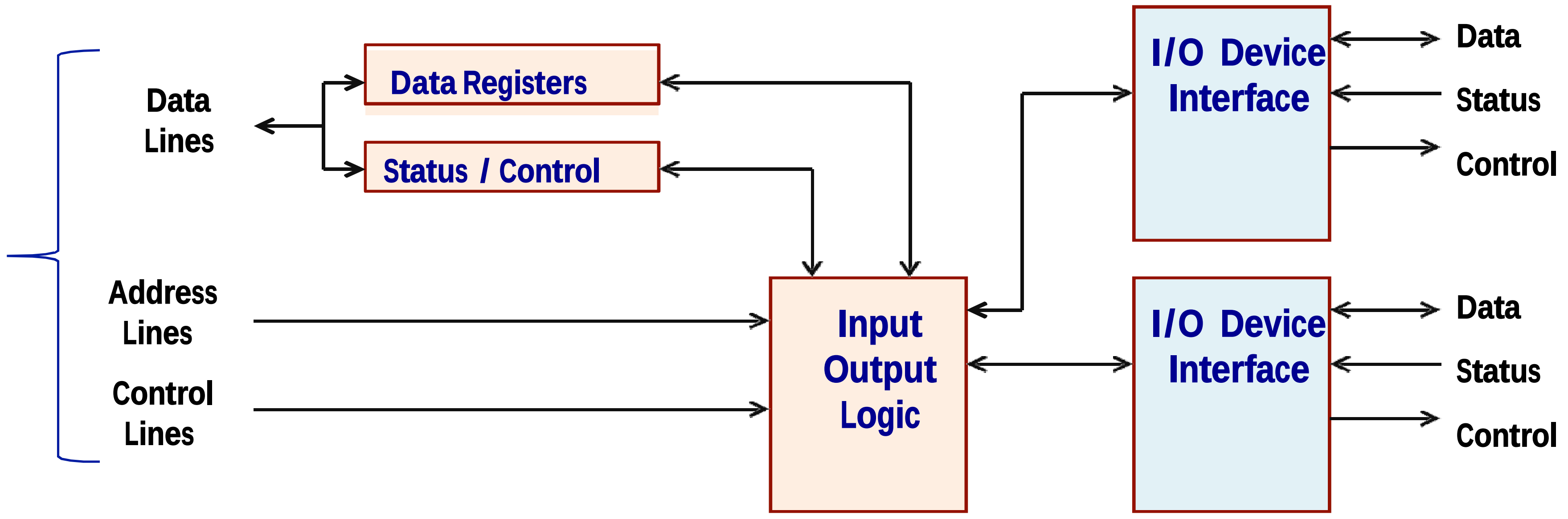
– Interfaces to one or more peripheral devices on the other side.



Typical I/O Device Interface



I/O Module Schematic



Typical Steps During I/O

a) Processor requests the I/O Module for device status.

b) I/O Module returns the status to the processor.

c) If the device is ready, processor requests data transfer.

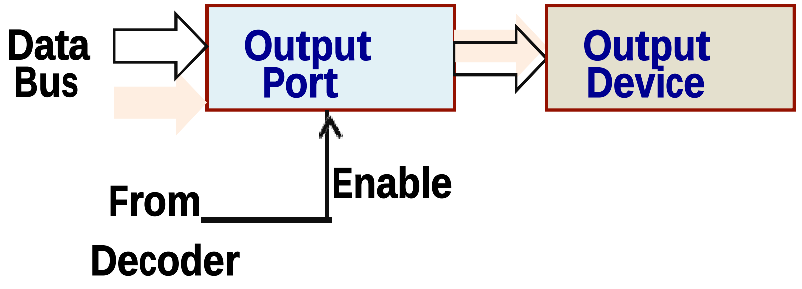
d) I/O Module gets data from device (say, input device).

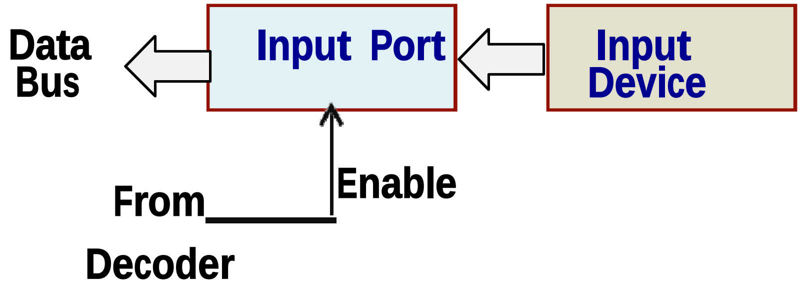
e) I/O Module transfers data to the processor.

f) Processor stores the data in memory.

How are I/O devices typically interfaced?

- **Through input and output ports.**
- **Output port:**
  - The register inputs are connected to the data bus, and the register outputs are connected to the output device.
- **Input port:**
  - The driver outputs are connected to the data bus, while the inputs are connected to the input device.





Memory-Mapped and I/O Mapped Device Interface

- **Memory-mapped device interface:**
  - The same address decoder selects memory and I/O ports.
  - Some of the memory address space is occupied by I/O devices.
  - All data transfer instructions to/from memory can be used to transfer data to/from I/O devices.
  - The processor need **not have separate instructions for I/O**, nor it need to specify whether an address generated by the CPU is a memory address or an I/O address.

- **I/O mapped device interface:**
  - –**Separate instructions** for I/O data transfer (say, IN and OUT).
  - –A **processor signal identifies** whether a generated address refers to a memory location or an I/O device.
  - –**Separate address decoders** for selecting memory and I/O ports.
  - –The complete memory address space can be utilized.

Data Transfer Techniques

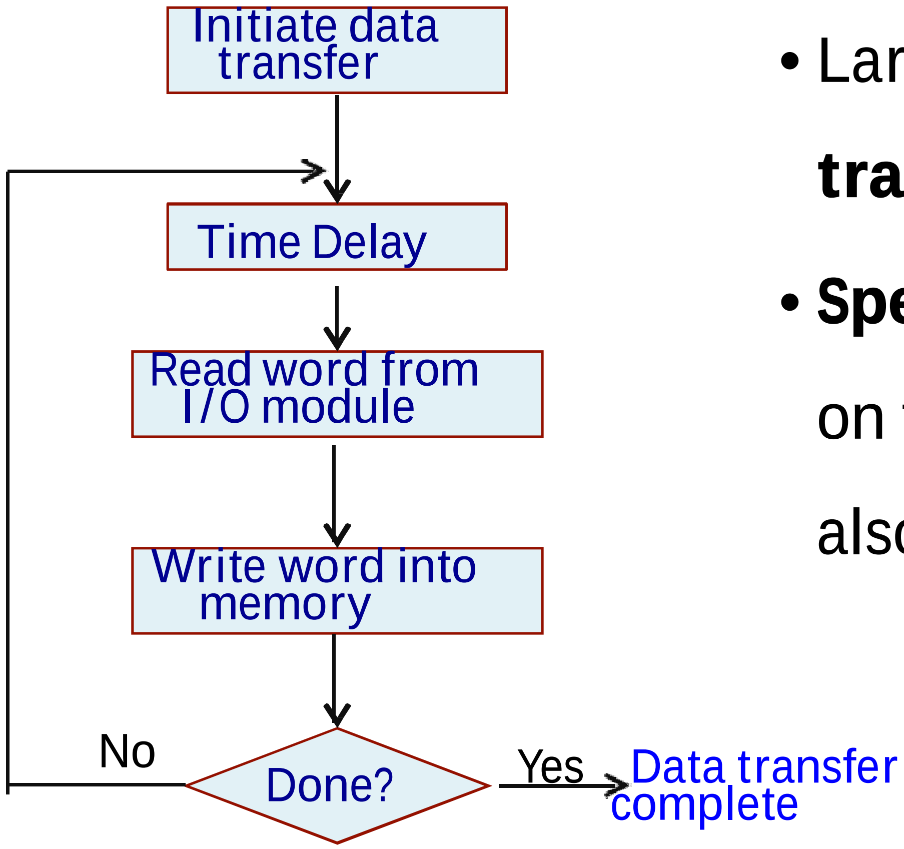
1. Programmed: CPU executes a program that transfers data between I/O device and memory.
   a)Synchronous
   b)Asynchronous
   c)Interrupt-driven
2. Direct Memory Access (DMA): An external controller directly transfers data between I/O device and memory without CPU intervention.

(a) Synchronous Data Transfer

- The I/O device **transfers data at a fixed rate** that is known to the CPU.
- The CPU **initiates the I/O operation** and transfers successive bytes/words after giving **fixed time delays**.

Characteristics:

- –During the time delay, CPU lies idle.
- –Not many I/O devices have strictly synchronous data transfer characteristics.



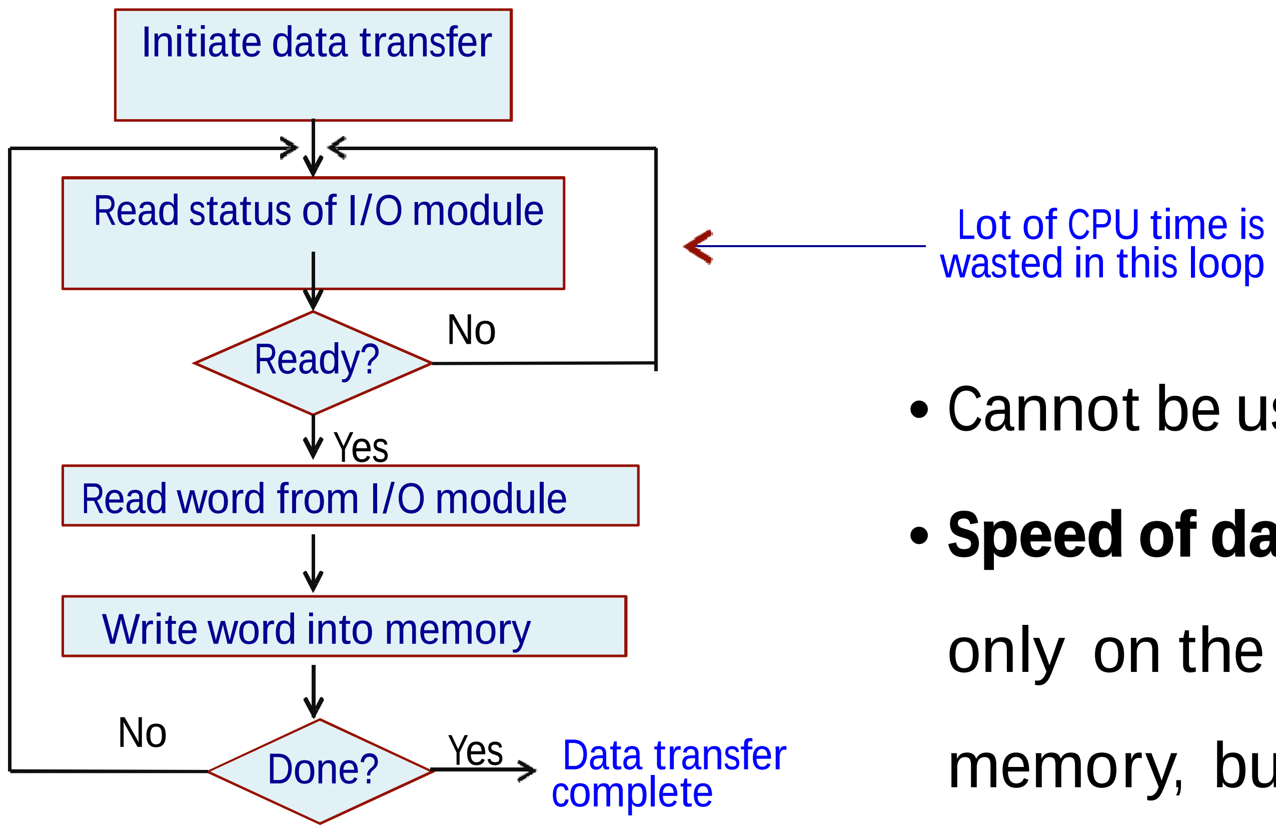
- **Error may occur** if the input device and the processor **get out of synchronization**.
- Large number of words **cannot be transferred** in one go.
- **Speed of data transfer** depends not only on the speed of I/O device and memory, but also on the execution time of the code.

(b) Asynchronous Data Transfer

- The **CPU does not know** when the I/O module will be ready to transfer the next word.
- CPU has to **check the status of the I/O module** to know when the device is ready to transfer the next word.
  - Called *handshaking*.

Characteristics:

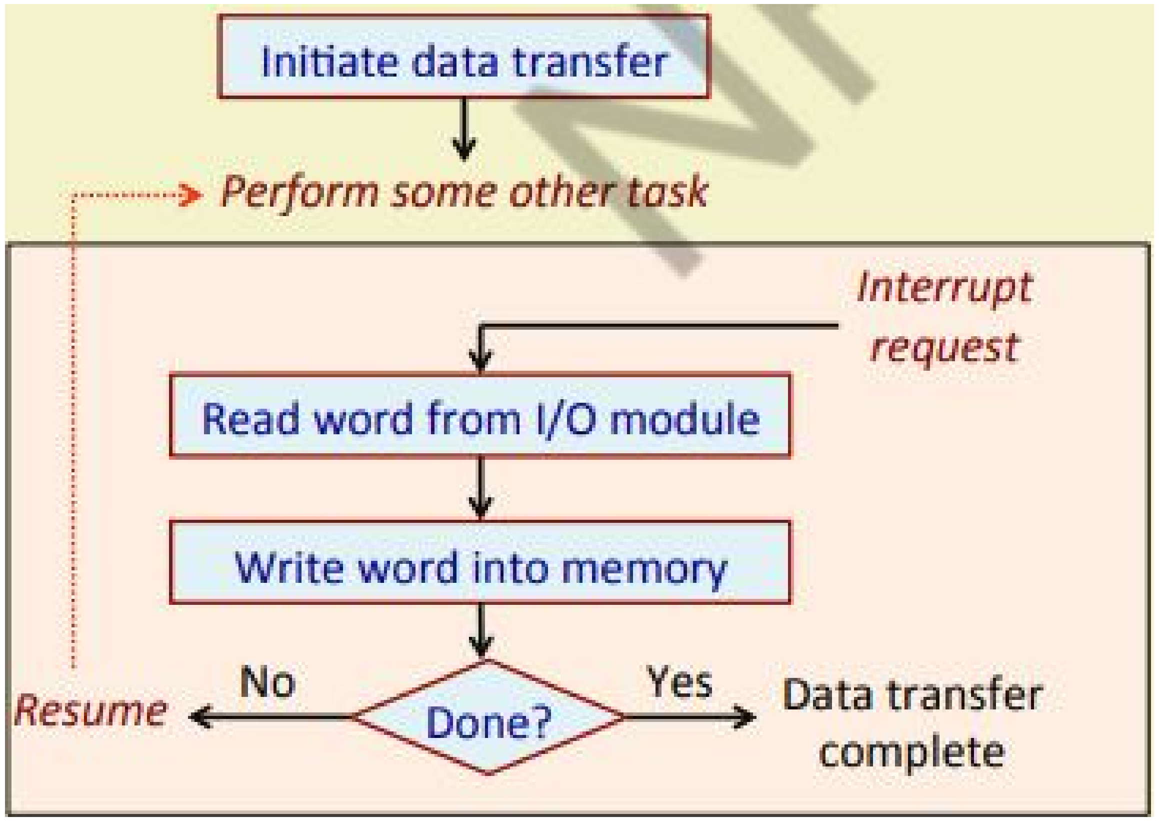
- While the CPU is checking whether the I/O module is ready, it cannot do anything else.
- Wasteful of CPU time for slow devices like keyboard or mouse.


Initiate data transfer
Read status of I/O module
Ready?
No
Yes
Read word from I/O module
Write word into memory
Done?
No
Yes
Data transfer complete
Lot of CPU time is wasted in this loop

- Cannot be used for **high-speed devices**.
- **Speed of data transfer** depends not only on the speed of I/O device and memory, but also on the execution time of the code.

(c) Interrupt-Driven Data Transfer

- The CPU initiates the data transfer and proceeds to perform *some other task*.
- When the I/O module is ready for data transfer, it informs the **CPU by activating a signal** (called ***interrupt request*****).**
- The **CPU suspends the task** it was doing, services the request (that is, carries out the data transfer), and returns back to the task it was doing.
- **Characteristics:**
  - CPU time is **not wasted** while checking the status of the I/O module.
  - CPU time is **required only during data transfer**, plus some overheads for transferring and returning control.



This part of the program that gets activated when interrupt request comes is called interrupt handler or interrupt service routine (ISR).

TEXT BOOK

Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, McGraw-Hill, 6th Edition 2012.

REFERENCES

1. David A. Patterson and John L. Hennessey, “Computer organization and design”, MorganKauffman ,Elsevier, 5th edition, 2014.
2. William Stallings, “Computer Organization and Architecture designing for Performance”, Pearson Education 8th Edition, 2010
3. John P.Hayes, “Computer Architecture and Organization”, McGraw Hill, 3rd Edition, 2002
4. M. Morris R. Mano “Computer System Architecture” 3rd Edition 2007
5. David A. Patterson “Computer Architecture: A Quantitative Approach”, Morgan Kaufmann; 5th edition 2011

THANK YOU