

CPE 460 Operating System Design

Lecture 1: Course Overview

Ahmed Tamrawi

February 8, 2017

Important Equations for the Class



Erwin Schrödinger (1887-1961)

Schrödinger Equation:

$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \Psi(x, t) + V(x) \Psi(x, t)$$

Schrödinger equation does for a quantum-mechanical particle what Newton's Second Law does for a classical particle. The Solution to Schrödinger equation to determine how a particle evolves in time, just as we use Newton's Second Law to solve for future position and momentum of a classical particle.

Further Reading:

1. https://simple.wikipedia.org/wiki/Schr%C3%B6dinger_equation
2. <https://www.quora.com/What-is-the-Schr%C3%B6dinger-wave-equation-and-what-are-its-applications>



Werner Heisenberg (1901-1976)

Heisenberg Uncertainty Principal:

$$\Delta x \Delta p \geq \frac{h}{4\pi} = \frac{\hbar}{2}$$

In the world of very small particles, one cannot measure any property of a particle without interacting with it in some way. This introduces an unavoidable uncertainty into the result. Thus, One can never measure all the properties exactly.

The more accurately you know the position (the smaller Δx is), the less accurately you know the momentum (the larger Δp is); and vice versa

Further Reading:

https://en.wikipedia.org/wiki/Uncertainty_principle





B.Eng. Computer Engineering
(Class of 2007)

IOWA STATE
UNIVERSITY

M.Sc. Computer Engineering
(Class of 2011)

IOWA STATE
UNIVERSITY

Ph.D.. Computer Engineering
(Class of 2016)



Secure Programming *Static Program Analysis*
Data & Pattern Mining

Software Analysis & Security

Bug finding and Malware detection *Build System Analysis*
Abstractions and Symbolic Evaluations

Quantum Physics
Biology
Astronomy



i'm **not** on
facebook®



WhatsApp





YOU

- *Name*
- *Year in undergraduate program.*
- *Something about you*
 - *Food you like.*
 - *Programming languages you used.*
 - *Open source projects you contributed to.*
- *What do you think of this course?*
- *What are your goals after graduation?*

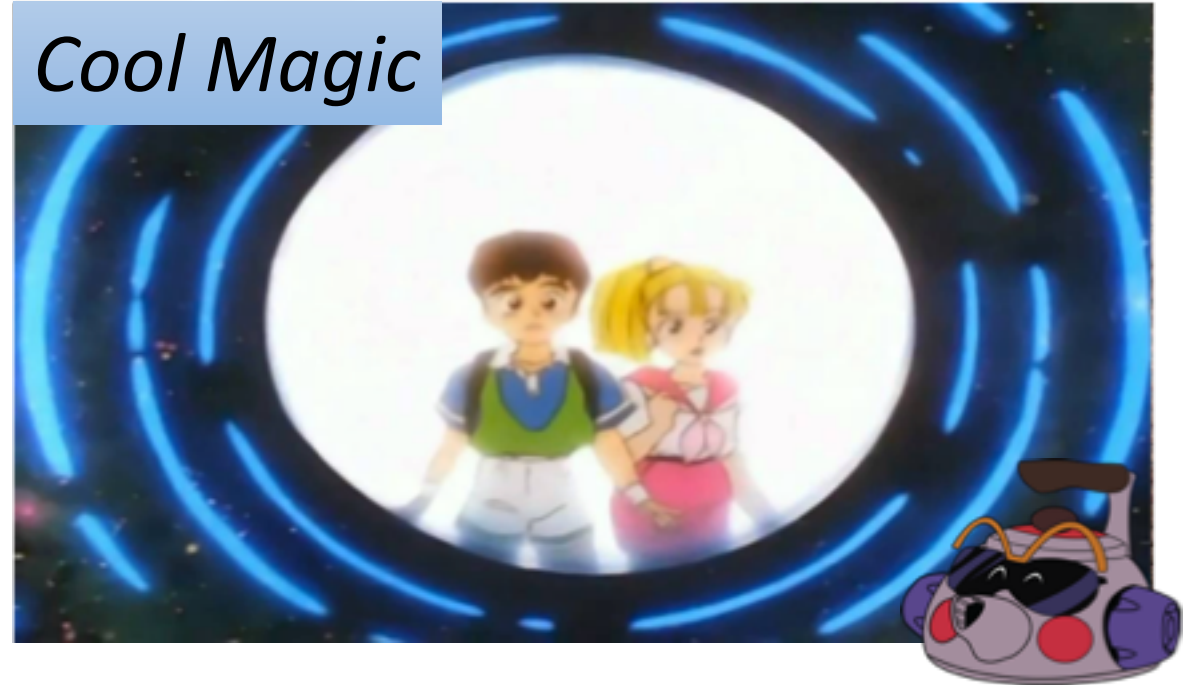
Syllabus

Goal of the Class



Cool Computing Stuff

Cool Magic



Improve our understanding of *how* computers work
to **minimize** the magic

My Goals for Lectures?

Convey some complex technical ideas

Teach you what you need to know to do the labs, assignments, and the project

Avoid being fired

Keep most of you awake for 90 minutes

Get you to laugh at dumb jokes

Lectures are *horrible* medium for learning complex ideas, many resource are available online.

The point of labs, homework, and project is to teach you things I want you to learn in the class

Avoid being fired

You probably should be getting more sleep

Gabriel Iglesias is more funnier (check him out)

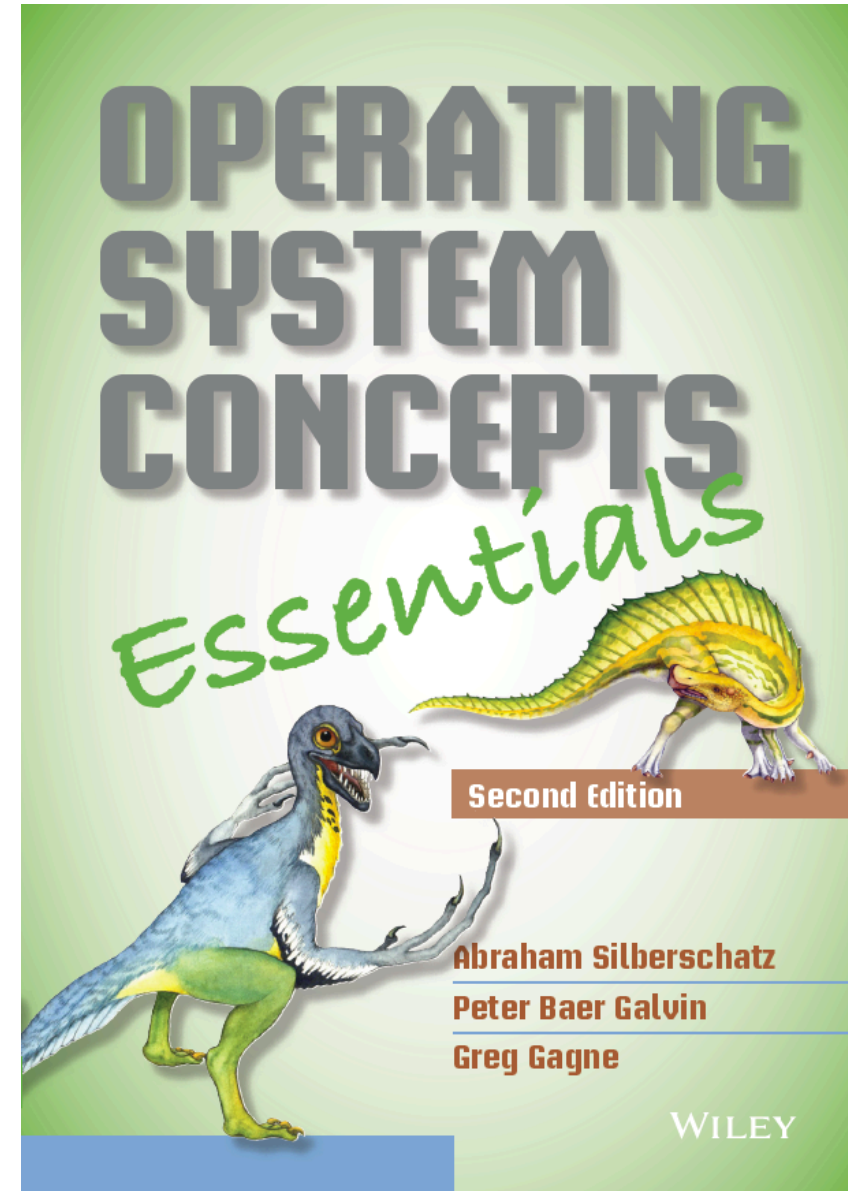


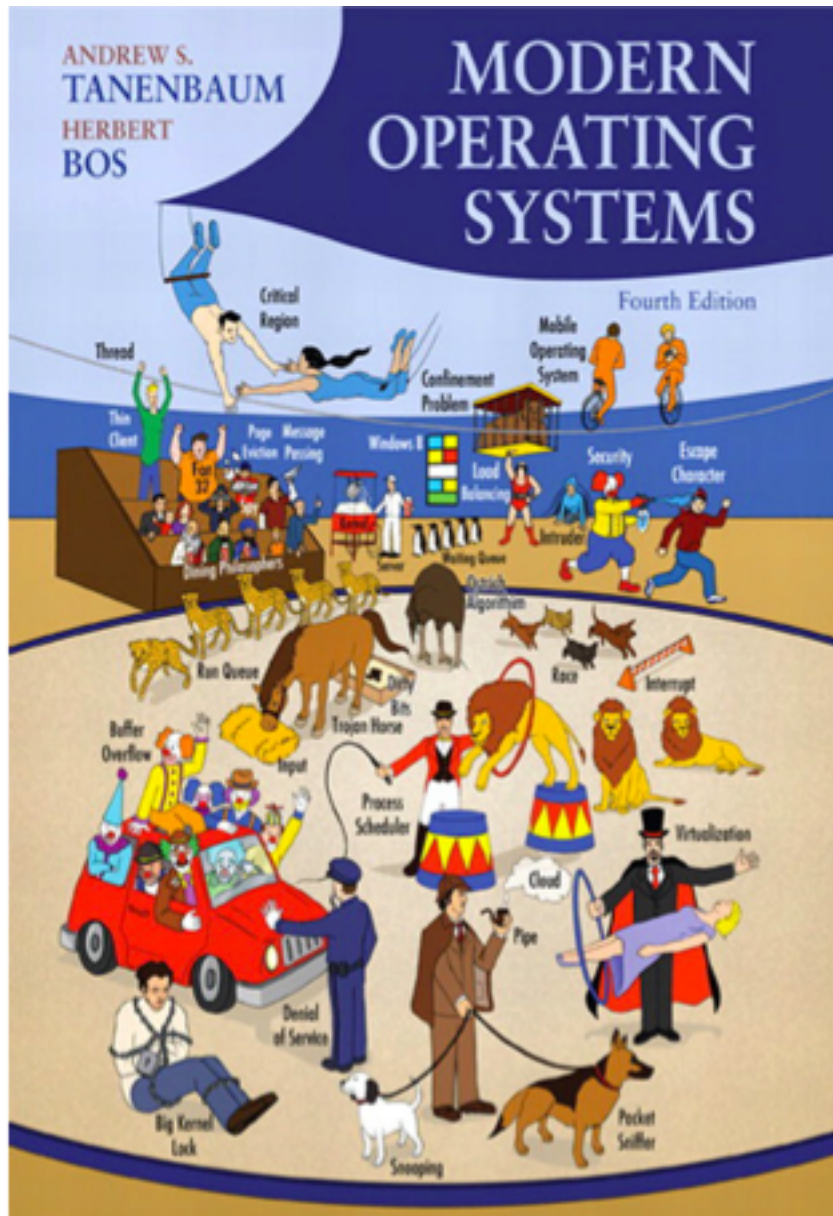
My Real Goal for Lectures

Provide **context** and **meaning** for the things you have or will later **learn on your own**

An **operating system** is a program that manages a computer's hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware. An amazing aspect of operating systems is how they vary in accomplishing these tasks. Mainframe operating systems are designed primarily to optimize utilization of hardware. Personal computer (PC) operating systems support complex games, business applications, and everything in between. Operating systems for mobile computers provide an environment in which a user can easily interface with the computer to execute programs. Thus, some operating systems are designed to be *convenient*, others to be *efficient*, and others to be some combination of the two.

A more common definition, and the one that we usually follow, is that the operating system is the one program running at all times on the computer—usually called the **kernel**. (Along with the kernel, there are two other types of programs: **system programs**, which are associated with the operating system but are not necessarily part of the kernel, and application programs, which include all programs not associated with the operation of the system.)





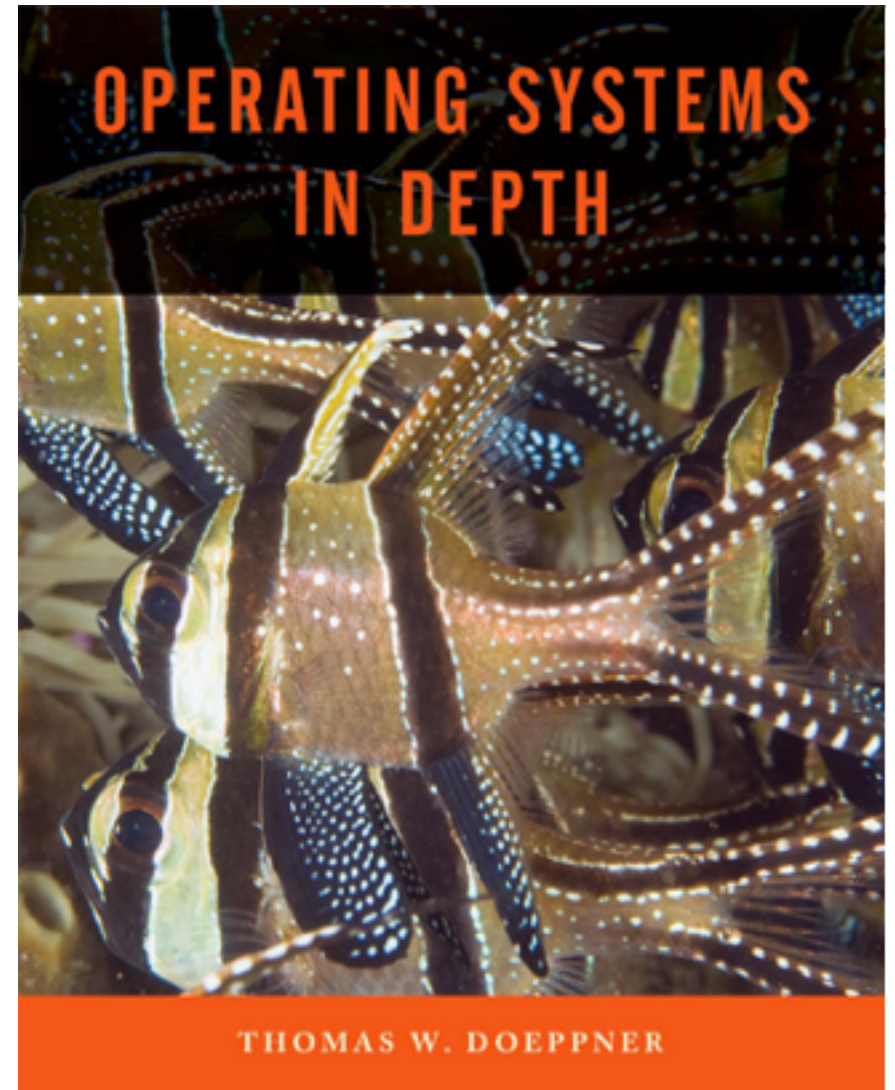
1.1 WHAT IS AN OPERATING SYSTEM?

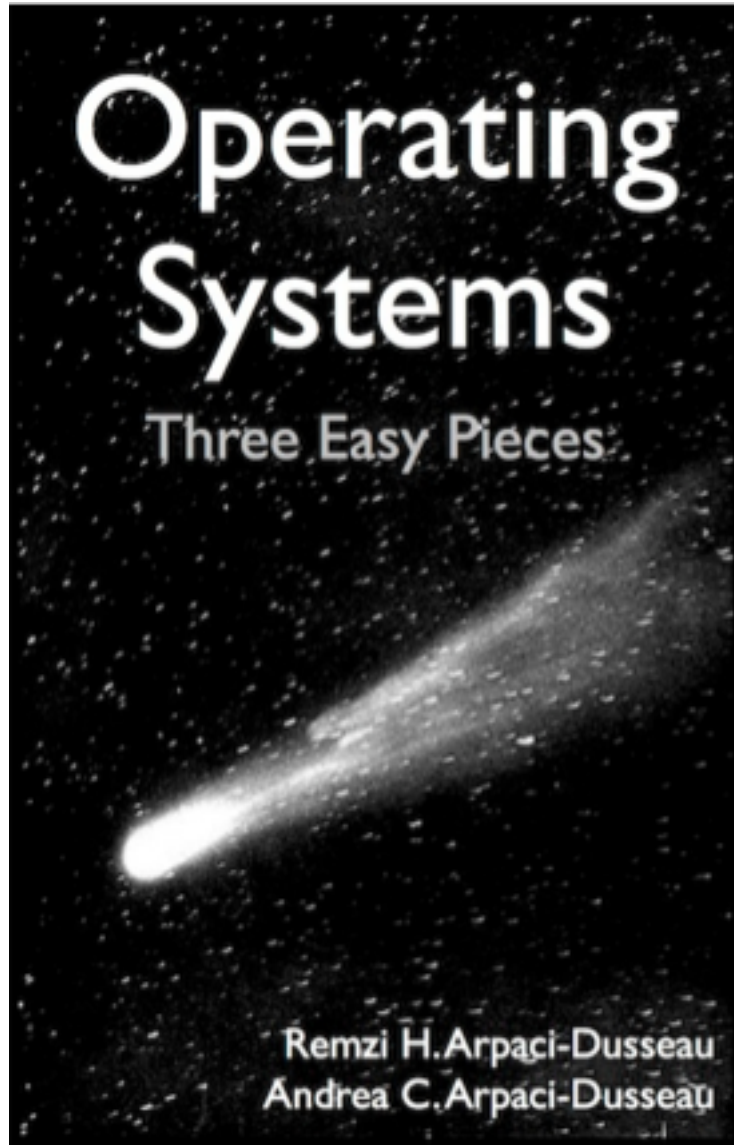
It is hard to pin down what an operating system is other than saying it is the software that runs in kernel mode—and even that is not always true. Part of the problem is that operating systems perform two essentially unrelated functions: providing application programmers (and application programs, naturally) a clean abstract set of resources instead of the messy hardware ones and managing these hardware resources. Depending on who is doing the talking, you might hear mostly about one function or the other. Let us now look at both.

1.1 OPERATING SYSTEMS

What's an operating system? You might say it's what's between you and the hardware, but that would cover pretty much all software. So let's say it's the software that sits between your software and the hardware. But does that mean that the library you picked up from some web site is part of the operating system? We probably want our operating-system definition to be a bit less inclusive. So, let's say that it's that software that almost everything else depends upon. This is still vague, but then the term is used in a rather nebulous manner throughout the industry.

Perhaps we can do better by describing what an operating system is actually supposed to do. From a programmer's point of view, operating systems provide useful abstractions of the underlying hardware facilities. Since many programs can use these facilities at once, the operating system is also responsible for managing how these facilities are shared.





There is a body of software, in fact, that is responsible for making it easy to run programs (even allowing you to seemingly run many at the same time), allowing programs to share memory, enabling programs to interact with devices, and other fun stuff like that. That body of software is called the **operating system (OS)**³, as it is in charge of making sure the system operates correctly and efficiently in an easy-to-use manner.

Do we like any of these definitions?

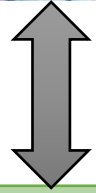
No universally accepted definition

I know that I like
Mansaf!





(Petra)



Petitions



House of Representative
(Parliament)



Resources & Commitments

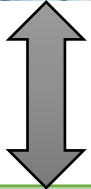


Arabic text from a news article or document, including a title and several lines of text.



Manage Resources

Provide Abstractions



Petitions



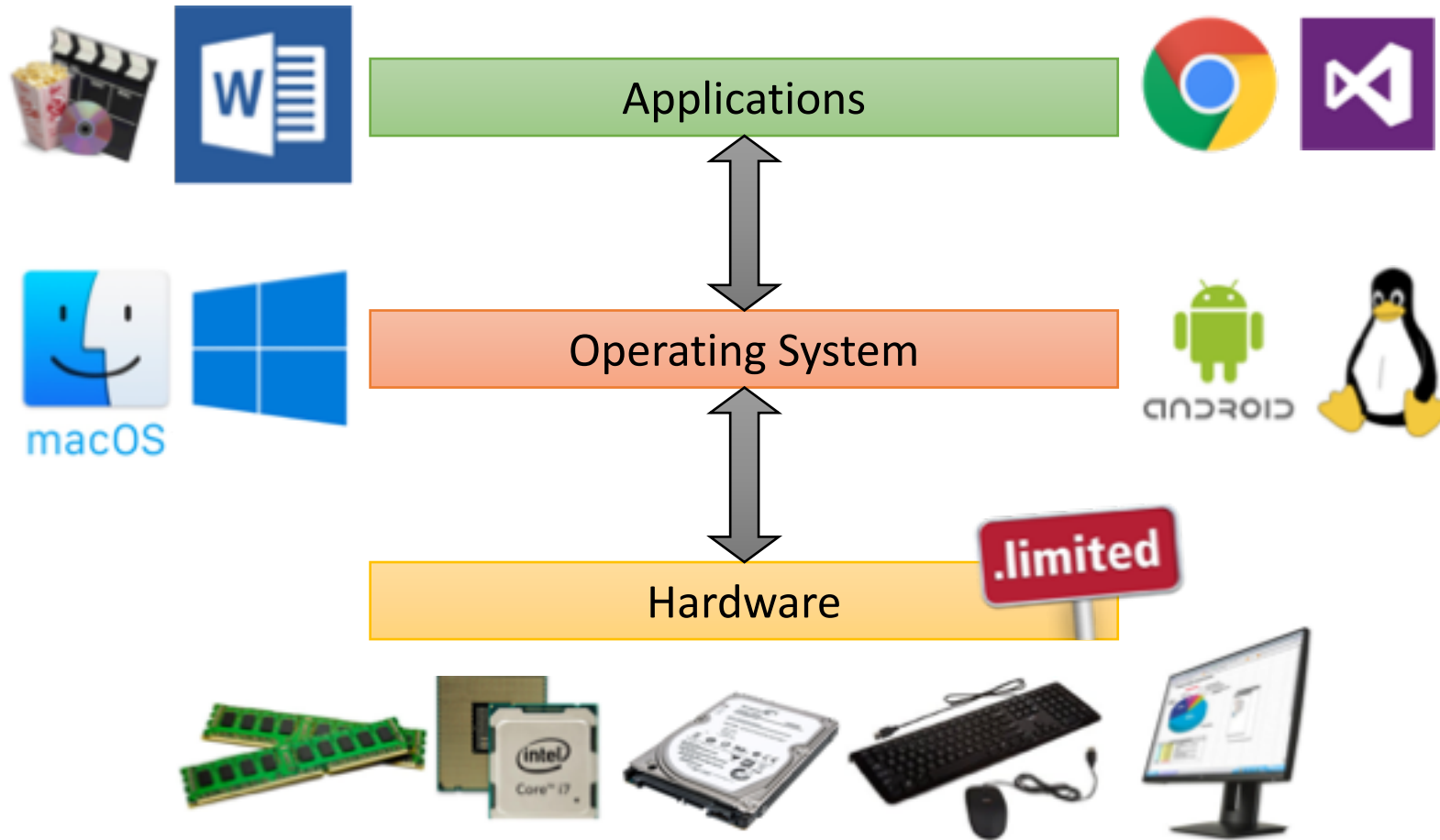
House of Representative
(Parliament)



Resources & Commitments



Realistic View of Operating System



CPE 460 OS Definition

An **operating system** is a program that
manages resources and *provide abstractions*

Main Ideas in CPE 460

Manage Resources

How do you *share* **processors, memory, and hardware devices** among programs?

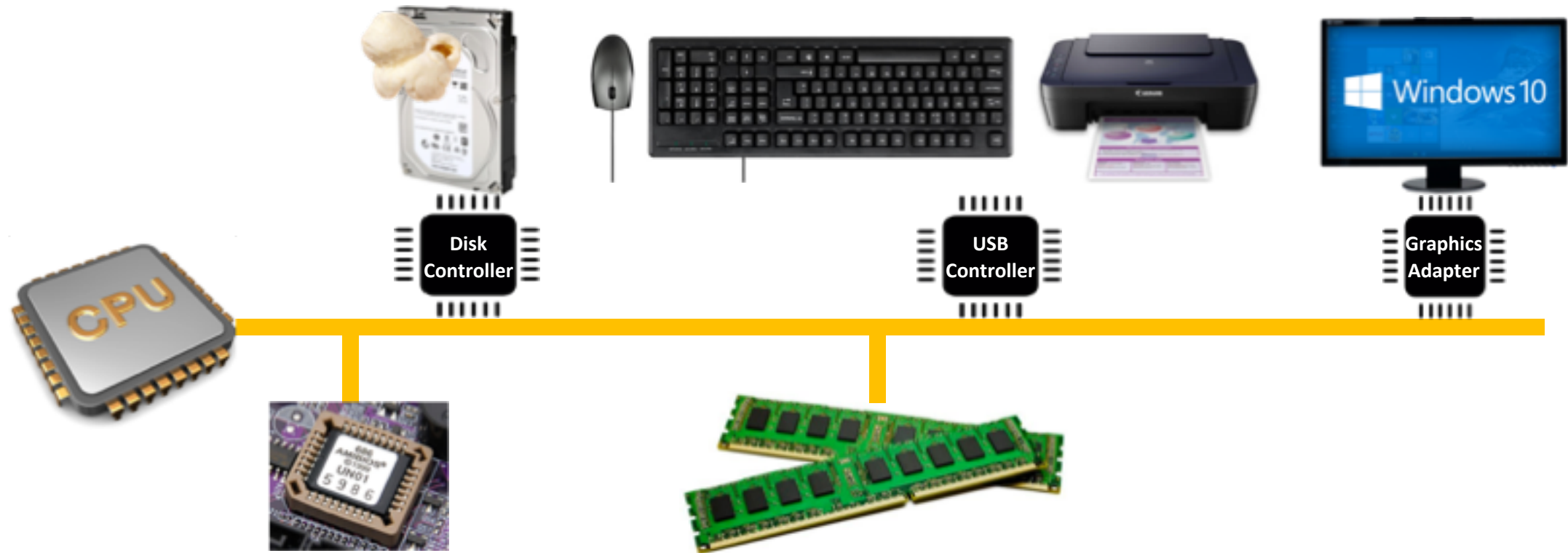
Provide Abstractions

How do you provide programs with **clean and easy to use** interfaces to resources, without sacrificing (too much) **efficiency and flexibility**?

Does it have an Operating System?



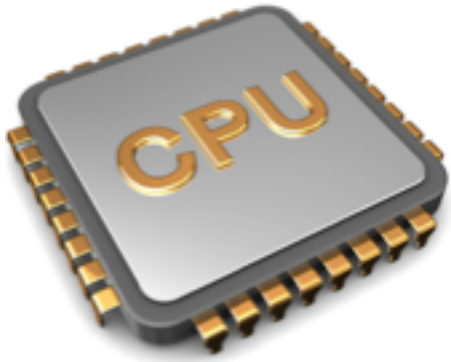
CPE 460 Computer System



What happens at Computer Startup?







Finds itself in Real Mode

Power On Self Test

Executes the code at address 0xFFFF0 which corresponds to BIOS

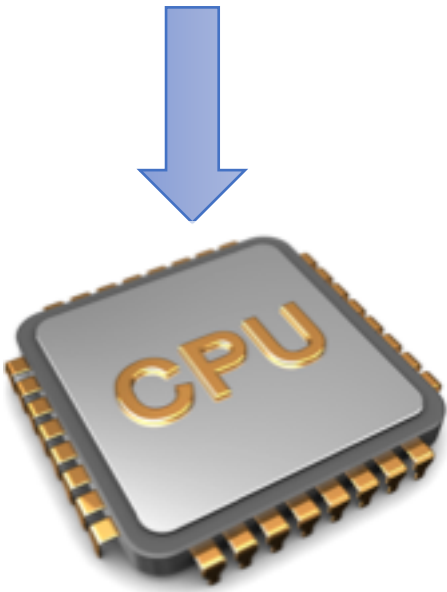


```
AWARD Modulation BIOS v6.0, An Energy Star Ally
Copyright (C) 1994-2001, Award Software, Inc.
ASUS P4T533-C ACPI BIOS Revision 1907 Beta 001
Intel(R) Pentium(R) 4 2000 MHz Processor
Memory Test : 262144K OK

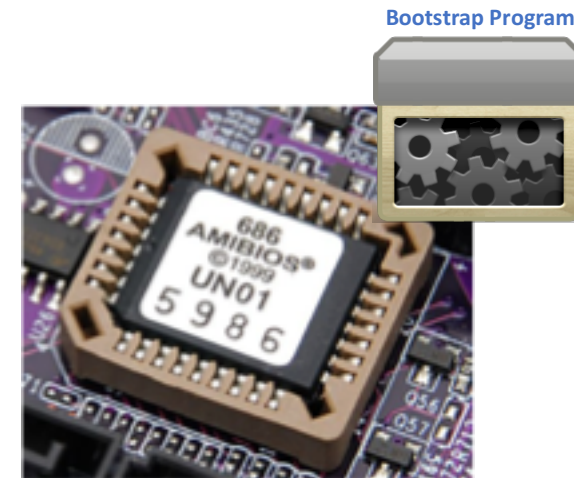
Award Plug and Play BIOS Extension v1.0b
Initialize Plug and Play Cards...
PMF Init Completed

Detecting Primary Master ... MAXTOR 6L448J2
Detecting Primary Slave ... ADIS CB-S52B/A
Detecting Secondary Master ... Skip
Detecting Secondary Slave ... None...

Press DEL to enter SETUP, ALT-F2 to enter EZ flash utility
06/28/2002-1058E/1G12/W6Z7-P4T533-C
```



- Finds itself in Real Mode
- Power On Self Test
- Executes the code at address 0xFFFF0 which corresponds to BIOS



- Autoprobing I/O ports
- Looks for bootloader in Boot Device
- It loads the first sector of a bootable device at 0x7C00 and jumps to it. Then it executes the MBR bootloader located in the first sector of a bootable disk (/dev/hda or /dev/sda)

CC BY **MASTER BOOT RECORD** > INVOKE-IR
 BY: JARED ATKINSON
 TEMPLATE BY: ANGE ALBERTIN

BOOT CODE

```

0001: 33 c9 86 00 8c 00 7c 8e c0 8e 08 00 7c 0f 00
0002: 00 49 00 02 fc f3 44 50 68 3c 0c c8 f8 09 04 00
0003: 80 8c 07 80 7e 00 00 7c 06 0f 81 0c 01 83 c3 10
0004: 42 f1 c0 18 88 36 00 15 c6 46 11 05 c6 46 10 00
0005: 84 41 88 aa 15 c9 13 50 72 0f 81 f8 15 aa 75 09
0006: f7 c1 05 00 74 01 f8 46 10 66 80 80 7e 30 00 74
0007: 26 66 68 00 00 00 00 00 66 ff 76 08 68 00 00 68 00
0008: 7c 68 00 68 10 60 84 c2 8a 16 00 88 f4 c0 13
0009: 9f 83 c4 10 9c 18 14 68 01 02 88 00 7c 8a 16 00
000a: 8a 76 0c 8a 4e 02 8a 6e 01 c0 13 66 62 7b 1c f8
000b: 4e 11 75 0c 80 7e 90 80 0f 84 8a 00 62 80 88 84
000c: 15 12 64 84 16 00 c0 13 50 68 9e 81 3e f8 70 15
000d: aa 71 68 ff 76 00 18 80 00 75 17 fa 80 81 66 64
000e: 68 81 90 80 0f 16 90 68 7c 00 90 ff 65 64 18 75
000f: 00 f8 68 00 88 c0 1a 68 23 c0 73 56 81 f8 14
0010: 43 50 42 75 32 81 f9 02 01 72 2c 66 68 0f 88 00
0011: 00 66 68 00 02 00 00 66 68 00 00 00 66 53 66
0012: 13 66 15 66 68 00 00 00 66 68 00 7c 00 00 c0
0013: 65 68 00 0f c0 1a 1a 12 f8 6a 00 7c 00 00 c0
0014: 18 a0 87 07 68 08 a0 86 07 68 93 a0 85 07 32 64
0015: 05 00 07 88 f0 ac 7c 00 74 09 88 07 00 84 c0 c0
0016: 30 10 42 f4 68 f0 28 c9 64 68 18 00 24 02 00 68
0017: 24 02 c3 a9 60 76 61 6c 69 64 20 70 62 72 74 69
0018: 74 69 6f 6e 20 74 62 6c 65 90 45 72 72 6f 72
0019: 20 6c 6f 61 64 69 6e 67 20 6f 70 65 72 61 74 69
001a: 66 67 20 71 79 71 74 65 80 60 60 69 71 69 64
001b: 67 20 6f 70 65 72 61 74 69 6e 67 20 71 79 71 74
001c: 65 60 90 00 00 61 78 6a 62 04 5a 70 00 00 70
001d: 21 00 07 f0 8f 01 00 08 00 00 00 36 00 8e f8
001e: ff ff 07 f6 ff ff 00 a3 30 06 00 00 00 00 70
001f: 00 00 00 80 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 80 00 00 00 00 00 00 00 00 00 00 00
  
```

CHS ADDRESSING

head - 1st byte
 sector - 2nd byte (0-1 bits)
 cylinder - 2nd byte (0-7 bits)
 3rd byte

PARTITION TABLE

status	starting head	starting sector	starting cylinder	partition type	ending head	ending sector	ending cylinder	relative start sector	total sectors
0x00 - Non-Bootable	0x20	0x21	0x00	0x07 - NTFS	0xFE	0x3F	0x3FF	0x800	0x6369000
0x80 - Bootable	0xFE	0x3F	0x3FF	0x07 - NTFS	0xFE	0x3F	0x3FF	0x636A000	0x966000
0x00 - EMPTY				0x00 - EMPTY					
0x00 - EMPTY				0x00 - EMPTY					

END OF MBR marker 0x55AA

PARTITION TYPES

0x00 - EMPTY	0x01 - LINUX
0x02 - FAT12	0x03 - VERBATIM
0x04 - FAT16	0x05 - LINUX_EXTENDED
0x06 - NTFS	0x07 - NTFS_VOLUME_001
0x08 - NTFS	0x09 - NTFS_VOLUME_002
0x0A - FAT32	0x0B - INFORMATION_1
0x0C - FAT32	0x0D - INFORMATION_2
0x0E - FAT32	0x0F - FAT32
0x10 - FAT32	0x11 - VERBATIM
0x12 - MS_DYNAMIC	0x13 - FAT32
0x14 - HIDDEN_FAT12	0x15 - HIDDEN_FAT16
0x16 - HIDDEN_FAT16	0x17 - HIDDEN_FAT32
0x18 - HIDDEN_FAT32	0x19 - HIDDEN_FAT32
0x1A - HIDDEN_FAT32	0x1B - HIDDEN_FAT32
0x1C - HIDDEN_FAT32	0x1D - HIDDEN_FAT32
0x1E - HIDDEN_FAT32	0x1F - HIDDEN_FAT32
0x20 - HIDDEN_FAT32	0x21 - HIDDEN_FAT32
0x22 - HIDDEN_FAT32	0x23 - HIDDEN_FAT32
0x24 - HIDDEN_FAT32	0x25 - HIDDEN_FAT32
0x26 - HIDDEN_FAT32	0x27 - HIDDEN_FAT32
0x28 - HIDDEN_FAT32	0x29 - HIDDEN_FAT32
0x2A - HIDDEN_FAT32	0x2B - HIDDEN_FAT32
0x2C - HIDDEN_FAT32	0x2D - HIDDEN_FAT32
0x2E - HIDDEN_FAT32	0x2F - HIDDEN_FAT32
0x30 - HIDDEN_FAT32	0x31 - HIDDEN_FAT32
0x32 - HIDDEN_FAT32	0x33 - HIDDEN_FAT32
0x34 - HIDDEN_FAT32	0x35 - HIDDEN_FAT32
0x36 - HIDDEN_FAT32	0x37 - HIDDEN_FAT32
0x38 - HIDDEN_FAT32	0x39 - HIDDEN_FAT32
0x3A - HIDDEN_FAT32	0x3B - HIDDEN_FAT32
0x3C - HIDDEN_FAT32	0x3D - HIDDEN_FAT32
0x3E - HIDDEN_FAT32	0x3F - HIDDEN_FAT32
0x40 - HIDDEN_FAT32	0x41 - HIDDEN_FAT32
0x42 - HIDDEN_FAT32	0x43 - HIDDEN_FAT32
0x44 - HIDDEN_FAT32	0x45 - HIDDEN_FAT32
0x46 - HIDDEN_FAT32	0x47 - HIDDEN_FAT32
0x48 - HIDDEN_FAT32	0x49 - HIDDEN_FAT32
0x4A - HIDDEN_FAT32	0x4B - HIDDEN_FAT32
0x4C - HIDDEN_FAT32	0x4D - HIDDEN_FAT32
0x4E - HIDDEN_FAT32	0x4F - HIDDEN_FAT32
0x50 - HIDDEN_FAT32	0x51 - HIDDEN_FAT32
0x52 - HIDDEN_FAT32	0x53 - HIDDEN_FAT32
0x54 - HIDDEN_FAT32	0x55 - HIDDEN_FAT32
0x56 - HIDDEN_FAT32	0x57 - HIDDEN_FAT32
0x58 - HIDDEN_FAT32	0x59 - HIDDEN_FAT32
0x5A - HIDDEN_FAT32	0x5B - HIDDEN_FAT32
0x5C - HIDDEN_FAT32	0x5D - HIDDEN_FAT32
0x5E - HIDDEN_FAT32	0x5F - HIDDEN_FAT32
0x60 - HIDDEN_FAT32	0x61 - HIDDEN_FAT32
0x62 - HIDDEN_FAT32	0x63 - HIDDEN_FAT32
0x64 - HIDDEN_FAT32	0x65 - HIDDEN_FAT32
0x66 - HIDDEN_FAT32	0x67 - HIDDEN_FAT32
0x68 - HIDDEN_FAT32	0x69 - HIDDEN_FAT32
0x6A - HIDDEN_FAT32	0x6B - HIDDEN_FAT32
0x6C - HIDDEN_FAT32	0x6D - HIDDEN_FAT32
0x6E - HIDDEN_FAT32	0x6F - HIDDEN_FAT32
0x70 - HIDDEN_FAT32	0x71 - HIDDEN_FAT32
0x72 - HIDDEN_FAT32	0x73 - HIDDEN_FAT32
0x74 - HIDDEN_FAT32	0x75 - HIDDEN_FAT32
0x76 - HIDDEN_FAT32	0x77 - HIDDEN_FAT32
0x78 - HIDDEN_FAT32	0x79 - HIDDEN_FAT32
0x7A - HIDDEN_FAT32	0x7B - HIDDEN_FAT32
0x7C - HIDDEN_FAT32	0x7D - HIDDEN_FAT32
0x7E - HIDDEN_FAT32	0x7F - HIDDEN_FAT32
0x80 - HIDDEN_FAT32	0x81 - HIDDEN_FAT32
0x82 - HIDDEN_FAT32	0x83 - HIDDEN_FAT32
0x84 - HIDDEN_FAT32	0x85 - HIDDEN_FAT32
0x86 - HIDDEN_FAT32	0x87 - HIDDEN_FAT32
0x88 - HIDDEN_FAT32	0x89 - HIDDEN_FAT32
0x8A - HIDDEN_FAT32	0x8B - HIDDEN_FAT32
0x8C - HIDDEN_FAT32	0x8D - HIDDEN_FAT32
0x8E - HIDDEN_FAT32	0x8F - HIDDEN_FAT32
0x90 - HIDDEN_FAT32	0x91 - HIDDEN_FAT32
0x92 - HIDDEN_FAT32	0x93 - HIDDEN_FAT32
0x94 - HIDDEN_FAT32	0x95 - HIDDEN_FAT32
0x96 - HIDDEN_FAT32	0x97 - HIDDEN_FAT32
0x98 - HIDDEN_FAT32	0x99 - HIDDEN_FAT32
0x9A - HIDDEN_FAT32	0x9B - HIDDEN_FAT32
0x9C - HIDDEN_FAT32	0x9D - HIDDEN_FAT32
0x9E - HIDDEN_FAT32	0x9F - HIDDEN_FAT32
0xA0 - HIDDEN_FAT32	0xA1 - HIDDEN_FAT32
0xA2 - HIDDEN_FAT32	0xA3 - HIDDEN_FAT32
0xA4 - HIDDEN_FAT32	0xA5 - HIDDEN_FAT32
0xA6 - HIDDEN_FAT32	0xA7 - HIDDEN_FAT32
0xA8 - HIDDEN_FAT32	0xA9 - HIDDEN_FAT32
0xAA - HIDDEN_FAT32	0xAB - HIDDEN_FAT32
0xAC - HIDDEN_FAT32	0xAD - HIDDEN_FAT32
0xAE - HIDDEN_FAT32	0xAF - HIDDEN_FAT32
0xB0 - HIDDEN_FAT32	0xB1 - HIDDEN_FAT32
0xB2 - HIDDEN_FAT32	0xB3 - HIDDEN_FAT32
0xB4 - HIDDEN_FAT32	0xB5 - HIDDEN_FAT32
0xB6 - HIDDEN_FAT32	0xB7 - HIDDEN_FAT32
0xB8 - HIDDEN_FAT32	0xB9 - HIDDEN_FAT32
0xBA - HIDDEN_FAT32	0xBB - HIDDEN_FAT32
0xBC - HIDDEN_FAT32	0xBD - HIDDEN_FAT32
0xBE - HIDDEN_FAT32	0xBF - HIDDEN_FAT32
0xC0 - HIDDEN_FAT32	0xC1 - HIDDEN_FAT32
0xC2 - HIDDEN_FAT32	0xC3 - HIDDEN_FAT32
0xC4 - HIDDEN_FAT32	0xC5 - HIDDEN_FAT32
0xC6 - HIDDEN_FAT32	0xC7 - HIDDEN_FAT32
0xC8 - HIDDEN_FAT32	0xC9 - HIDDEN_FAT32
0xCA - HIDDEN_FAT32	0xCB - HIDDEN_FAT32
0xCC - HIDDEN_FAT32	0xCD - HIDDEN_FAT32
0xCE - HIDDEN_FAT32	0xCF - HIDDEN_FAT32
0xD0 - HIDDEN_FAT32	0xD1 - HIDDEN_FAT32
0xD2 - HIDDEN_FAT32	0xD3 - HIDDEN_FAT32
0xD4 - HIDDEN_FAT32	0xD5 - HIDDEN_FAT32
0xD6 - HIDDEN_FAT32	0xD7 - HIDDEN_FAT32
0xD8 - HIDDEN_FAT32	0xD9 - HIDDEN_FAT32
0xDA - HIDDEN_FAT32	0xDB - HIDDEN_FAT32
0xDC - HIDDEN_FAT32	0xDD - HIDDEN_FAT32
0xDE - HIDDEN_FAT32	0xDE - HIDDEN_FAT32
0xE0 - HIDDEN_FAT32	0xE1 - HIDDEN_FAT32
0xE2 - HIDDEN_FAT32	0xE3 - HIDDEN_FAT32
0xE4 - HIDDEN_FAT32	0xE5 - HIDDEN_FAT32
0xE6 - HIDDEN_FAT32	0xE7 - HIDDEN_FAT32
0xE8 - HIDDEN_FAT32	0xE9 - HIDDEN_FAT32
0xEA - HIDDEN_FAT32	0xEB - HIDDEN_FAT32
0xEC - HIDDEN_FAT32	0xED - HIDDEN_FAT32
0xEE - HIDDEN_FAT32	0xEF - HIDDEN_FAT32
0xF0 - HIDDEN_FAT32	0xF1 - HIDDEN_FAT32
0xF2 - HIDDEN_FAT32	0xF3 - HIDDEN_FAT32
0xF4 - HIDDEN_FAT32	0xF5 - HIDDEN_FAT32
0xF6 - HIDDEN_FAT32	0xF7 - HIDDEN_FAT32
0xF8 - HIDDEN_FAT32	0xF9 - HIDDEN_FAT32
0xFA - HIDDEN_FAT32	0xFB - HIDDEN_FAT32
0xFC - HIDDEN_FAT32	0xFD - HIDDEN_FAT32
0xFE - HIDDEN_FAT32	0xFF - HIDDEN_FAT32

CC BY **NTFS VOLUME BOOT RECORD** > INVOKE-IR
 BY: JARED ATKINSON
 TEMPLATE BY: ANGE ALBERTIN

FILE HEADER

```

0000: 68 52 90 46 54 46 53 20 20 20 02 08 00 00
0001: 3f 0f ff 00 00 08 00 00
0002: ff ff 79 07 00 00 00 00
0003: 00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00
0004: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0005: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0009: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000a: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000b: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000c: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000d: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000e: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000f: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0011: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0013: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0014: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0016: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0017: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0018: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0019: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001a: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001b: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001c: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001d: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001e: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001f: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

BIOS PARTITION BLOCK

fields	values
jump instruction	jmp 0x00000054
OEM ID	NTFS
bytes per sector	0x200
sectors per cluster	0x08
reserved sectors	0x00
media descriptor	0xFB
sectors per track	0x3F
number of heads	0xFF
hidden sectors	0x800
total sectors	0x6368FFF
MFT first cluster #	0xC0000
MFT max first cluster #	0x02
clusters per MFT record	0xF6
clusters per index block	0x01
volume serial #	E3133CD4233CD4CA
checksum	0x00000000

BOOTSTRAP CODE

```

170: 41 fa 01 68 03 00 0f 88 61 c3 a2 f6 01 68 09 00
171: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
172: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
173: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
174: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
175: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
176: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
177: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
178: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
179: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
181: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
182: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
183: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
184: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
185: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
186: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
187: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
188: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
189: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
191: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
192: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
193: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
194: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
195: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
196: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
197: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
198: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
199: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

END OF SECTOR marker 0xAAS5

Error Message A disk read error occurred BOOTMGR is compressed Press Ctrl+Alt+Del to restart

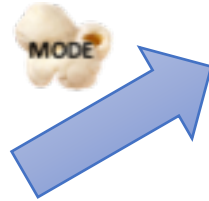
<http://www.invoke-ir.com/2015/05/ontheforensictrail-part2.html>

Any program to run **must** be loaded in memory





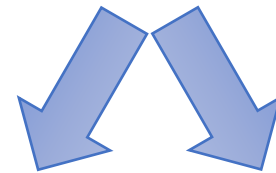
The kernel is decompressed from its image and its loaded into memory



Autoprobing I/O ports



init process

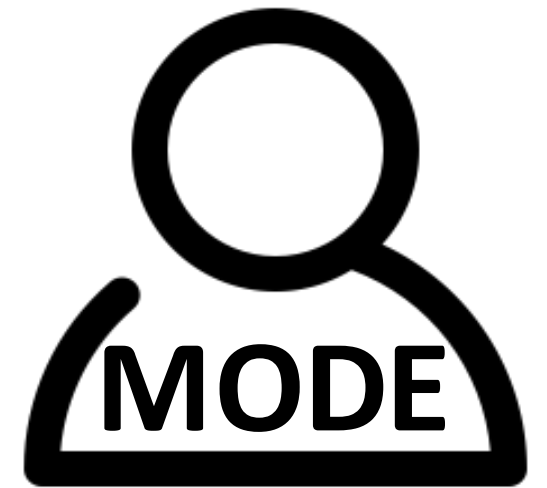


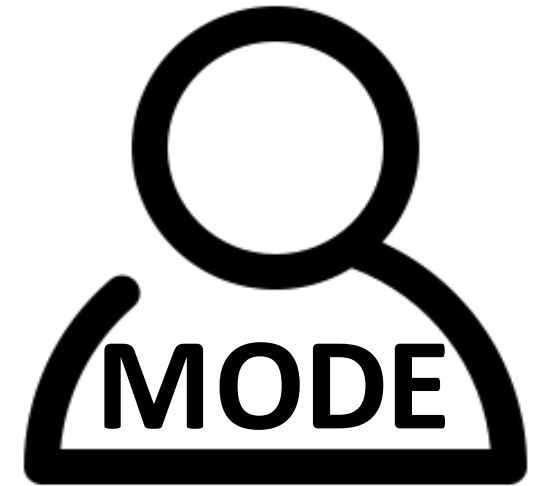
System Processes



System Daemons

The kernel is decompressed from its image and its loaded into memory





Wait for Event to Occur



انا مشن فاهم حاجة خالص



What happens when you move the cursor?



Mouse sends out pulses,
one pulse for every 1000th
of an inch or so

What happens when you move the cursor?



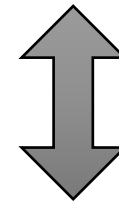
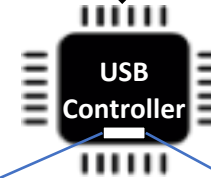
Mouse sends out pulses,
one pulse for every 1000th
of an inch or so

The pulses are received
through a USB packet or
through an old serial line

What happens when you move the cursor?

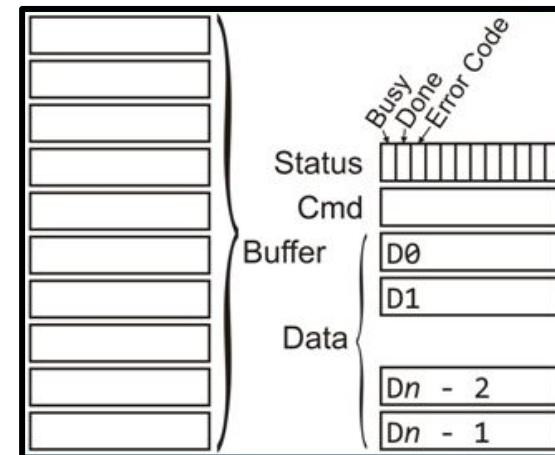


Hardware Interrupt

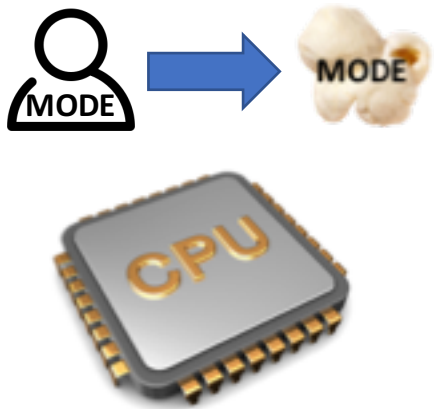


Mouse sends out pulses, one pulse for every 1000th of an inch or so

The pulses are received through a USB packet or through an old serial line



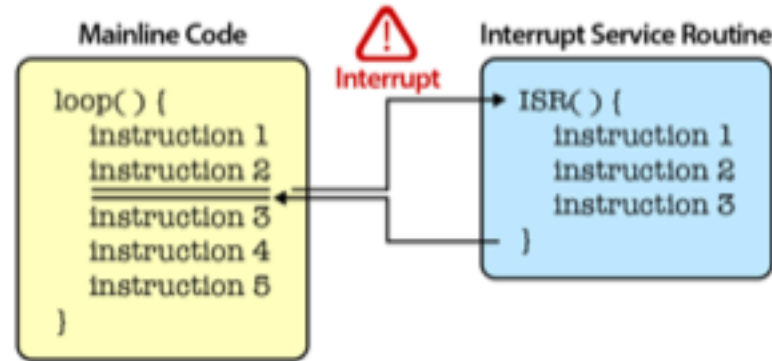
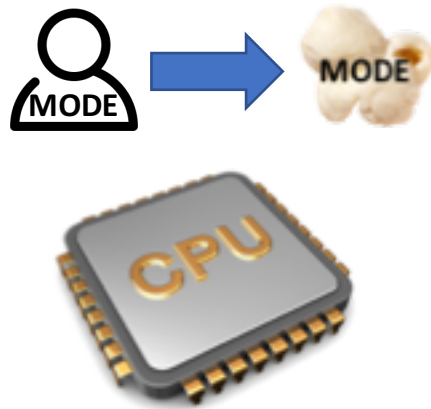
What happens when CPU is interrupted?



CPU preserves the current state of the CPU by storing registers and the program counter

Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**

What happens when CPU is interrupted?



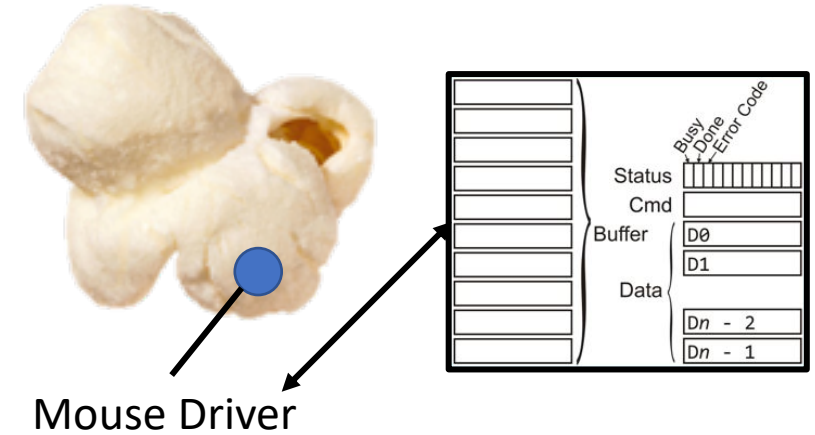
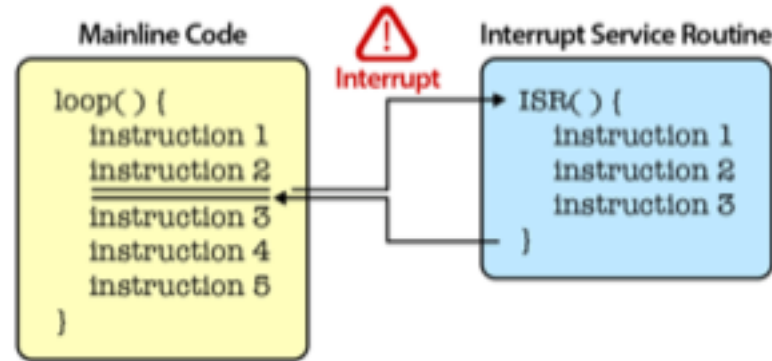
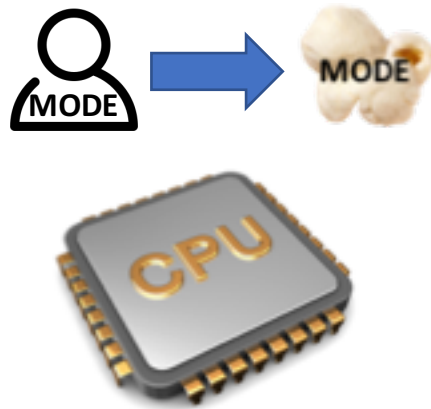
CPU preserves the current state of the CPU by storing registers and the program counter

Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**

Separate segments of code determine what action should be taken for each type of interrupt

Reads the interrupt and realizes it's from the mouse, and calls the proper ISR which calls the mouse driver.

What happens when CPU is interrupted?



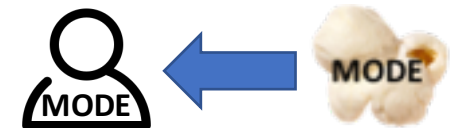
CPU preserves the current state of the CPU by storing registers and the program counter

Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**

Separate segments of code determine what action should be taken for each type of interrupt

Reads the interrupt and realizes it's from the mouse, and calls the proper ISR which calls the mouse driver.

Mouse driver adds the x and y increments to its current cursor position and return the result to OS





How to notify Monitor of cursor movement?

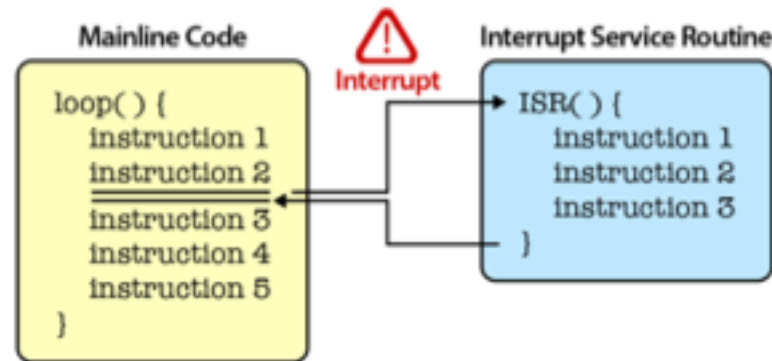


Software **Interrupt (Trap)**

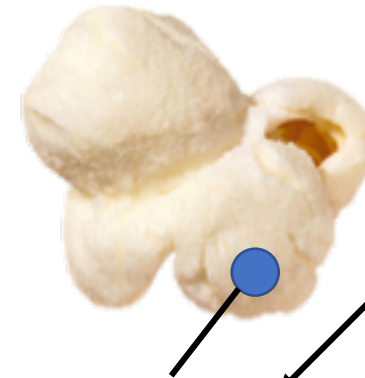
OS gets interrupted through a **system call** to update the screen



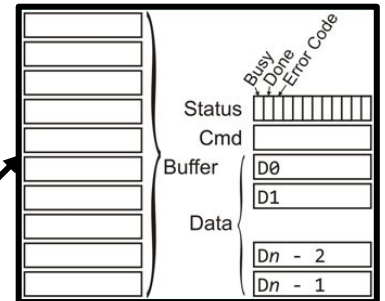
OS preserves the current state of the CPU by storing registers and the program counter



Reads the interrupt and realizes it's from OS to monitor. It calls the display driver with the updated screen



Display Driver



Monitor device drivers sets the proper registers and buffer data in the graphics adapter







Software Interrupt (Trap)



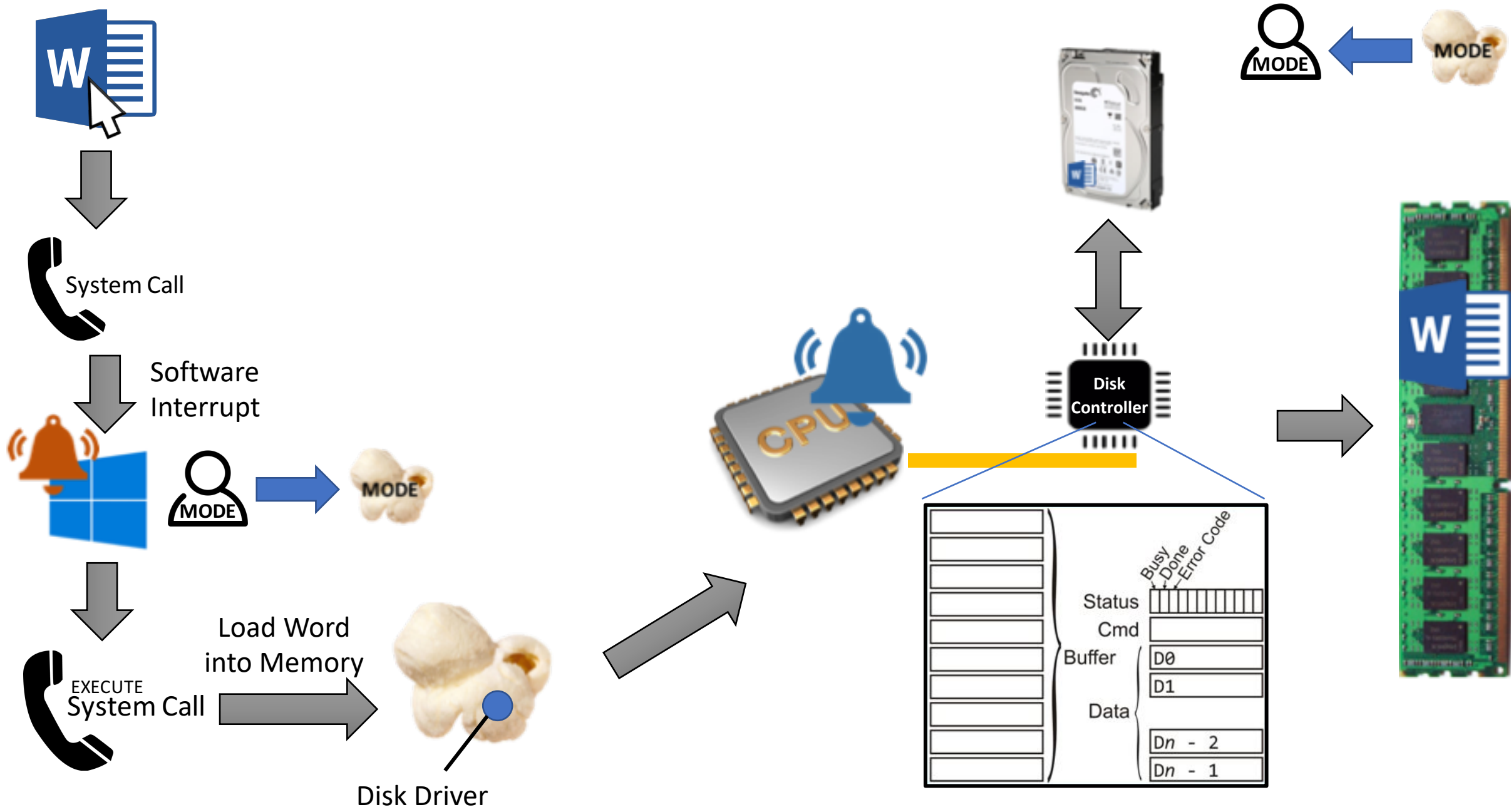
EXECUTE
System Call

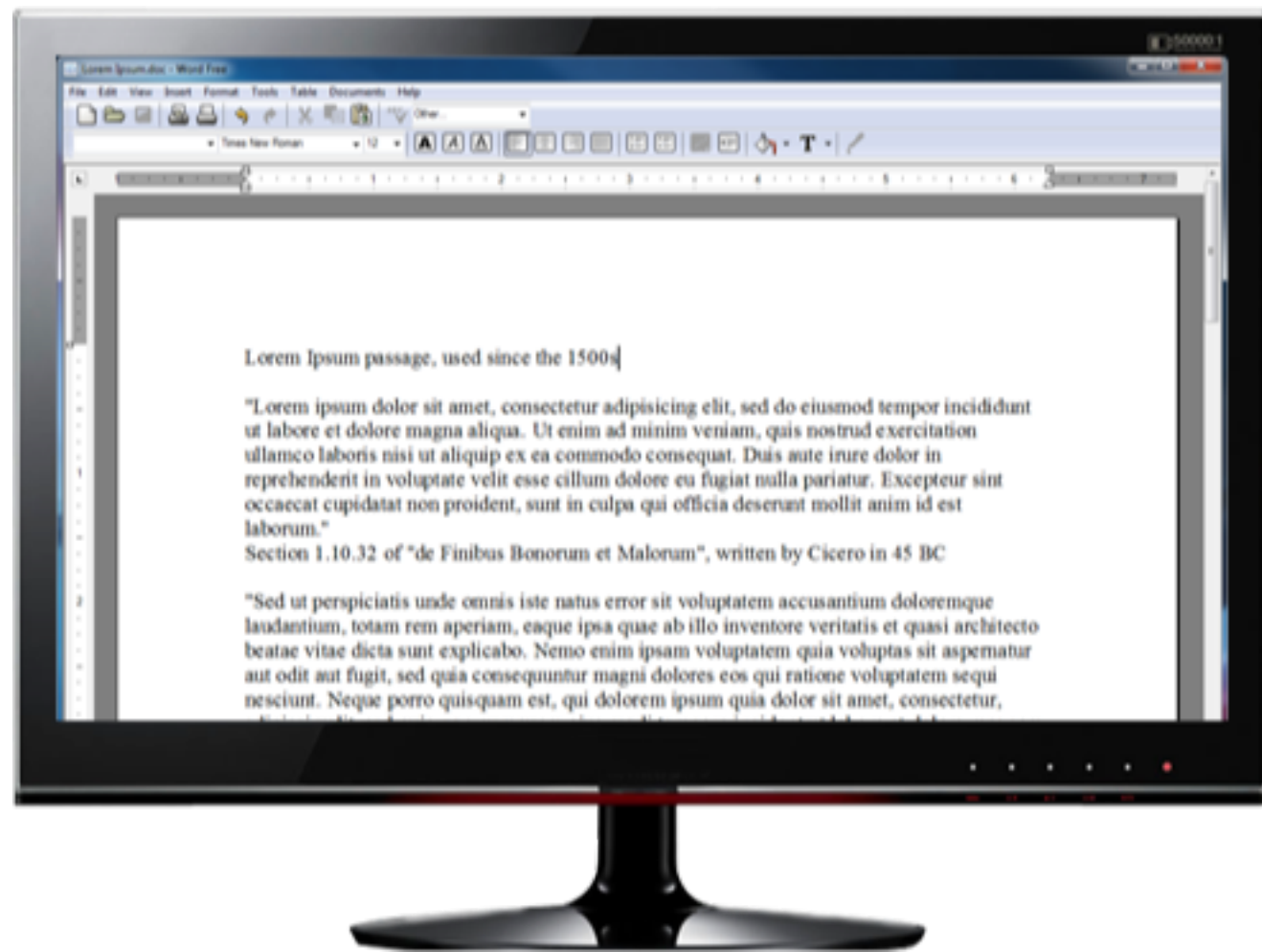




Any program to run **must** be loaded in memory







An operating system is **interrupt driven**







As long as their processes fit in memory, we do not have a memory problem



Each process needs resources to accomplish its task: CPU, memory, I/O, files, etc.



Process termination requires reclaim of any reusable resources

Typically system has many processes running concurrently,
how this is achieved?



Many Processes

Creating/deleting user and system processes

Suspending/resuming processes

Process Synchronization & Communication

Process Management
(Chapters 3,4 & 5)





The screenshot shows a window titled 'Tasks (10)' with tabs for 'Permissions', 'Http Headers', 'Properties', 'Preview', 'Versions', and 'EventLog'. The main area contains a table of tasks with columns for 'Task', 'Size', '%', 'Progress', 'Status', and 'Speed'. The tasks are all 'Downloading' files from 'sample-domain.com' to 'C:\Users\moises\...'.

Task	Size	%	Progress	Status	Speed
Downloading sample-domain.com/DSC04233.JPG to C:\Users\moises\...	3820155	27.32	<div style="width: 27.32%;"></div>	Running	76.97 KB/s
Downloading sample-domain.com/DSC04231.JPG to C:\Users\moises\...	4482289	59.73	<div style="width: 59.73%;"></div>	Running	198.60 KB/s
Downloading sample-domain.com/DSC04230.JPG to C:\Users\moises\...	4371329	75.15	<div style="width: 75.15%;"></div>	Running	288.12 KB/s
Downloading sample-domain.com/DSC04229.JPG to C:\Users\moises\...	4211990	36.66	<div style="width: 36.66%;"></div>	Running	101.64 KB/s
Downloading sample-domain.com/New Folder/DSC04228.JPG to C:\U...	4074587	21.01	<div style="width: 21.01%;"></div>	Running	73.83 KB/s
Downloading sample-domain.com/New Folder/DSC04229.JPG to C:\U...	4211990	47.27	<div style="width: 47.27%;"></div>	Running	171.19 KB/s
Downloading sample-domain.com/New Folder/DSC04230.JPG to C:\U...	4371329	10.27	<div style="width: 10.27%;"></div>	Running	38.85 KB/s
Downloading sample-domain.com/New Folder/DSC04233.JPG to C:\U...	3820155	9.89	<div style="width: 9.89%;"></div>	Running	102.77 KB/s
Downloading sample-domain.com/New Folder/DSC04234.JPG to C:\U...	3148655	41.13	<div style="width: 41.13%;"></div>	Running	632.29 KB/s
Downloading sample-domain.com/New Folder/subfolder/DSC03588.J...	4201270	1.39	<div style="width: 1.39%;"></div>	Running	

At the bottom, there are buttons for 'Running (10)', 'Quered', 'Stopped', 'Failed', 'All (10)', 'Start All', 'Pause All', and 'Cancel All'.

The memory is not enough memory for all my processes!

Memory is not Enough

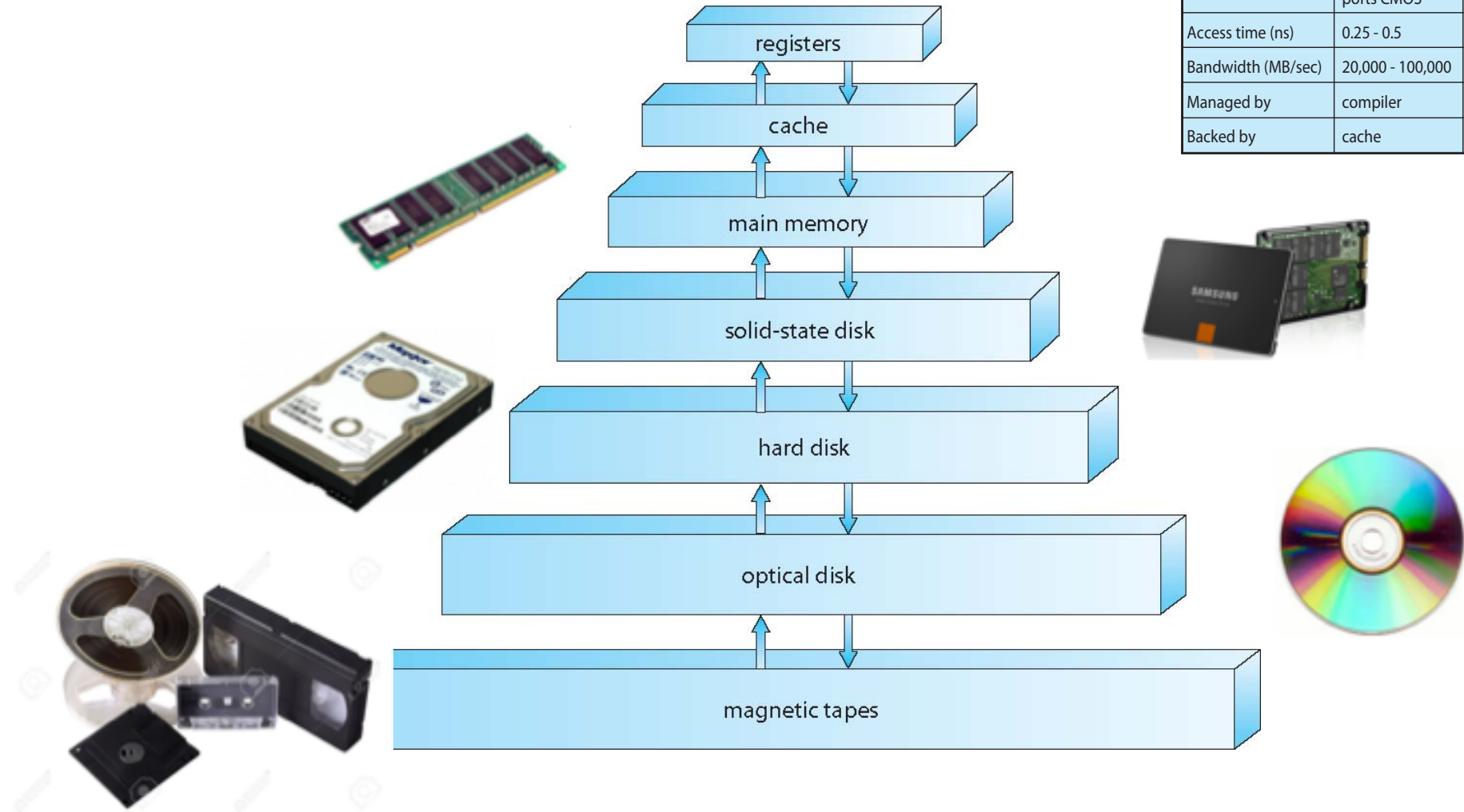
Keeping track of which parts of memory are currently being used and by whom

Deciding which processes and data to move into and out of memory

Allocating and deallocating memory space as needed

Memory Management
(Chapters 7 & 8)





Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Different Kinds of Storage Devices

Usually disks is used to store data that does not fit in main memory or data that must be kept for a “long” period of time

Entire speed of computer operation hinges on disk subsystem and its algorithms

Free-space management, Storage Allocation, and Disk Scheduling

Mass-Storage Management (Chapter 9)





OS provides uniform, logical view of information storage

Abstracts physical properties to logical storage unit : files, directories

Bits, Bytes, and Files

Access control to determine who can access what

Creating and deleting files and directories

Mapping and Backing files onto secondary storage

File-System Management
(Chapters 10 & 11)



Many I/O Devices

Hides peculiarities of hardware devices from the user

Memory management of I/O including buffering, caching, spooling

General device-driver interface



I/O Management
(Chapter 12)





Protection – any mechanism for controlling access of processes or users to resources defined by the OS

Security – defense of the system against internal and external attacks including: denial-of-service, worms, viruses, identity theft, theft of service



Protection & Security (Chapters 13 & 14)





キャプテン