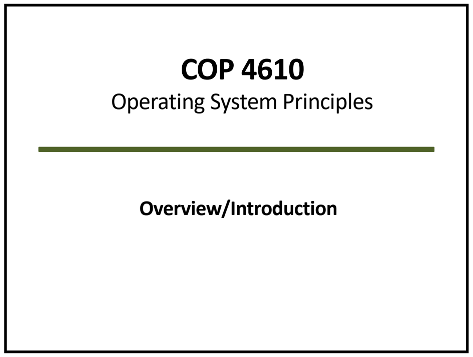
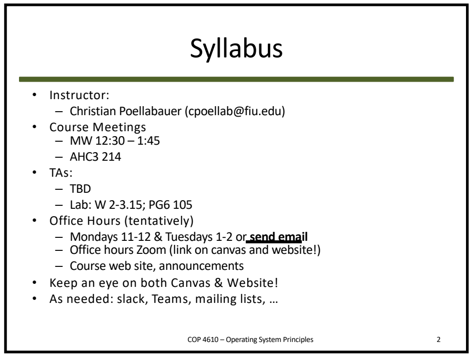


COP 4610
Operating System Principles

Overview/Introduction

1

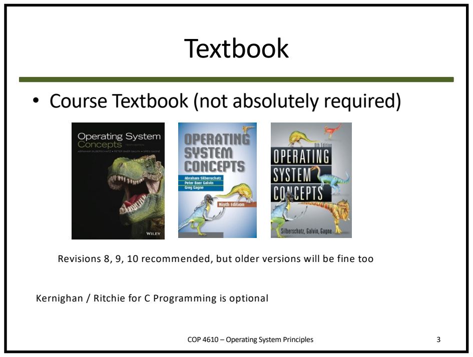


Syllabus

- Instructor:
  - Christian Poellabauer (cpoellab@fiu.edu)
- Course Meetings
  - MW 12:30 – 1:45
  - AHC3 214
- TAs:
  - TBD
  - Lab: W 2-3.15; PG6 105
- Office Hours (tentatively)
  - Mondays 11-12 & Tuesdays 1-2 or send email
  - Office hours Zoom (link on canvas and website!)
  - Course web site, announcements
- Keep an eye on both Canvas & Website!
- As needed: slack, Teams, mailing lists, …

COP 4610 – Operating System Principles 2

2



Textbook

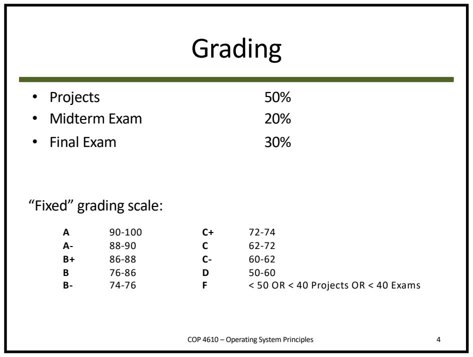
- Course Textbook (not absolutely required)

Revisions 8, 9, 10 recommended, but older versions will be fine too

Kernighan / Ritchie for C Programming is optional

COP 4610 – Operating System Principles 3

3



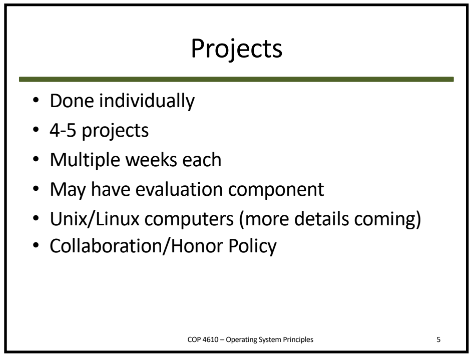
Grading

• Projects	50%
• Midterm Exam	20%
• Final Exam	30%

“Fixed” grading scale:

A	90-100	**C+**	72-74
A-	88-90	**C**	62-72
B+	86-88	**C-**	60-62
B	76-86	**D**	50-60
B-	74-76	**F**	< 50 OR < 40 Projects OR < 40 Exams

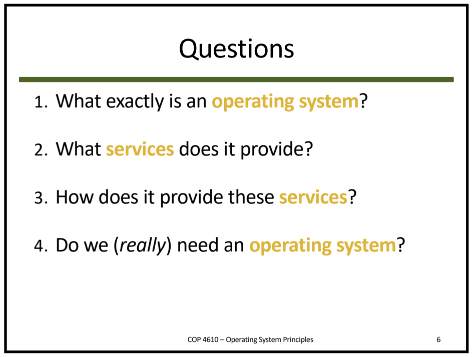
4



Projects

- Done individually
- 4-5 projects
- Multiple weeks each
- May have evaluation component
- Unix/Linux computers (more details coming)
- Collaboration/Honor Policy

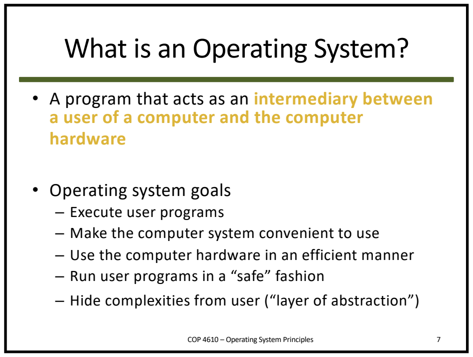
5



Questions

1. What exactly is an operating system?
2. What services does it provide?
3. How does it provide these services?
4. Do we (*really*) need an operating system?

6

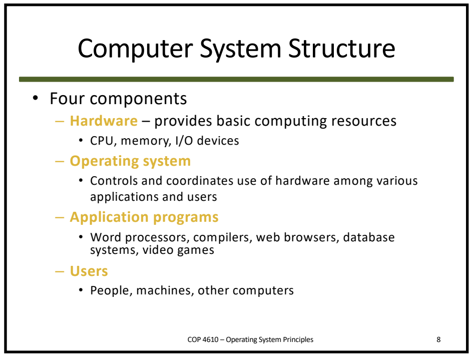


What is an Operating System?

- A program that acts as an **intermediary between a user of a computer and the computer hardware**

- Operating system goals
  - Execute user programs
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner
  - Run user programs in a “safe” fashion
  - Hide complexities from user (“layer of abstraction”)

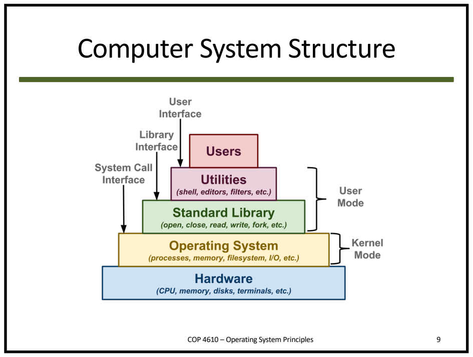
7



Computer System Structure

- Four components
  - **Hardware** – provides basic computing resources
    - CPU, memory, I/O devices
  - **Operating system**
    - Controls and coordinates use of hardware among various applications and users
  - **Application programs**
    - Word processors, compilers, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers

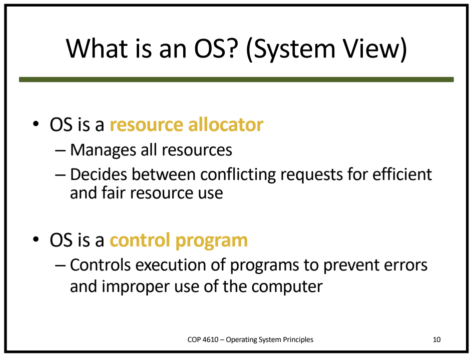
8



Computer System Structure

User Interface
Library Interface
System Call Interface
Users
Utilities
(shell, editors, filters, etc.)
Standard Library
(open, close, read, write, fork, etc.)
Operating System
(processes, memory, filesystem, I/O, etc.)
Hardware
(CPU, memory, disks, terminals, etc.)
User Mode
Kernel Mode

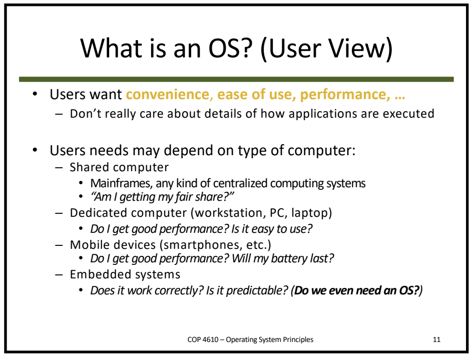
9



What is an OS? (System View)

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

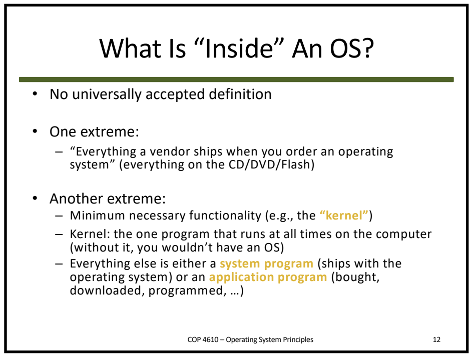
10



What is an OS? (User View)

- Users want **convenience, ease of use, performance, …**
  - Don't really care about details of how applications are executed
- Users needs may depend on type of computer:
  - Shared computer
    - Mainframes, any kind of centralized computing systems
    - *"Am I getting my fair share?"*
  - Dedicated computer (workstation, PC, laptop)
    - *Do I get good performance? Is it easy to use?*
  - Mobile devices (smartphones, etc.)
    - *Do I get good performance? Will my battery last?*
  - Embedded systems
    - *Does it work correctly? Is it predictable? (**Do we even need an OS?**)*

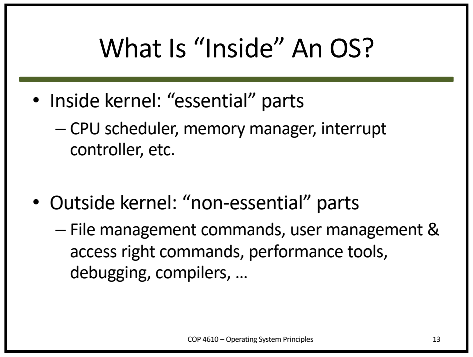
11



What Is "Inside" An OS?

- No universally accepted definition
- One extreme:
  - "Everything a vendor ships when you order an operating system" (everything on the CD/DVD/Flash)
- Another extreme:
  - Minimum necessary functionality (e.g., the **"kernel"**)
  - Kernel: the one program that runs at all times on the computer (without it, you wouldn't have an OS)
  - Everything else is either a **system program** (ships with the operating system) or an **application program** (bought, downloaded, programmed, …)

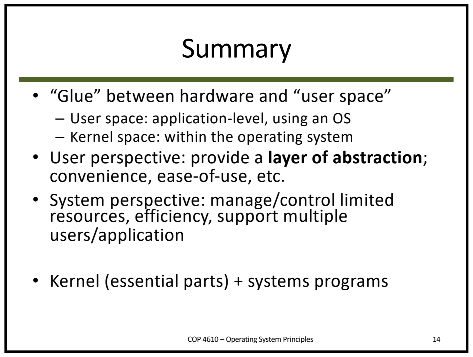
12



What Is "Inside" An OS?

- Inside kernel: "essential" parts
  - CPU scheduler, memory manager, interrupt controller, etc.
- Outside kernel: "non-essential" parts
  - File management commands, user management & access right commands, performance tools, debugging, compilers, …

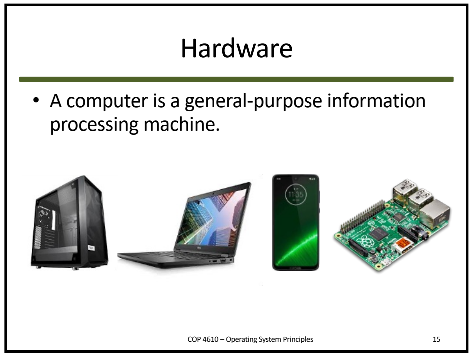
13



Summary

- "Glue" between hardware and "user space"
  - User space: application-level, using an OS
  - Kernel space: within the operating system
- User perspective: provide a **layer of abstraction**; convenience, ease-of-use, etc.
- System perspective: manage/control limited resources, efficiency, support multiple users/application
- Kernel (essential parts) + systems programs

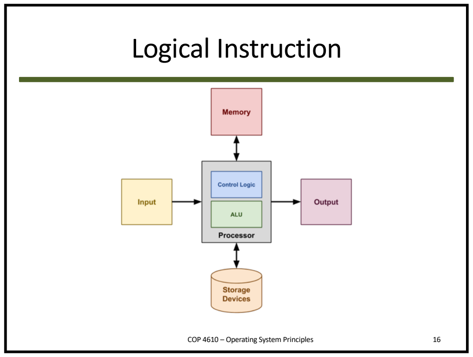
14



Hardware

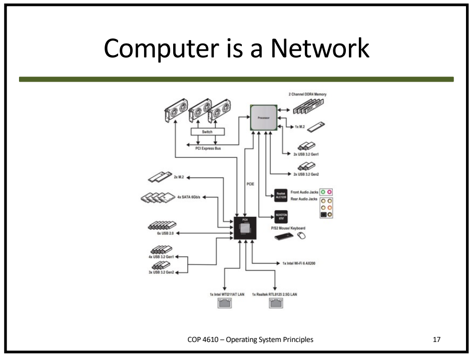
- A computer is a general-purpose information processing machine.

15



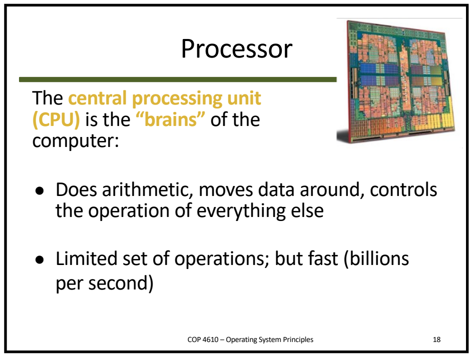
Logical Instruction

16



Computer is a Network

17

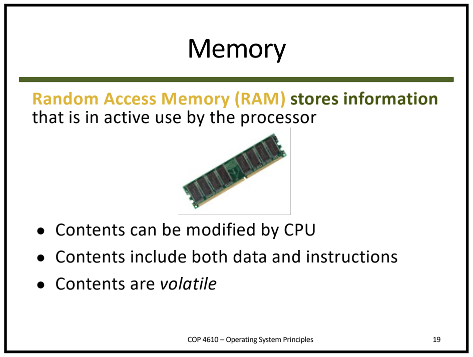


Processor

The **central processing unit (CPU)** is the **"brains"** of the computer:

- Does arithmetic, moves data around, controls the operation of everything else
- Limited set of operations; but fast (billions per second)

18



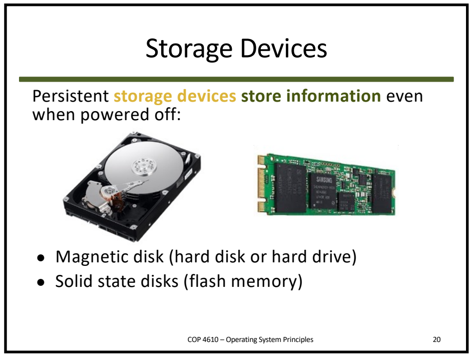
Memory

Random Access Memory (RAM) stores information that is in active use by the processor

- Contents can be modified by CPU
- Contents include both data and instructions
- Contents are *volatile*

COP 4610 – Operating System Principles 19

19



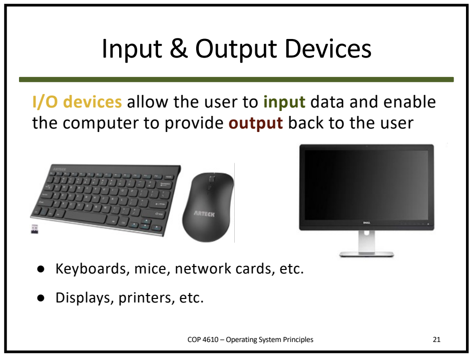
Storage Devices

Persistent **storage devices store information** even when powered off:

- Magnetic disk (hard disk or hard drive)
- Solid state disks (flash memory)

COP 4610 – Operating System Principles 20

20



Input & Output Devices

I/O devices allow the user to **input** data and enable the computer to provide **output** back to the user

- Keyboards, mice, network cards, etc.
- Displays, printers, etc.

COP 4610 – Operating System Principles 21

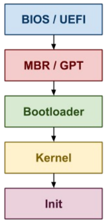
21

Computer Startup

- **Bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution (Unix: 1st process = swapper/sched, 2nd process = init)

22

Boot Sequence: BIOS



1. BIOS / UEFI

BIOS: Basic Input/Output System
UEFI: Unified Extensible Firmware Interface

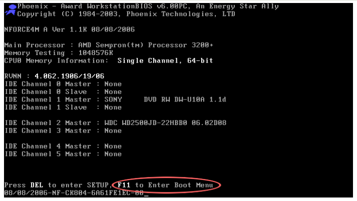
a. Usually stored in ROM

b. Performs some basic system integrity checks

c. Searches for device to boot or primary bootloader

23

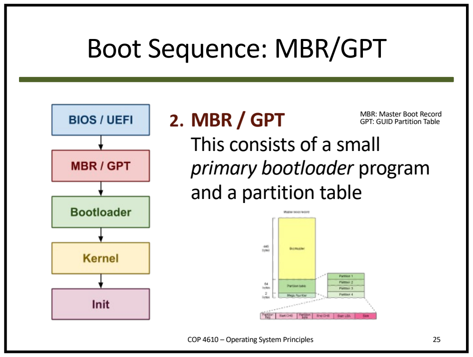
BIOS



```
Phoenix - Award WorkstationBIOS v6.00PC, An Energy Star Ally
Copyright (C) 1984-2003, Phoenix Technologies, LTD

NFORCE4M A Ver 1.1K 08/08/2006

Main Processor : AMD Sempron(tm) Processor 3200+
Memory Testing : 1048576K
CPU0 Memory Information:  Single Channel, 64-bit

RVNN : 4.062.1906/19/06
IDE Channel 0 Master : None
IDE Channel 0 Slave  : None
IDE Channel 1 Master : SONY     DVD RW DW-U10A 1.1d
IDE Channel 1 Slave  : None

IDE Channel 2 Master : WDC WD2500JD-22HBB0 06.02D08
IDE Channel 3 Master : None

IDE Channel 4 Master : None
IDE Channel 5 Master : None



Press DEL to enter SETUP, F11 to Enter Boot Menu
08/08/2006-NF-CK804-6A61FE1EC-00_
```

Basic Input / Output System

24



Boot Sequence: MBR/GPT

BIOS / UEFI → MBR / GPT → Bootloader → Kernel → Init

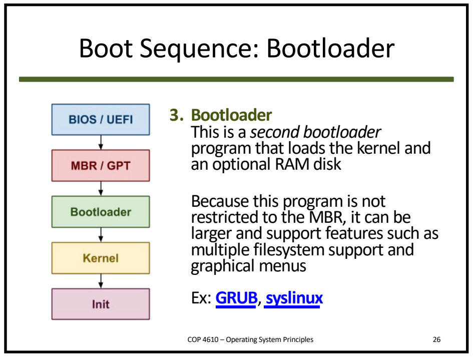
2. MBR / GPT

MBR: Master Boot Record
GPT: GUID Partition Table

This consists of a small *primary bootloader* program and a partition table

Master boot record
446 bytes
Bootloader
64 bytes
Partition table
2 bytes
Magic Number
Partition 1
Partition 2
Partition 3
Partition 4
Partition flag
Start CHS
Partition type
End CHS
Start LBA
Size

25



Boot Sequence: Bootloader

BIOS / UEFI → MBR / GPT → Bootloader → Kernel → Init

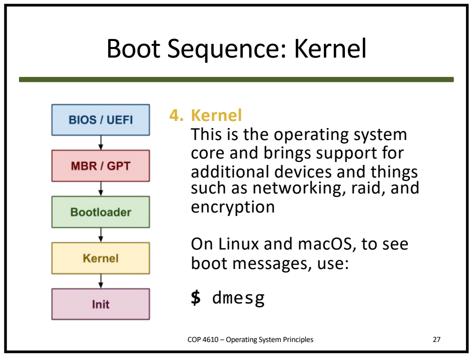
3. Bootloader

This is a *second bootloader* program that loads the kernel and an optional RAM disk

Because this program is not restricted to the MBR, it can be larger and support features such as multiple filesystem support and graphical menus

Ex: **GRUB**, **syslinux**

26



Boot Sequence: Kernel

BIOS / UEFI → MBR / GPT → Bootloader → Kernel → Init

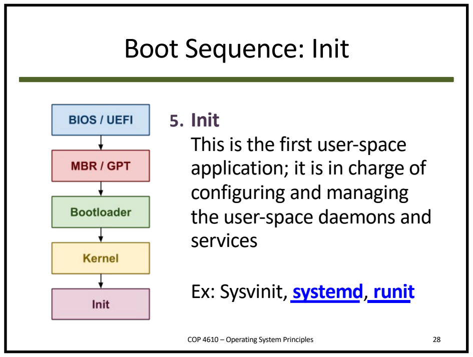
4. Kernel

This is the operating system core and brings support for additional devices and things such as networking, raid, and encryption

On Linux and macOS, to see boot messages, use:

```
$ dmesg
```

27



Boot Sequence: Init

BIOS / UEFI

MBR / GPT

Bootloader

Kernel

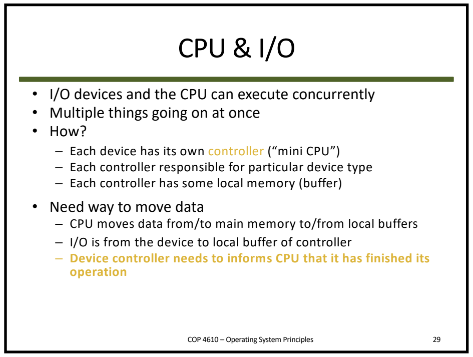
Init

5. Init

This is the first user-space application; it is in charge of configuring and managing the user-space daemons and services

Ex: Sysvinit, systemd, runit

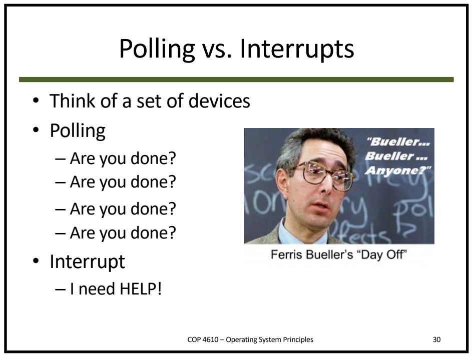
28



CPU & I/O

- I/O devices and the CPU can execute concurrently
- Multiple things going on at once
- How?
  - Each device has its own controller ("mini CPU")
  - Each controller responsible for particular device type
  - Each controller has some local memory (buffer)
- Need way to move data
  - CPU moves data from/to main memory to/from local buffers
  - I/O is from the device to local buffer of controller
  - **Device controller needs to informs CPU that it has finished its operation**

29

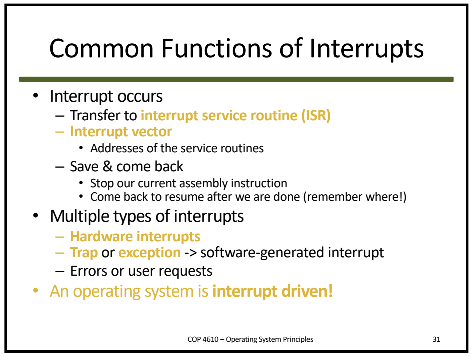


Polling vs. Interrupts

- Think of a set of devices
- Polling
  - Are you done?
  - Are you done?
  - Are you done?
  - Are you done?
- Interrupt
  - I need HELP!

Ferris Bueller's "Day Off"

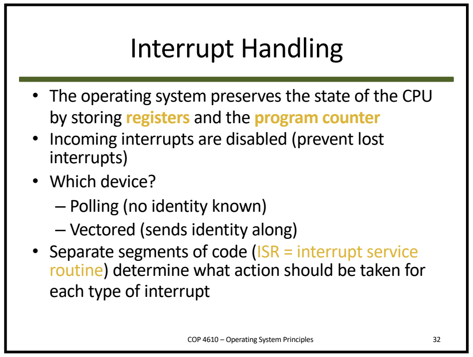
30



Common Functions of Interrupts

- Interrupt occurs
  - Transfer to **interrupt service routine (ISR)**
  - **Interrupt vector**
    - Addresses of the service routines
  - Save & come back
    - Stop our current assembly instruction
    - Come back to resume after we are done (remember where!)
- Multiple types of interrupts
  - **Hardware interrupts**
  - **Trap** or **exception** -> software-generated interrupt
  - Errors or user requests
- An operating system is **interrupt driven!**

31

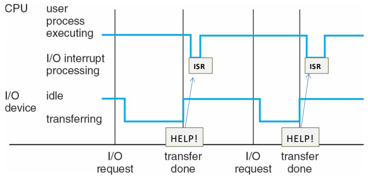


Interrupt Handling

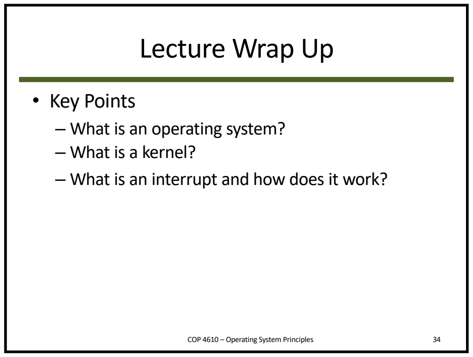
- The operating system preserves the state of the CPU by storing **registers** and the **program counter**
- Incoming interrupts are disabled (prevent lost interrupts)
- Which device?
  - Polling (no identity known)
  - Vectored (sends identity along)
- Separate segments of code (ISR = interrupt service routine) determine what action should be taken for each type of interrupt

32

Interrupt Timeline



33



Lecture Wrap Up

- Key Points
  - What is an operating system?
  - What is a kernel?
  - What is an interrupt and how does it work?

COP 4610 – Operating System Principles 34

34