Operating Systems (CS3500)
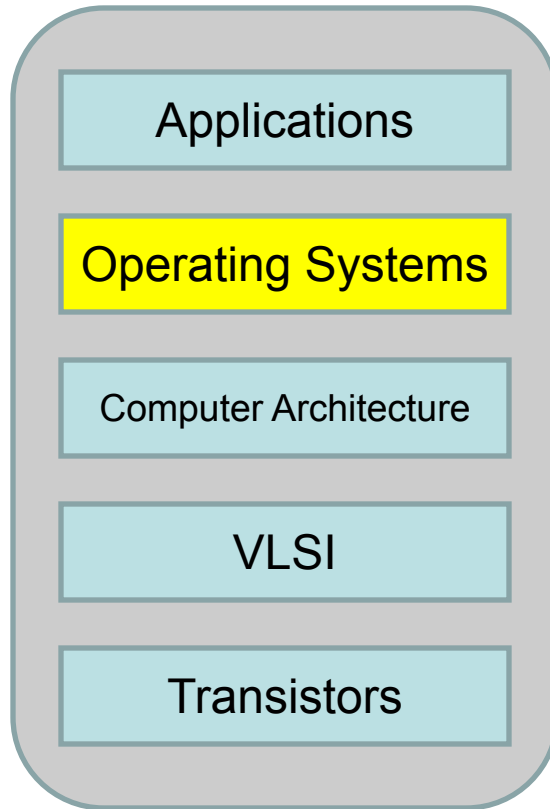
Introduction

Chester Rebeiro

IIT Madras

Layers in a Computer System

| Applications |
| Operating Systems |
| Computer Architecture |
| VLSI |
| Transistors |

Use of Operating Systems

- **Hardware Abstraction**

  turns hardware into something that applications can use

- **Resource Management**

  manage system's resources (security)

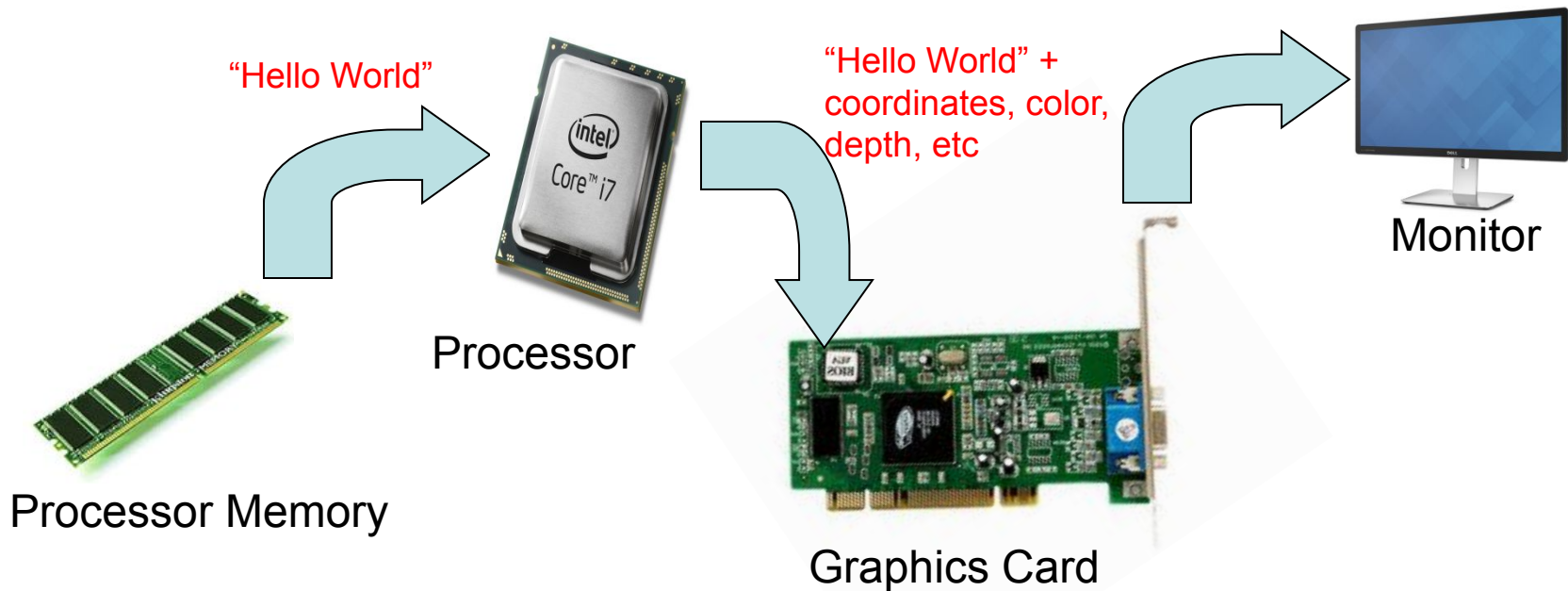# A Simple Program

What is the output of the following program?

```c
#include <stdio.h>

int main(){
    char str[] = "Hello World\n";
    printf("%s", str);
}
```

How is the string displayed on the screen?
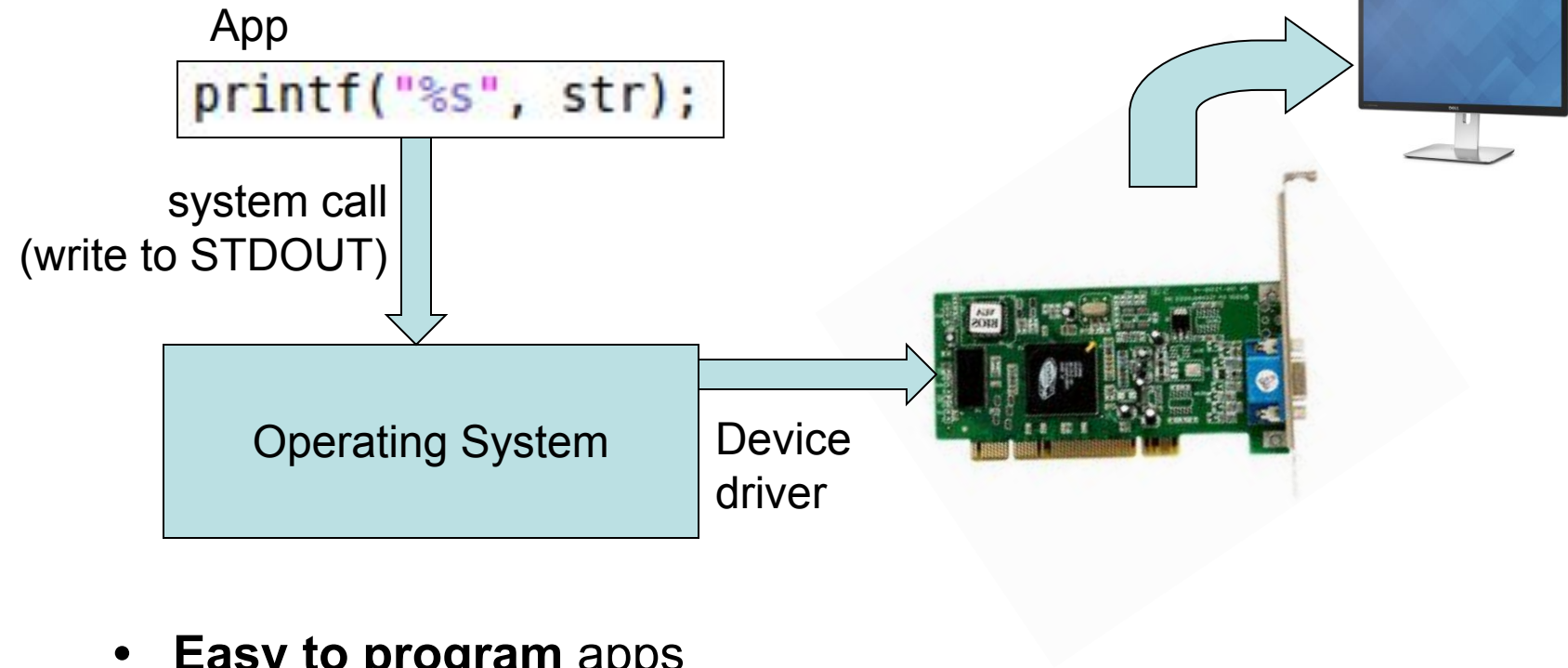
Displaying on the Screen

"Hello World"

Processor

Processor Memory

"Hello World" +
coordinates, color,
depth, etc

Graphics Card

Monitor

- Can be complex and tedious
- Hardware dependent

Without an OS, all programs need to take care of every nitty gritty detail.
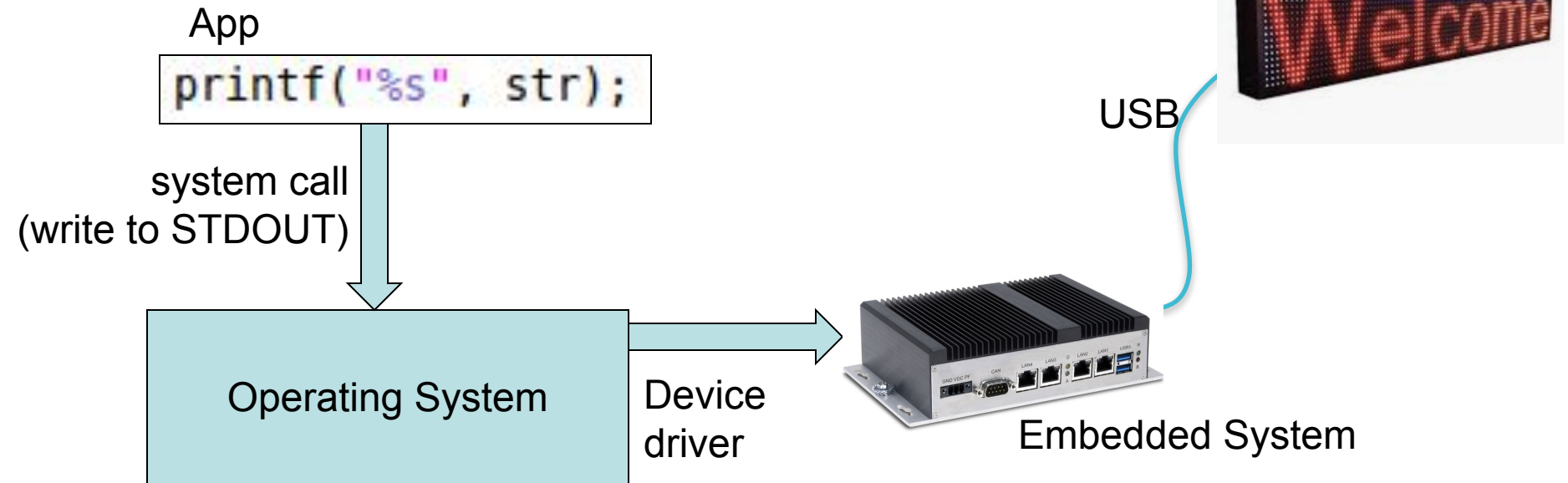
Operating Systems Provide Abstraction

On a desktop

App

```
printf("%s", str);
```

system call
(write to STDOUT)

Operating System

Device driver

- **Easy to program** apps
  - No more nitty gritty details for programmers
- **Reusable functionality**
  - Apps can reuse the OS functionality

Operating Systems Provide Abstraction

On an embedded device

App

```
printf("%s", str);
```

system call
(write to STDOUT)

Operating System
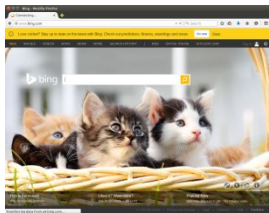
Device
driver

USB

Embedded System

- **Portable**
  - OS interfaces are consistent. The app does not change when hardware changes
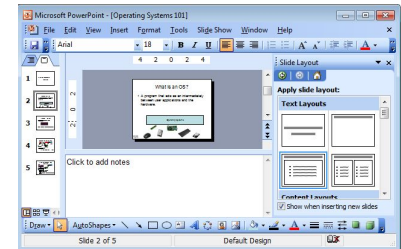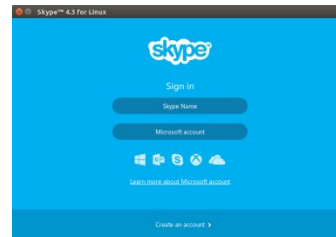
OS as a Resource Manager

- Multiple apps but limited hardware

Apps

Operating Systems

A few processors

OS as Resource Manager

- OS must manage CPU, memory, network, disk etc…

- Resource management
  - allows multiple apps to share resources
  - protects apps from each other
  - Improves performance by efficient utilization of resources

Sharing the CPU

App1	App2	App3	App4

Who uses the CPU?

App1	App2	App3	App4

time
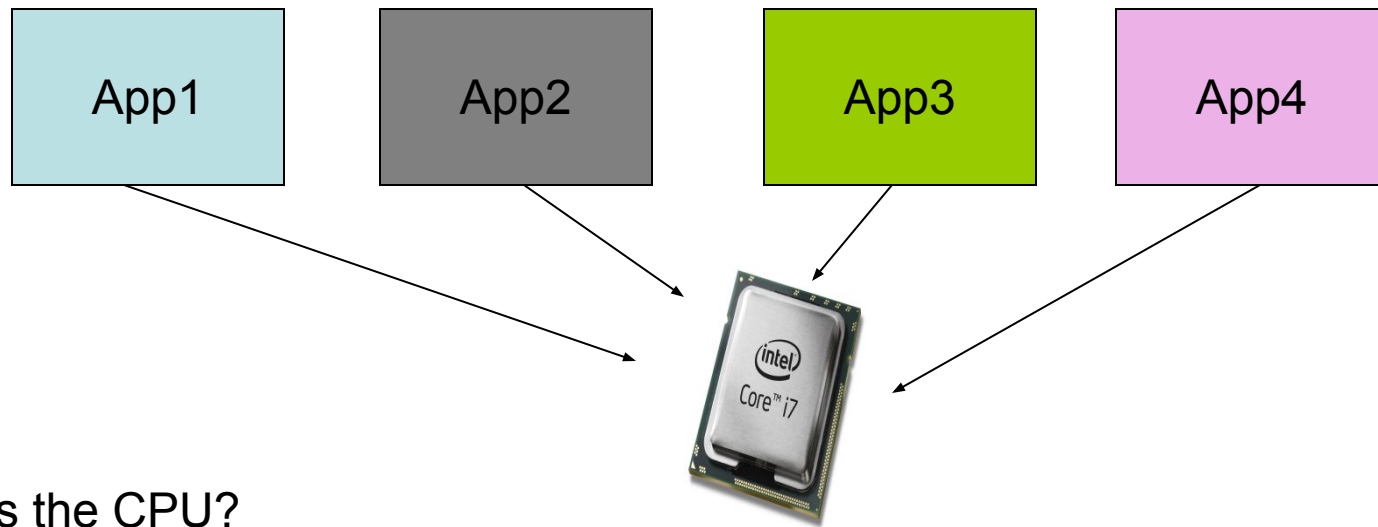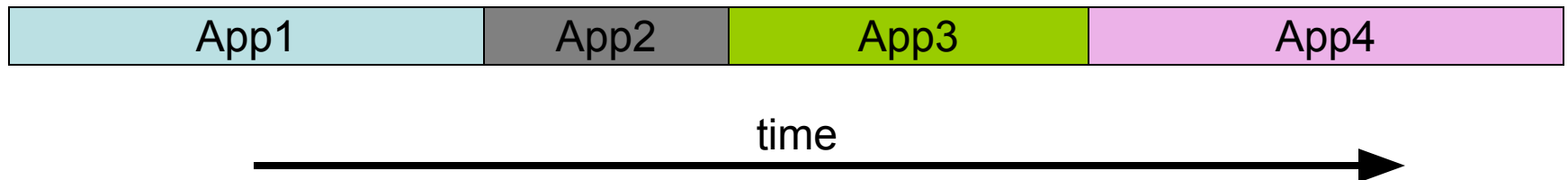
the main job is to let multiple apps run on the same processor
ms dos made it sequential where one app runs after the previous one is done running

Operating Systems Types

- Application Specific
  - Embedded OS
    - eg. Contiki OS, for extremely memory constraint environments
  - Mobile OS
    - Android, iOS, Ubuntu Touch, Windows Touch
  - RTOS
    - QNX, VxWorks, RTLinux
  - Secure Environments
    - SeLinux, SeL4
  - For Servers
    - Redhat, Ubuntu, Windows Server
  - Desktops
    - Mac OS, Windows, Ubuntu, Linux

xv6

- Designed for pedagogical reasons
- Unix like (version 6)
  - Looks very similar to modern Linux operating systems
  - (https://pdos.csail.mit.edu/6.828/2020/xv6.html)
- Hardware : RISC V
  - Open source ISA architecture.

Course Structure

- ## Syllabus
  - Overview of Operating Systems
  - PC Hardware
  - Memory Management
  - Interrupts
  - Context Switching
  - Processes
  - Scheduling
  - Cooperating Processes
  - Synchronization
  - File Systems
  - Security

Textbooks / References

- **"xv6: a simple, Unix-like teaching operating system"**, Revision 8, by Russ Cox, Frans Kaashoek, Robert Morris

- **"Operating System Concepts"**, 8th edition, by Adraham Silberschatz, Pert B. Galvin, and Greg Gagne, Wiley-India edition

- **"Operating Systems: Three Easy Pieces"**, Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau (University of Wisconsin-Madison) https://pages.cs.wisc.edu/~remzi/OSTEP/

- The xv6 source code

Logistics

Classes will be held from 28/July/2025 in <u>Slot</u> B.

- Monday : 09:00 - 09:50 AM

- Tuesday : 08:00 - 08:50 AM

- Friday : 12:00 - 12:50 PM

- Tuesday (Q slot): 14:00 - 16:45 PM (will be used for Lab and Tutorials)

Slides / Videos / Schedule

- Slides and Schedule will be put up on course webpage:
  - https://sites.google.com/cse.iitm.ac.in/os-2025/home

- Assignments will need to be submitted in Moodle

Grading

- Q1: 10%
- Q2: 10%
- End Sem: 40% (entire syllabus)
- Lab: 40%

Attendance

- Attendance as per institute norms.

Operating Systems

(How did it all start?)

The lab is going to be identical to the MIT course
The idea of this lab is to modify the OS itself and serve as application of the CoA course
This is different from the CoA course in being closer to software

OS Evolution

- **Evolution driven by Hardware improvements + User needs**

  - eg. low power requirements, Increased / reduced security, lower latency

  - Evolution by

    - New/better abstractions
    - New/better resource management
    - New/better low level implementations

Gen 1: Vacuum Tubes

- **Hardware**
  - Vacuum tubes and IO with punchcards
  - Expensive and slow

- **User Apps**
  - Generally straightforward numeric computations done in machine language
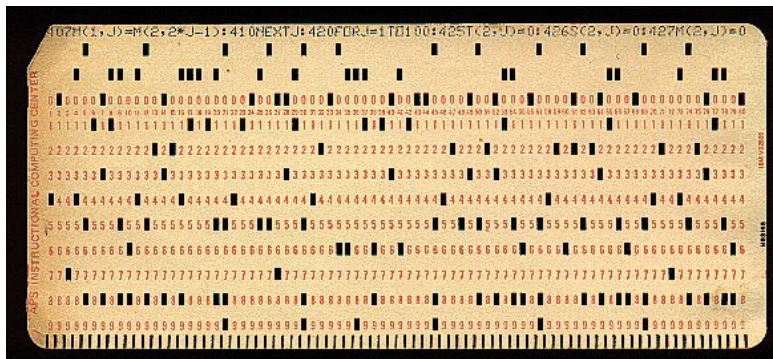
IBM Punch card

ENIAC

Gen 1 : OS

- OS: Unheard of
- Human feeds program and prints output

George Ryckman, on IBM's first computer

Each user was allocated a minimum 15-minute slot, of which time he usually spent 10 minutes in setting up the equipment to do his computation ... By the time he got his calculation going, he may have had only 5 minutes or less of actual computation completed—wasting two thirds of his time slot.

The cost of wastage was $146,000 per month (in 1954 US Dollars)

Gen 2 : Mainframes

- Hardware
  - transistors
- User Programs
  - Assembly or Fortran entered using punch cards
- OS : Batch systems
  - Possibly greatest invention in OS
    - Computers may be able to schedule their own workload by means of software

Batch Systems
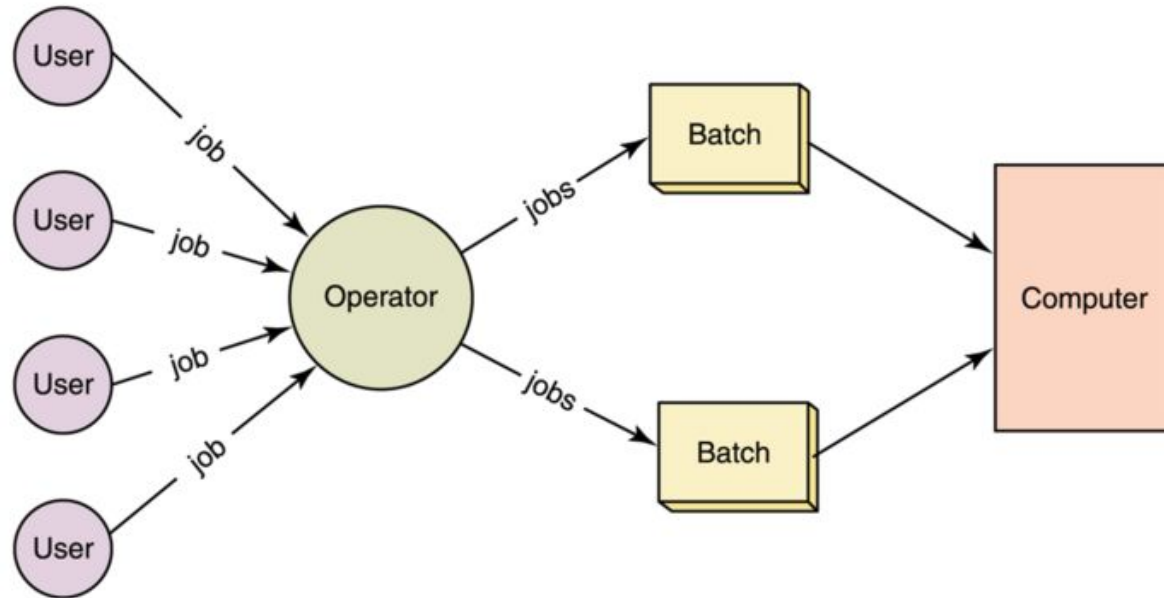
- Operator collects jobs (through punch cards) and feeds it into a magnetic tape drive
- Special Program reads a job from input tape drive and on completion writes result to output tape drive
- The next program is then read and executed
- Printing was done offline

Batch Systems (pros.)

- Pros
  - Better utilization of machine

Batch Systems (cons.)

- In Batch Systems execute time includes reading from input and writing to output.
- I/O considerably slower than execution
  - Magnetic tapes were best read sequentially
- Therefore programmer must wait for long time

| Input Magnetic Tape	→	CPU	→	Output Magnetic Tape
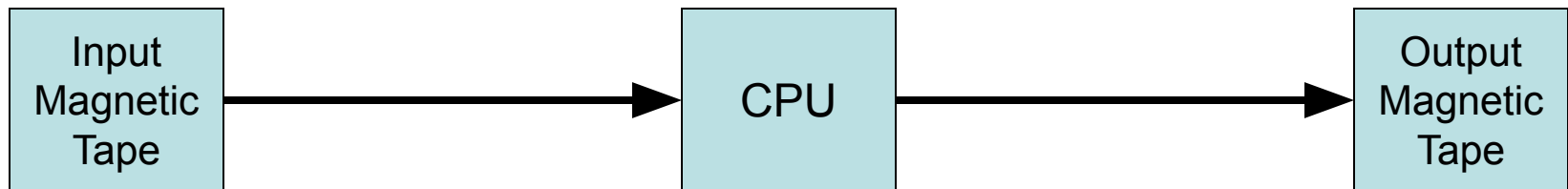
Gen 3 : Mini computers

- **Hardware**
  - SSI/MSI/LSI ICs
  - Random access memories
  - Interrupts (used to simulate concurrent execution)
- **User Programs**
  - High level languages (Fotran, COBOL, C, …)
- **Operating Systems**
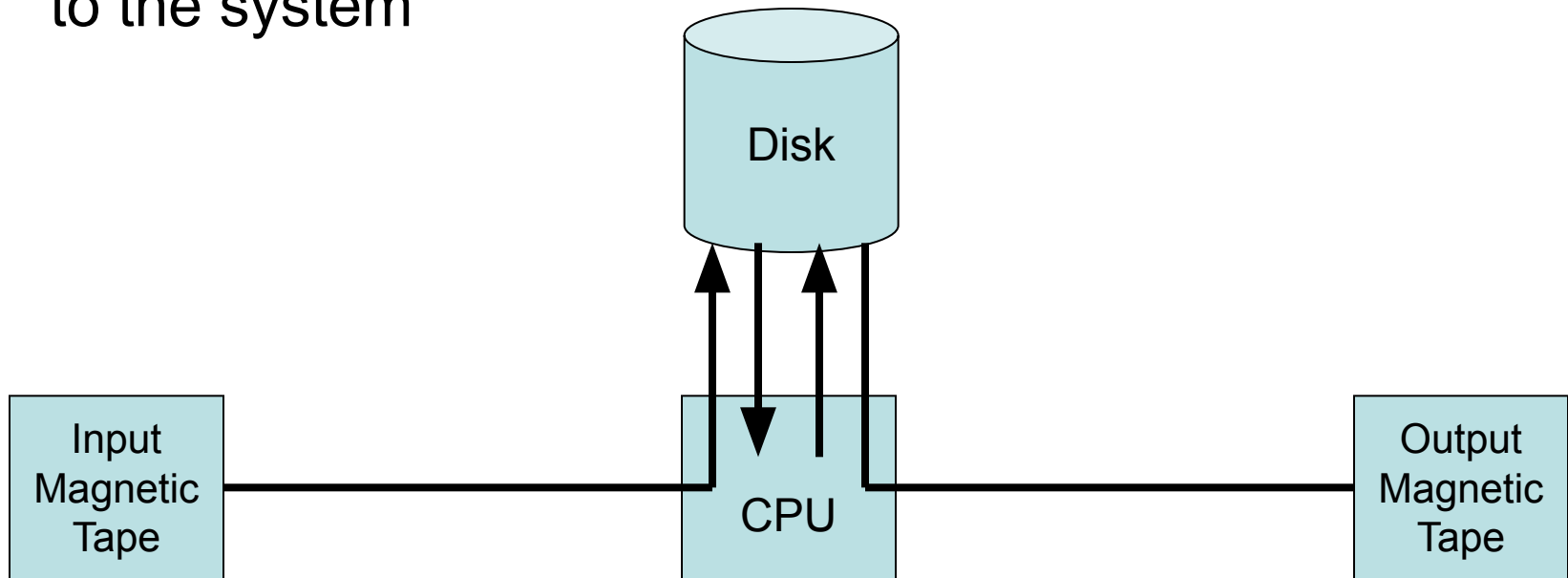 - Multiprogramming
  - Spooling

Multiprogramming

- Multiple jobs in memory
  - When one waits for I/O the next job executes

- OS controls
  - scheduling of jobs
  - Protection between jobs

OS
Job 1
Job 2
Job 3

Memory partitions
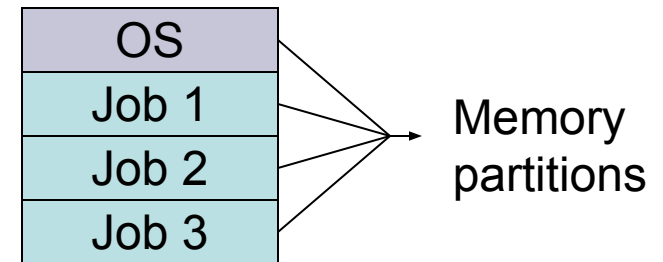
Multiprogramming with
3 jobs in memory

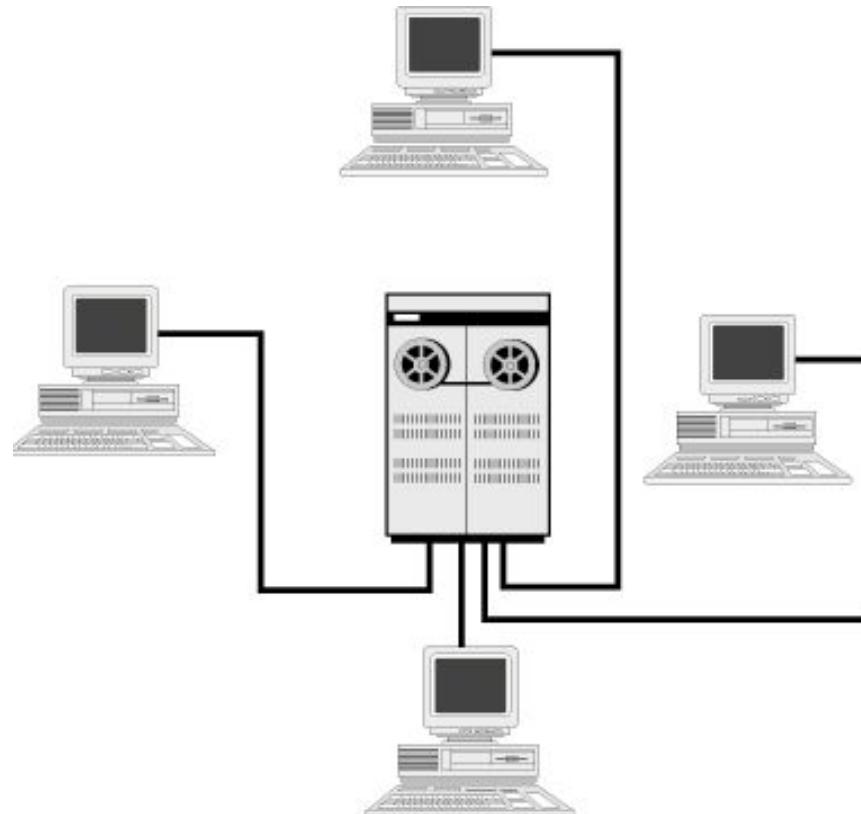# Spooling

- Uses buffers to continuously stream inputs and outputs to the system

- Pros : better utilization / throughput
- Cons : still not interactive

Timesharing

Who uses the CPU?

Terminal 1	Terminal 2	Terminal 3	Terminal 4

time →

Timesharing

John McCarthy, 1962

By a time-sharing computer system I shall mean one that interacts with many simultaneous users through a number of remote consoles. Such a system will look to each user like a large private computer...When the user wants service, he simply starts typing in a message requesting the service. The computer is always ready to pay attention to any key that he may strike.

Because programs may ... do only relatively short pieces of work between human interactions, it is uneconomical to have to shuttle them back and forth continually to and from secondary storage. Therefore, there is a requirement for a large primary memory ... The final requirement is for secondary storage large enough to maintain the users' files so that users need not have separate card or tape input-output units.

Multics, 1964

- Multiplexed Information and Computing Service
- Ambitious project started in MIT
- Introduced several new OS features but was not successful by itself
  - Segmented and Virtual memory
  - High level language support
  - Multi language support
  - **Security**
  - File system hierarchies
  - Relational databases
  - Shared memory multiprocessor

Gen 4 : Personal Computers

- Hardware
  - VLSI ICs
- User Programs
  - High level languages
- Operating Systems
  - Multi tasking
  - More complex memory management and scheduling
  - Synchronization
  - Examples : Windows, Linux, etc

Unix

- Dennis Ritchie and Ken Thomson tried to find an alternative for Multics

- Appeared at the right time

Slater, 1987

> The advent of the smaller computers, the minis—especially the PDP-11—had spawned a whole new group of computer users who were disappointed with existing operating software. They were ready for Unix. Most of Unix was not new, but rather what Ritchie calls "a good engineering application of ideas that had been around in some form and [were now] made convenient to use."

Unix adopted

- Spread and soon became widely adopted

Aho, 1984

> In the commercial world there are 100,000 Unix systems in operation ... Virtually every major university throughout the world now uses the Unix system.
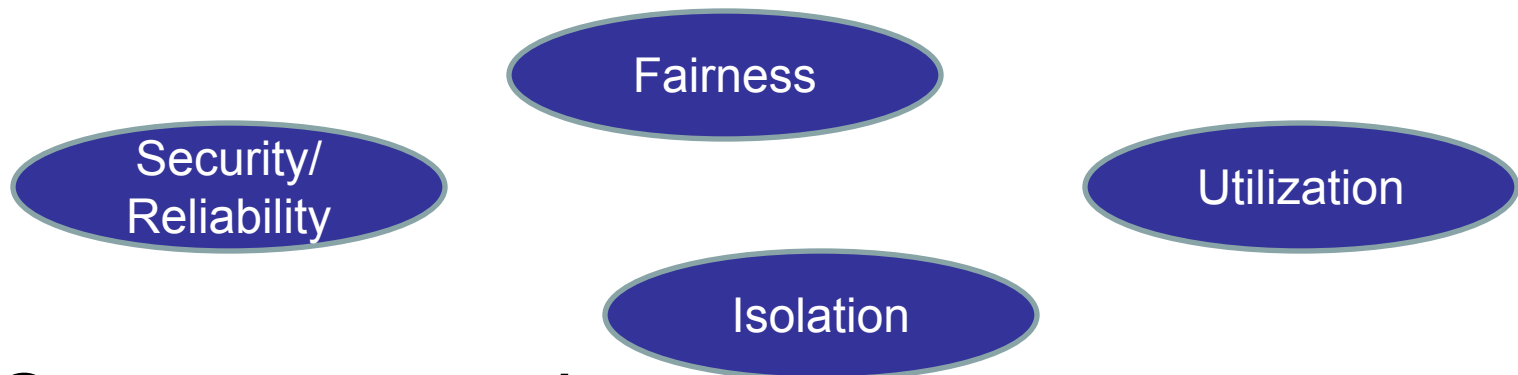
- Wide spread adoption arguably a hindrance to research?
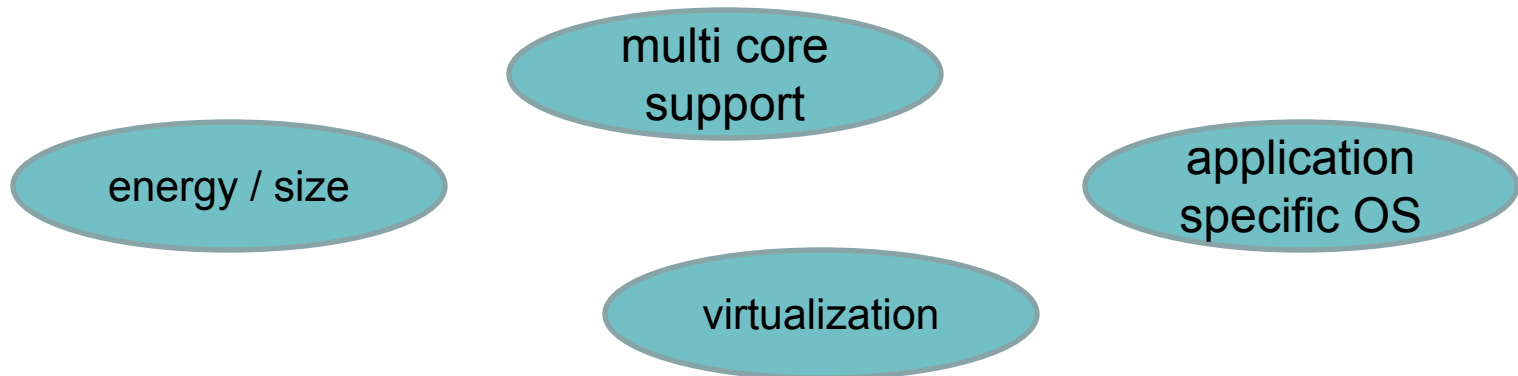
Smartphones & Tablets

- Hardware
  - VLSI ICs, low power requirements & high compute power
- Operating Systems
  - User friendly
  - Power awareness
  - Always connected
  - Offload to cloud
  - Better protection, Virtual machines
  - Examples : Android, iOS

OS Buzzwords

- Buzzwords that have been around

Fairness

Security/Reliability

Utilization

Isolation

- Contemporary buzzwords

multi core support

energy / size

application specific OS

virtualization

OS Research Trends

Small
(footprint,
Minimum energy
requirements)

Security, Reliability
(fewer errors,
Formally verified,
fault tolerant,
isolation)

Features
(better device support,
Multi core support)