CSE 325-Operating System

Lecture: 1-3

Dr. Shamim Akhter

Computer Science and Engineering

People and Places

Instructor:

Dr. Shamim Akhter @ 5th Floor, Room 626, CSE, EWU

http://ca.linkedin.com/pub/dr-md-shamim-akhter/66/966/a3a

- **Advising Hour:**
  - Sunday 3:10-4:40
  - Thursday 1:30-3:30

- **E-mail:** shamimakhter@ewubd.edu

- **Course Web Link:**
  - Google Drive: goo.gl/4IsTZN

- **Course Schedule:**

2 Lectures per week (90 Minutes each)
1 Lab per week (120 Minutes each)

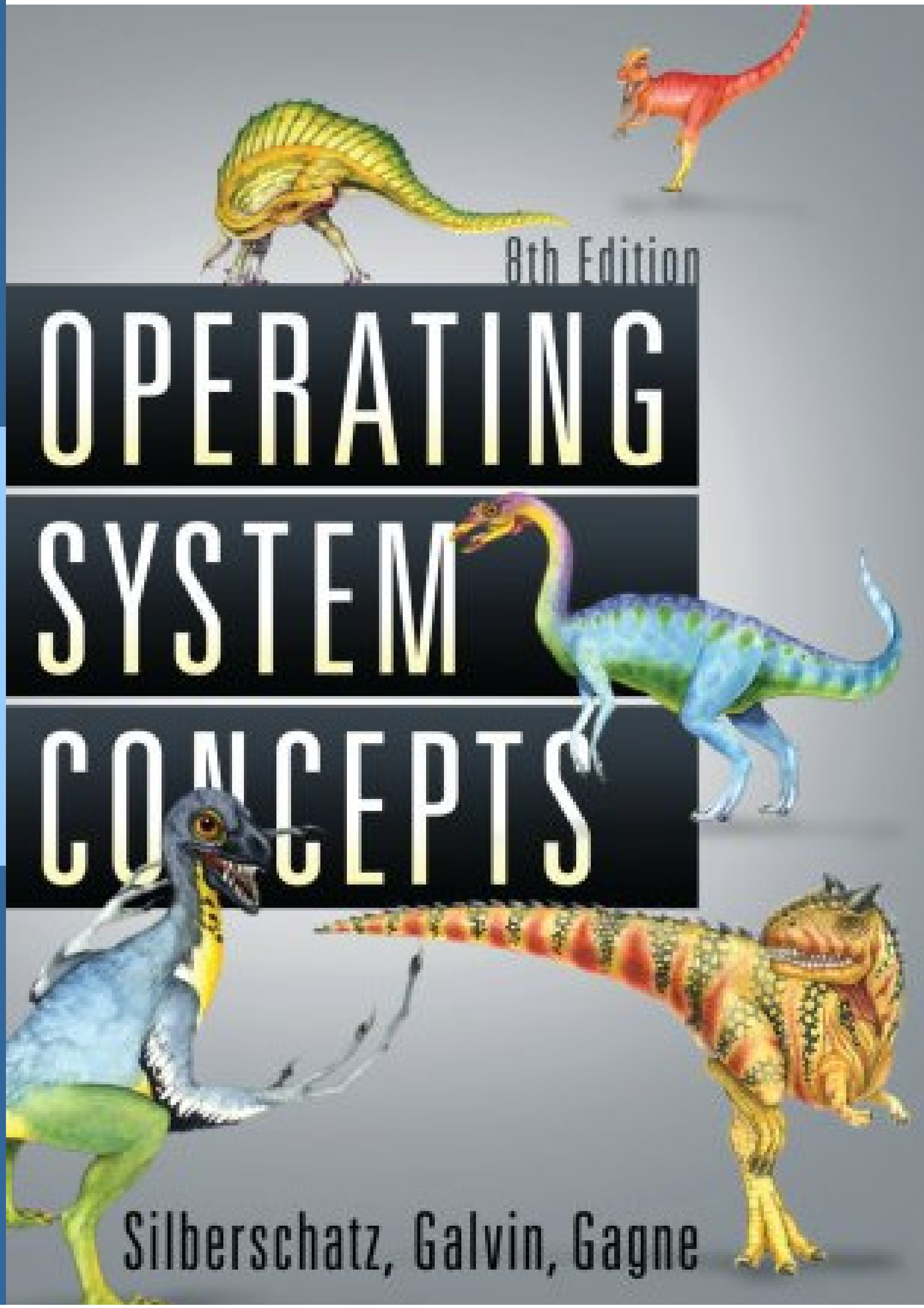
Course Description

- This course introduces the principles and techniques for the **design and implementation of operating systems**:
  - **interrupts,**
  - **computer resource management**
    - memory management,
    - processor management,
    - I/O management,
    - file management,
    - process management and
    - security management,
  - **inter process communication.**
- Additionally topics are: Multithreaded OS, and Concurrent computations.
- This course includes a project implementation.

Course Outcomes

Successful completion of this course, you will be able to:

- Identify different components of Operating System.
- Describe different methods of resource management.
- Apply resource management techniques for resource constained problems.
- Investigate memory and I/O management issues.


8th Edition
OPERATING
SYSTEM
CONCEPTS
Silberschatz, Galvin, Gagne

Text book

WELCOME TO

CSE 325-Operating System Concepts

Spring, 2017

Lecture 1: Introduction

Objective

Overview of the major OS components.

Computer System Structure

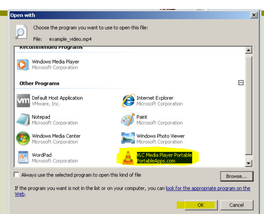
Hardware – provides basic computing resources

- CPU, memory, I/O devices

Operating system

- Controls and coordinates use of hardware among various applications and users

Application programs

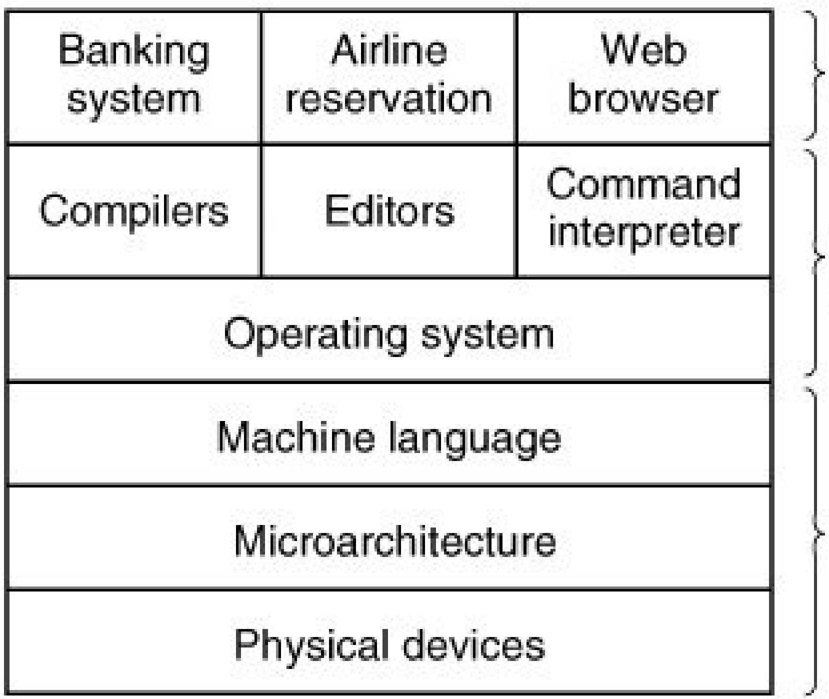


- define the ways in which the system resources are used to solve the computing problems of the users
- Word processors, compilers, web browsers, database systems, video games

Users

People, machines, other computers

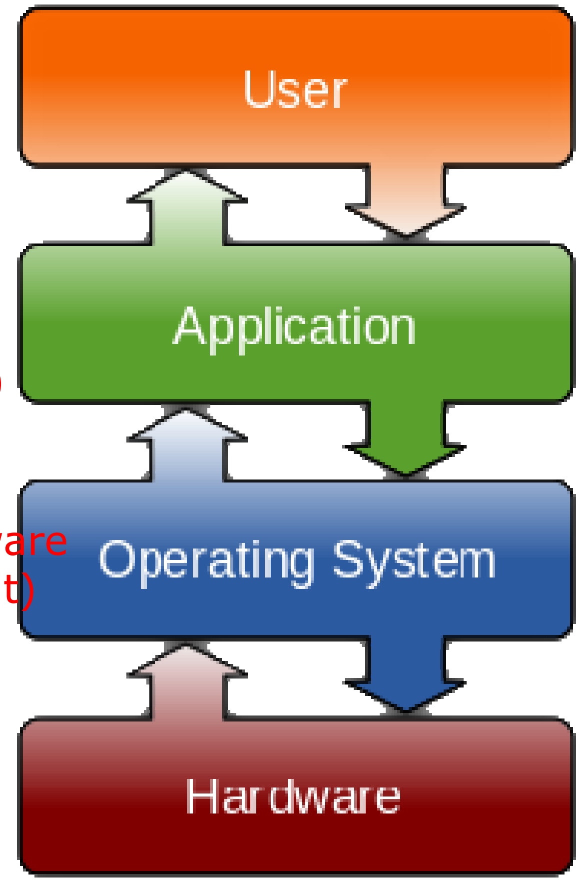
Components of a Computer System



Application programs
services to the user
(e.g.Word Processor)

System programs
services to the hardware
(e.g.disk defrangment)

Hardware



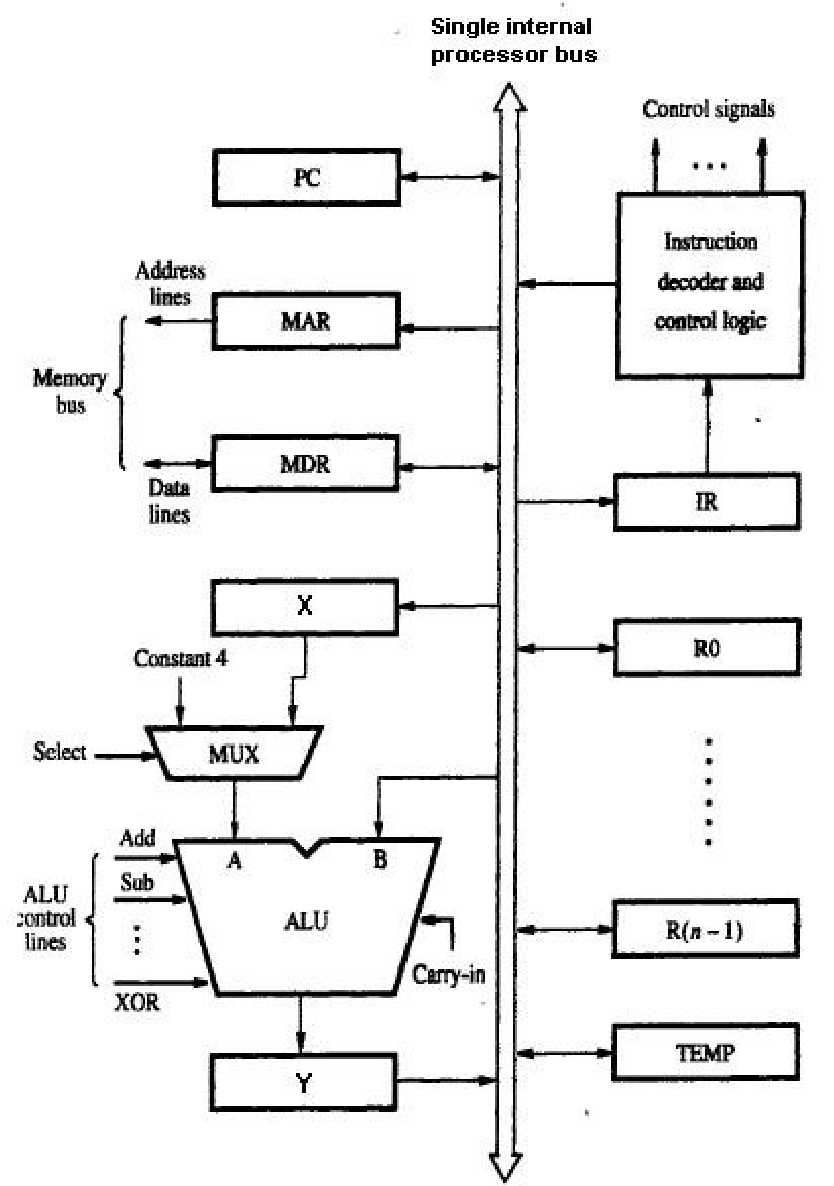
Tanenbaum@Modern OS

Microarchitecture- Computer Organization

- The way a given instruction set architecture (ISA) is implemented on a processor

The ISA includes :
- the execution model,
- processor registers,
- address and data formats.

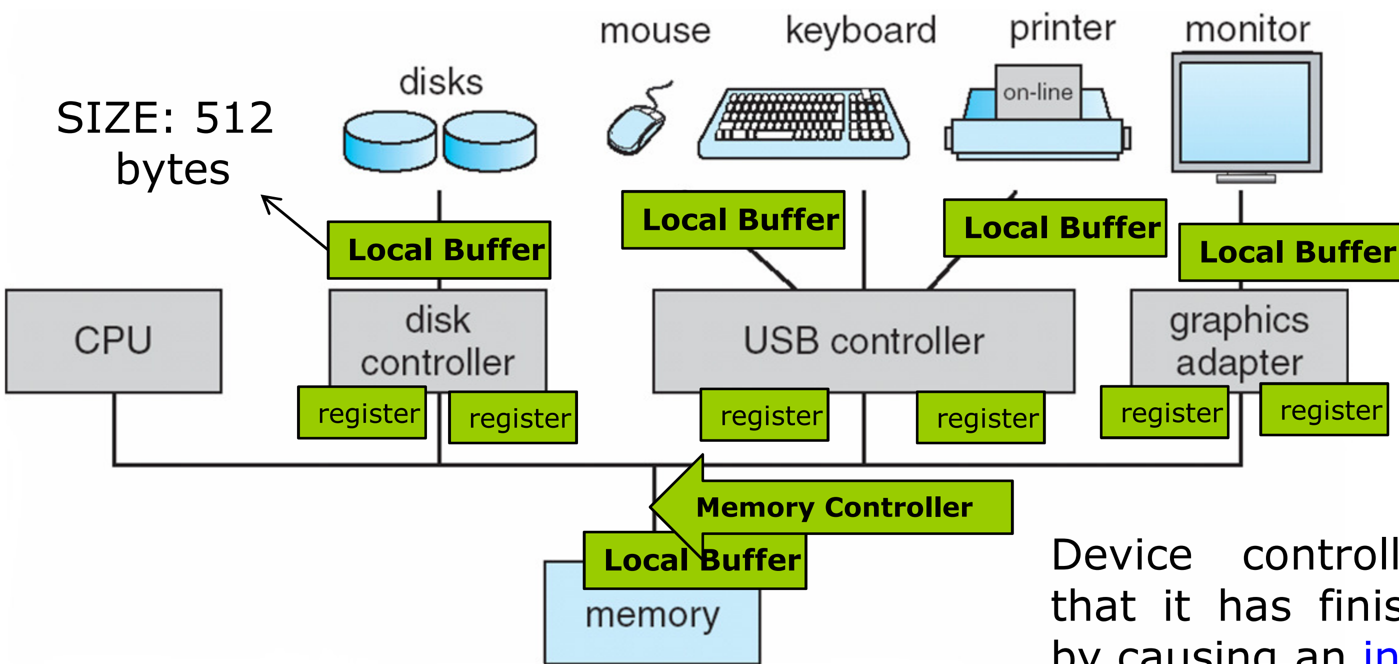
The microarchitecture includes:
- the parts of the processor and
- how these interconnect and
- interoperate to implement ISA.



Single Bus Organization

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory & CPU cycles

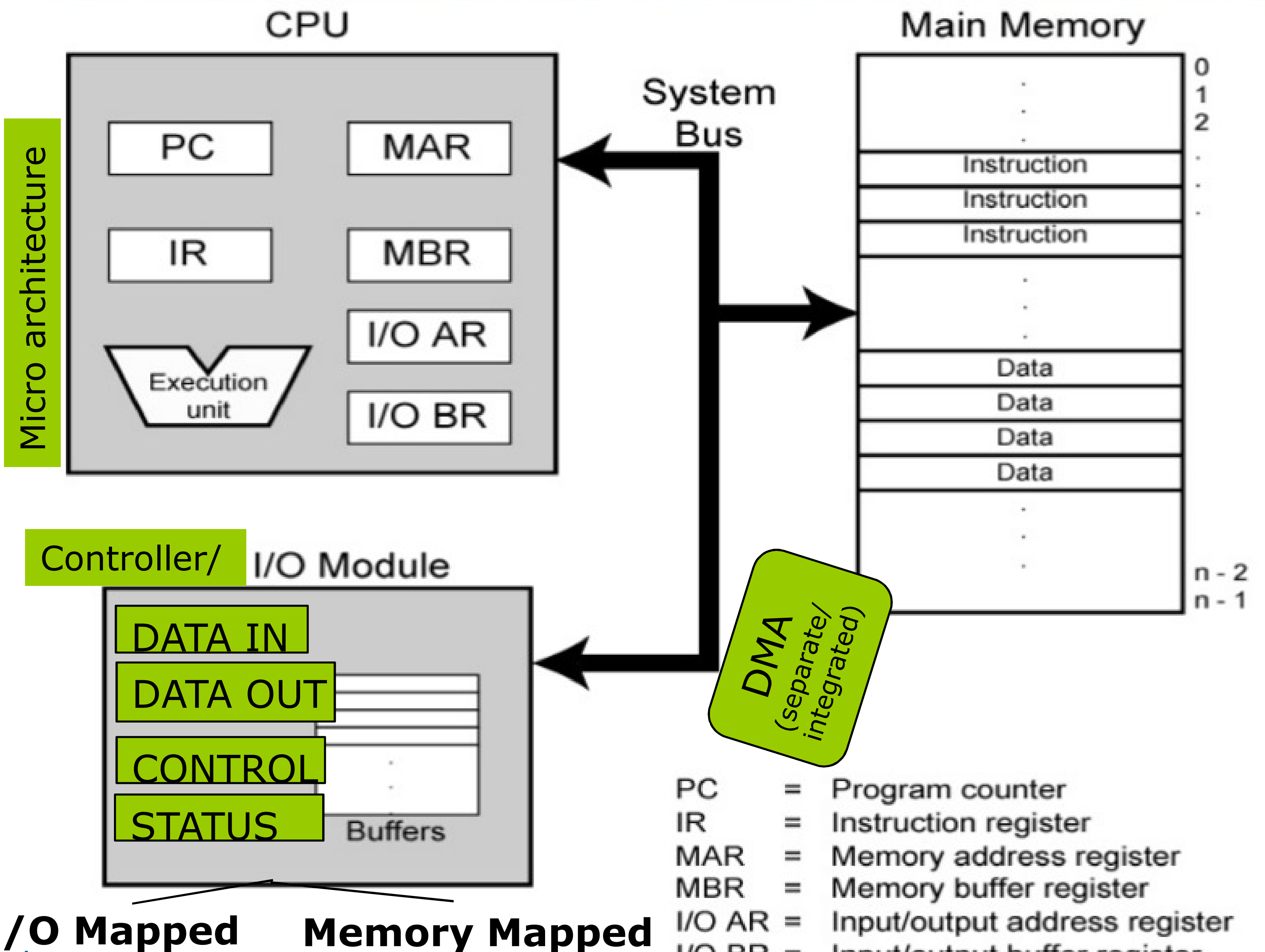


Device controller informs CPU that it has finished its operation by causing an interrupt.

The **controller** is the hardware that controls the communication between the system and the peripheral drive unit. It takes care of low level operations such as error checking, moving disk heads, data transfer, and location of data on the device.

Computer Components: Top Level View

William Stallings, Operating systems, 7th Edition, **Chapter 1, Sec: 1.1**



I/O Mapped: Special I/O instruction in/out instruction

Example from the Intel architecture: `out 0x21,AL`

```
000-00F DMA Controller
020-021 Interrupt Controller
040-043 Timer
3D0-3DF  Graphics Controller
```

Memory mapped I/O: No Special Instruction load/store instructions

Controller registers/memory maps in physical address space. I/O accomplished with load and store instructions

Example: **load 0xFFFF0000, AL (Control)**
load 0xFFFF0004, BX (Data)

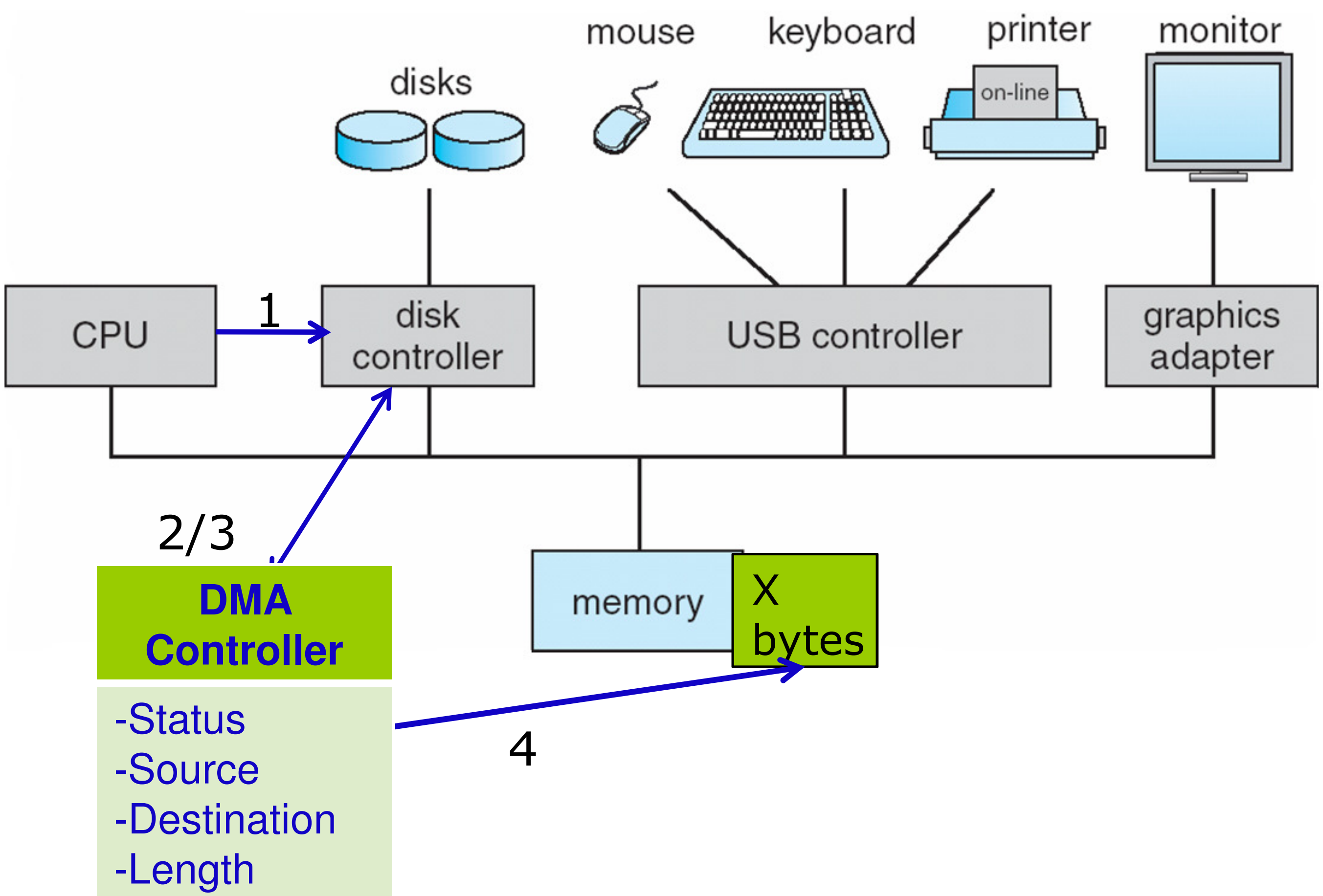
1-Ready
0-not

32 bits		**R**

32 bits Data

DMA – Direct Memory Access

- High Speed, I/O device



Block of data

How does a program execute?

- C Program

int A[2]; int C[2]; int B[2];

P=A[0];

Q=B[0];

C[0]=P+Q;

First.c

Compile

- Assembly

Load $t0, [A]

Load $t1, [B]

ADD $t3, $t0, $t1

STORE $t3, [C]

First.asm

First.o

Linker

- Executable

0001……………………..00

0000……………………..01

00011…………………..11

100……………………….11

Loader

- Memory

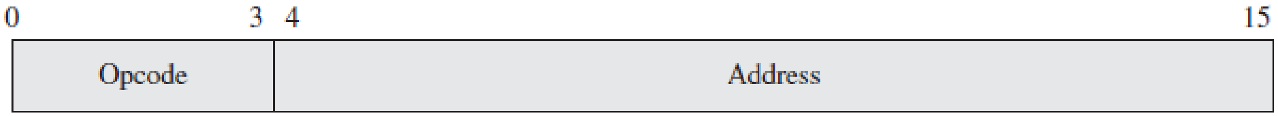
1	0001……………………..00
2	0000……………………..01
3	00011…………………..11
4	100……………………….11
5	

PC=1

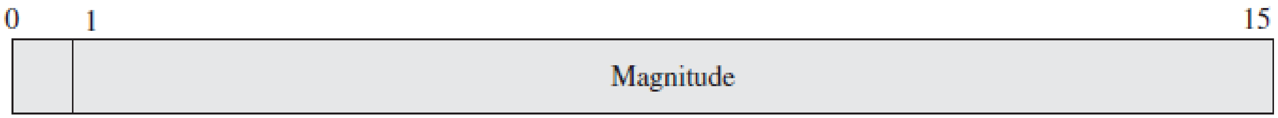
IR= 0001…………………..00

Hypothetical Machine

Example:
Program Execution



(a) Instruction format



(b) Integer format

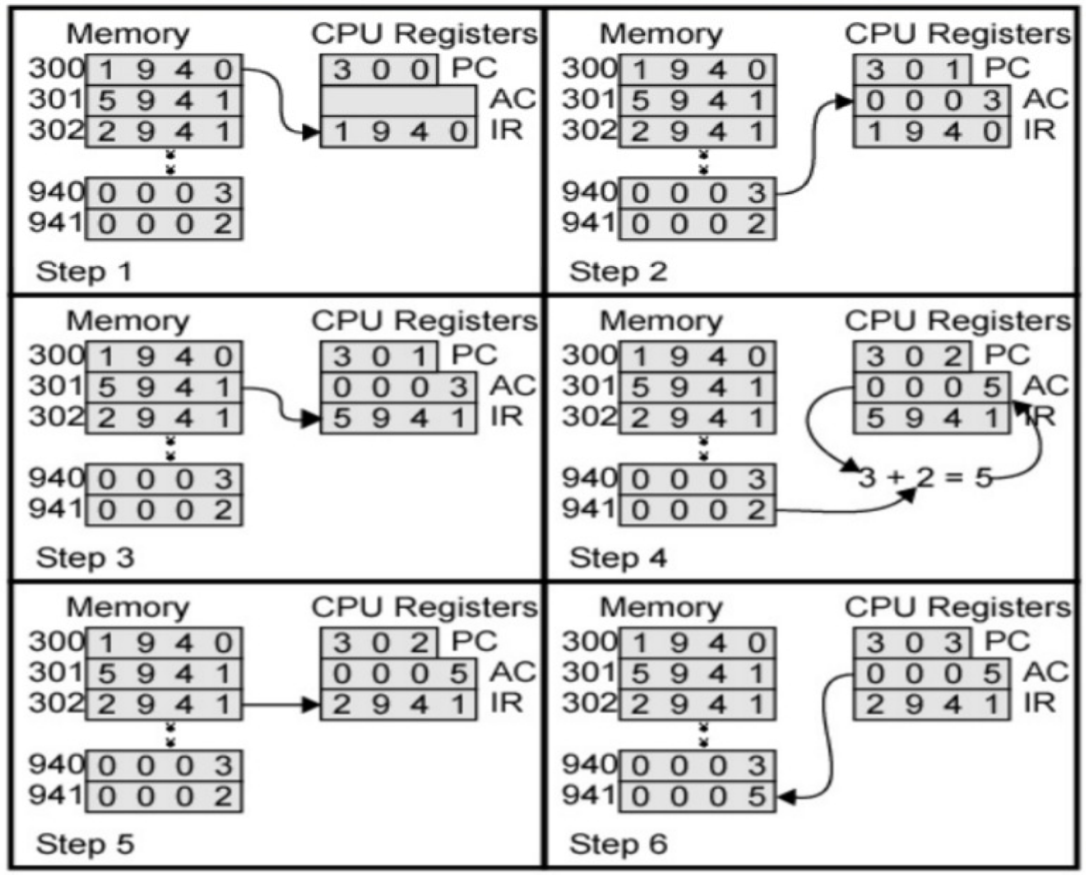
Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 3.4 Characteristics of a Hypothetical Machine



CPU to Memory

William Stallings, Operating systems, 7th Edition, **Chapter 1, Sec: 1.3**

) Suppose the hypothetical processor has three I/O instructions: LOAD, ADD and STORE. 16-bit instruction format is as follows:

Opcode (4-bit)	Address (12-bits)

0011(3) Opcode means Load AC from I/O and the 12-bit address identifies a particular external device.
0101(5) Opcode means Add to AC from memory and the 12-bit address identifies memory location.
0111(7) Opcode means Store AC to I/O and the 12-bit address identifies a particular external device.

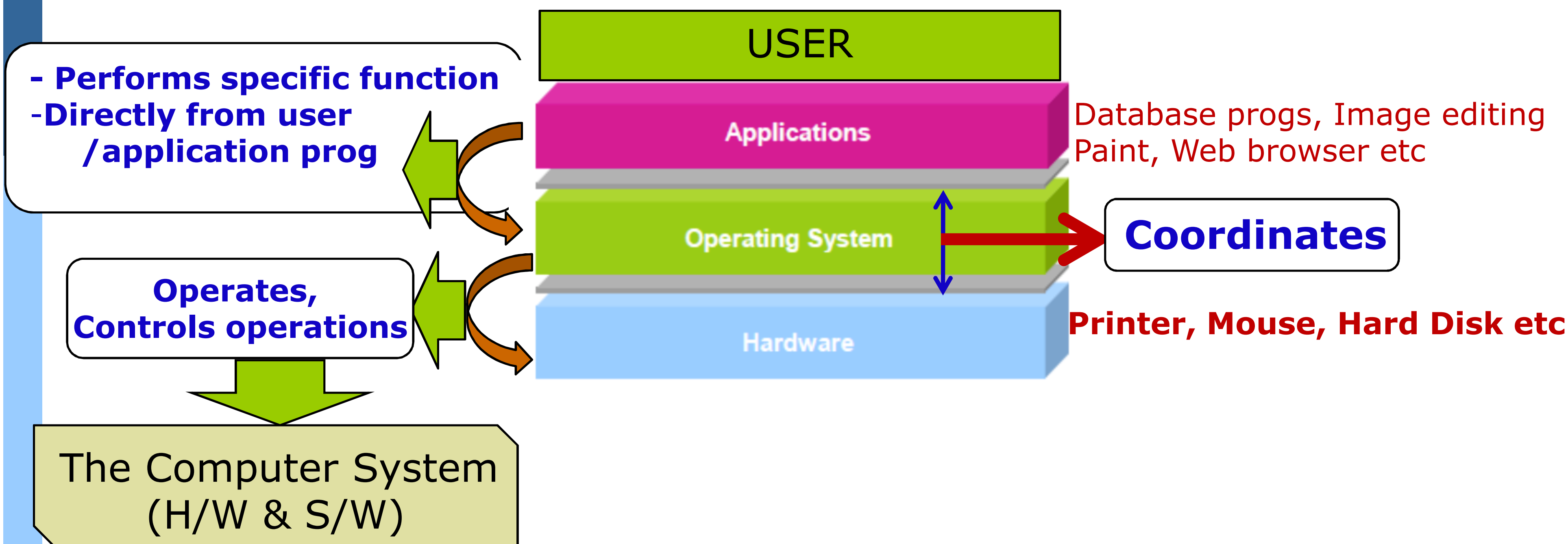
Assume that the next value retrieved from device 5 is 3 and that location 940 contains a value of 2.

Execute the following program and find the values of PC, AC and IR at each step.

300	3 0 0 5	PC=300 AC=? IR=?
301	5 9 4 0	PC=? AC=? IR=?
302	7 0 0 6	PC=? AC=? IR=?

What is an Operating System?

- A program that acts as an intermediary bet^n a user & H/W



GOALS:

» Convenient to use

» Execute user programs & make solving user problems easier

» Use H/W in an efficient manner

Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use.

One goal of the operating system is to increase the utilization of resources.

Utilization =useful time/total time

For example, the OS should avoid wasting CPU time because the disk is rotating or wasting switching among tasks. Waiting for I/O.

- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer. e.g. Memory Protection.

An example comparing life with/without OS

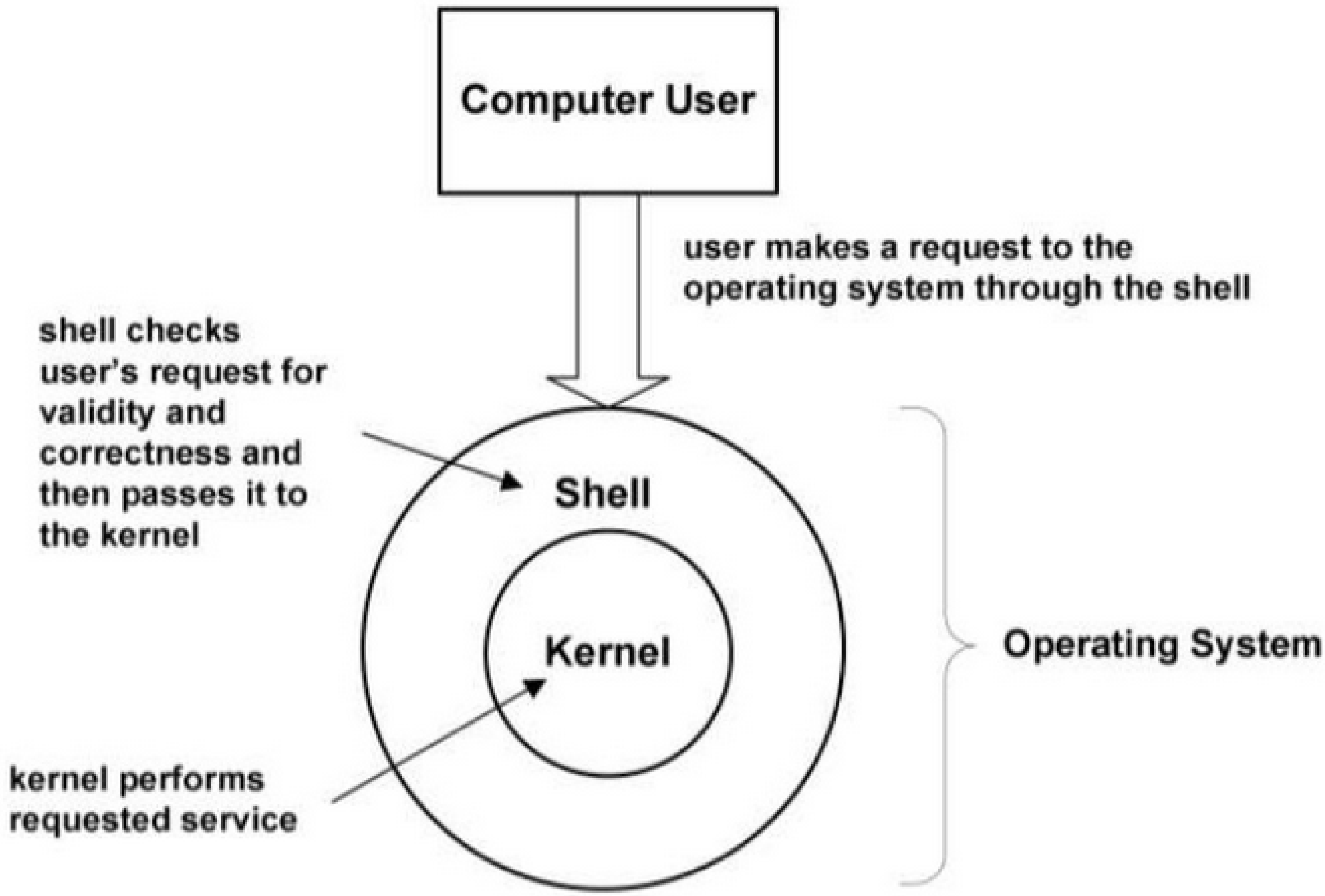
Life with an OS

```
file = open ("test.txt",
   O_WRONLY);
write (file, "test", 4);
close (file);
```

Life without an OS

- Blocks, platter, track, and sector
- Where is this file on disk? Which platter, track, and sectors?
- Code needs to change on a different system

How Does User Communicate?



The Kernel and the Shell

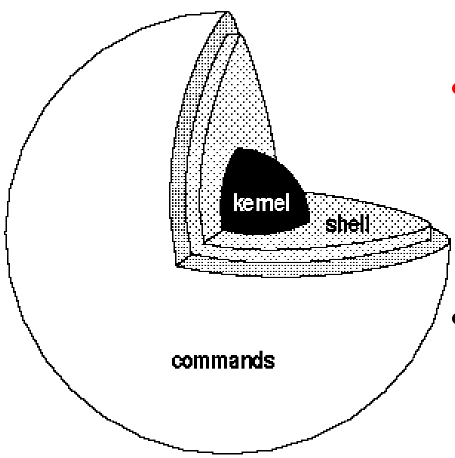
The Kernel and The Shell

An Operating system has two parts: kernel and shell

KERNEL (resides in main memory):

- is the heart and soul of an OS. Useful applications and utilities are added over the kernel, then the complete package becomes an OS.
- directly controls the computer hardware by performing OS services **using specific system calls or requests or hardware interrupts.**
- hides the hardware details from the user and application programs.

Example,



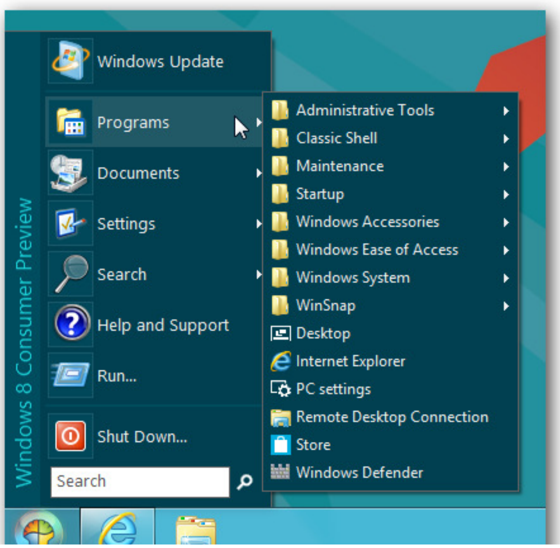
- Linux is a kernel as it does not include applications-file-system utilities, windowing systems and graphical desktops, system administrator commands, text editors, compilers etc.
- So, various companies add these kind of applications over Linux kernel and provide their operating system like **ubuntu, centOS, redHat** etc.

An Operating system has two parts: kernel and shell

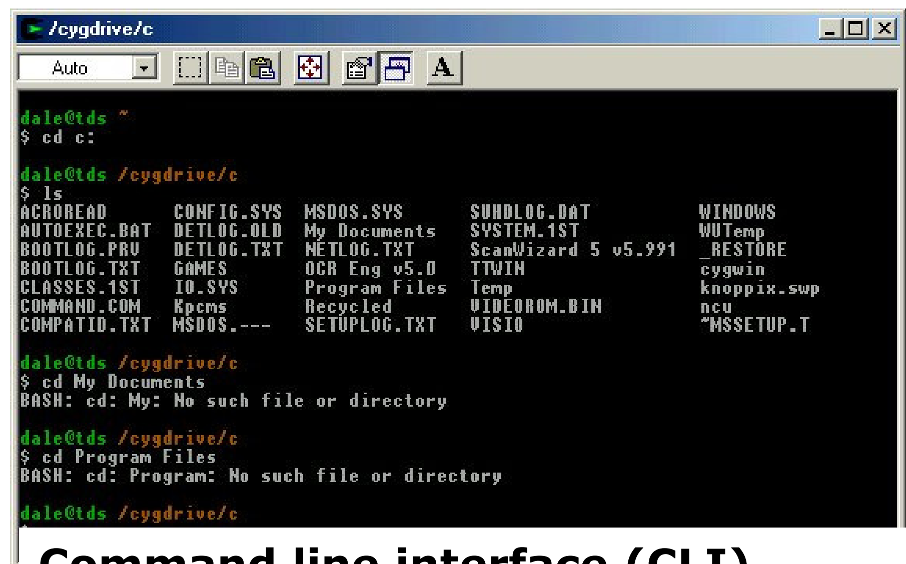
shell (command interpreter): Part of OS

- Serves as the interface between user and kernel.
- User enters commands via shell in order to use the kernel service.

Type of shells

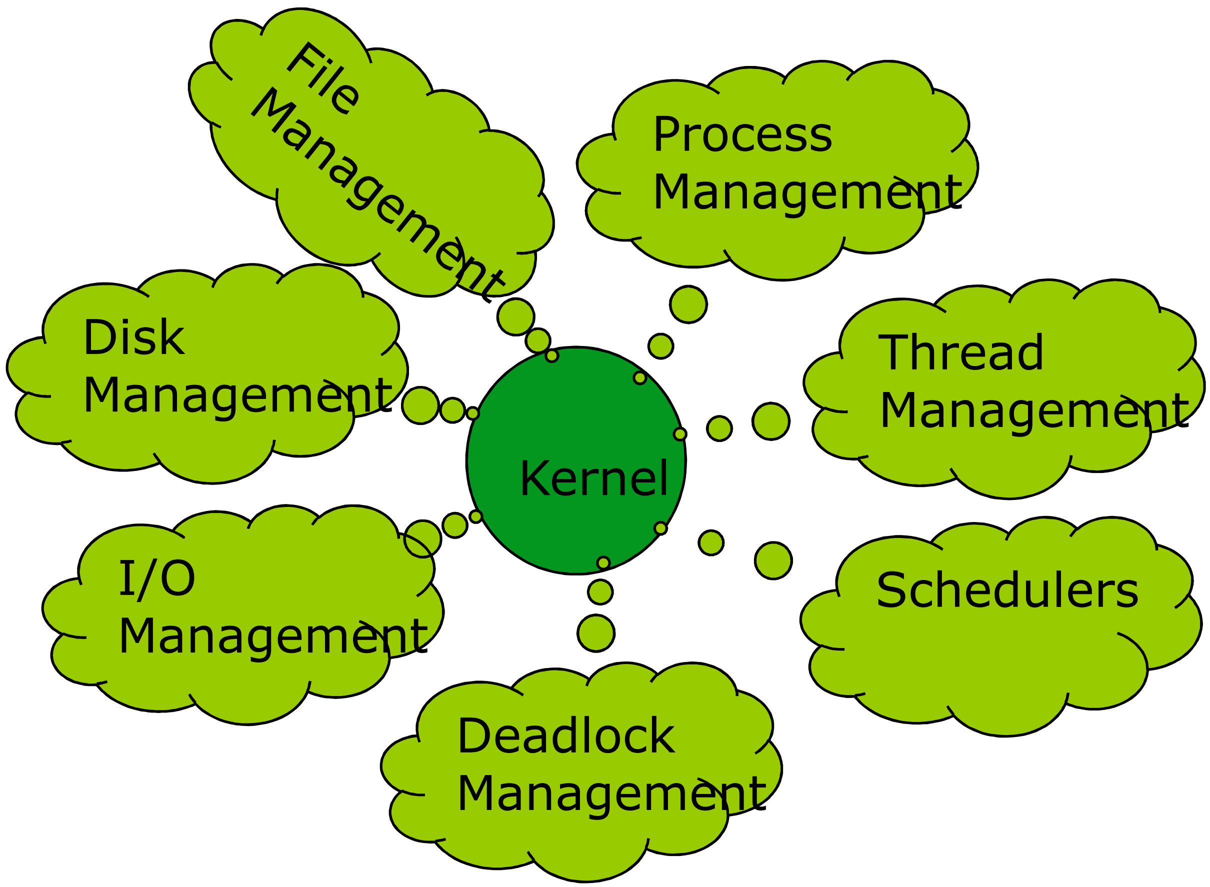


Graphical User Interface (GUI)
Windows/Mac OS

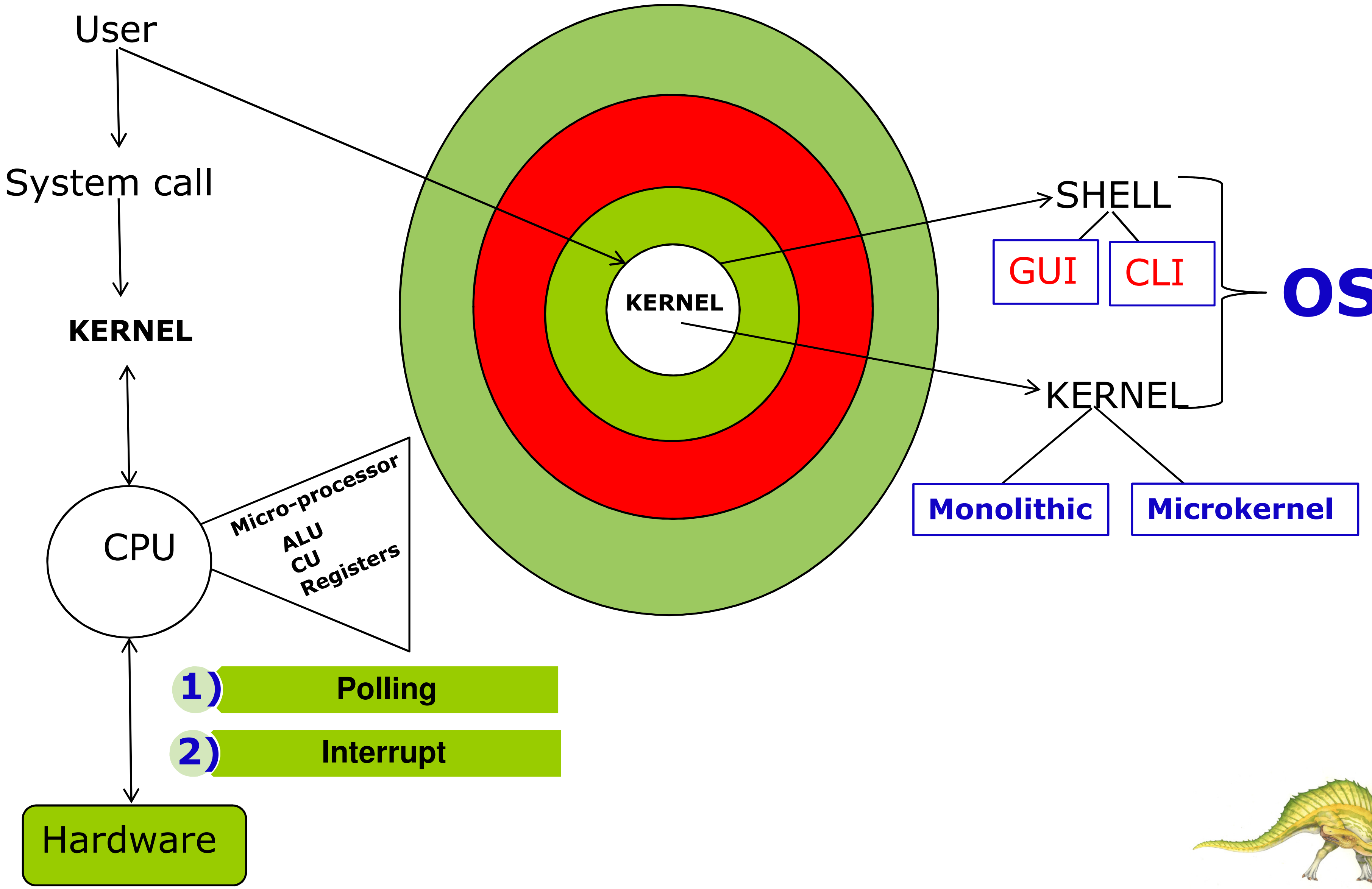


Command line interface (CLI)
Dos/Linux

OS Organization



How does user communicate with OS



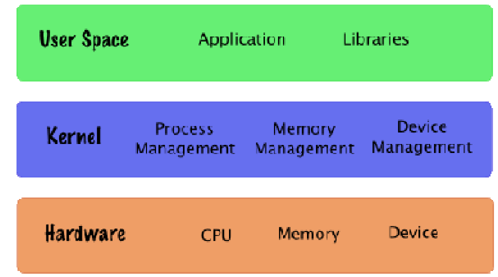
Kernel Types

Monolithic Kernel	Micro Kernel
All the parts of a kernel - Scheduler, file system, memory, device drivers etc. are in one unit within the kernel.	Only the important parts are in kernel - IPC, basic scheduler, basic memory management, basic I/O primitives etc. - Others are in user space.
Signals & sockets to ensure IPC	-Message Passing system to ensure IPC
Advantages - Faster processing	Advantages - Crash resistant - Portable - Smaller size
Disadvantages -Crash insecure -Porting Inflexibility -Kernel size explosion	Disadvantages - slower processing due to additional message passing
MSDOS, Unix, Linux	Windows NT

Transition from User to Kernel Mode

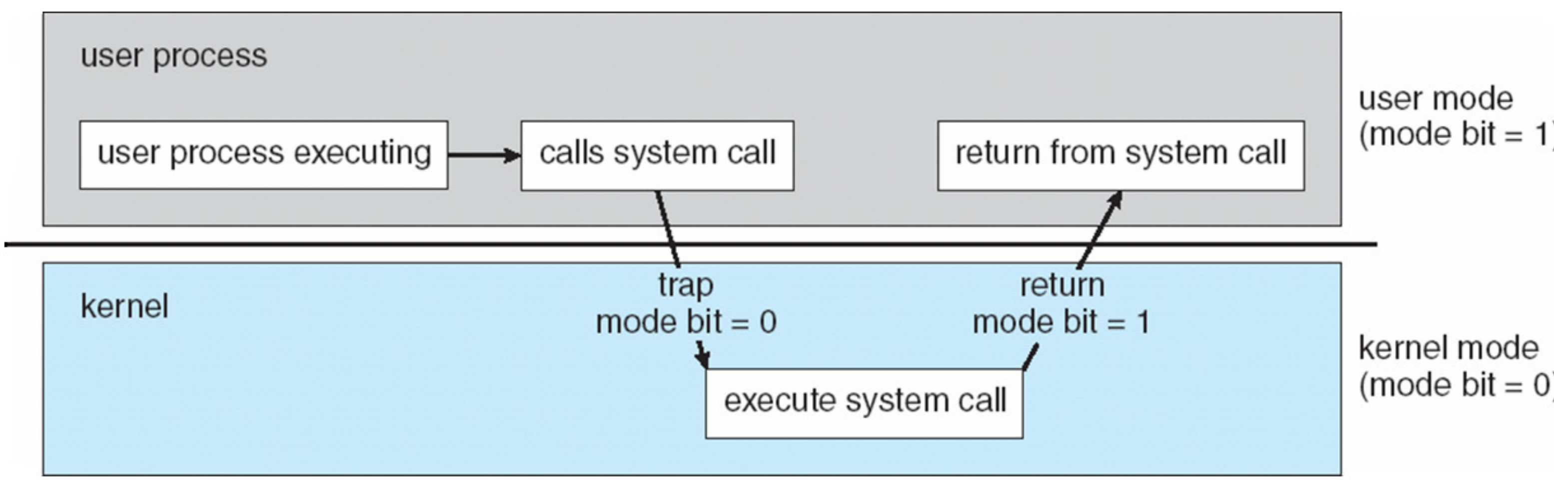
Dual-mode operation allows OS to protect itself and other system components.

- User mode
- Kernel/Supervisor/Monitor/System Mode.



Mode bit @ Process Status Word (PSW)-Resister

- Provides ability to distinguish when system is running user code or kernel code.
- Some instructions designated as **privileged**, only executable in kernel mode. Example: load/save from protected memory, initiate I/O etc.
- System call changes mode to kernel, return from call resets it to user.



Working with User and Kernel mode

scanf(“%d”,&a); // $t0 holds a value

PC

0	move $v0, 5
1	systemcall
2	store $v0, 0($t0)

1

2

Kernel I/O subsystem

3

4

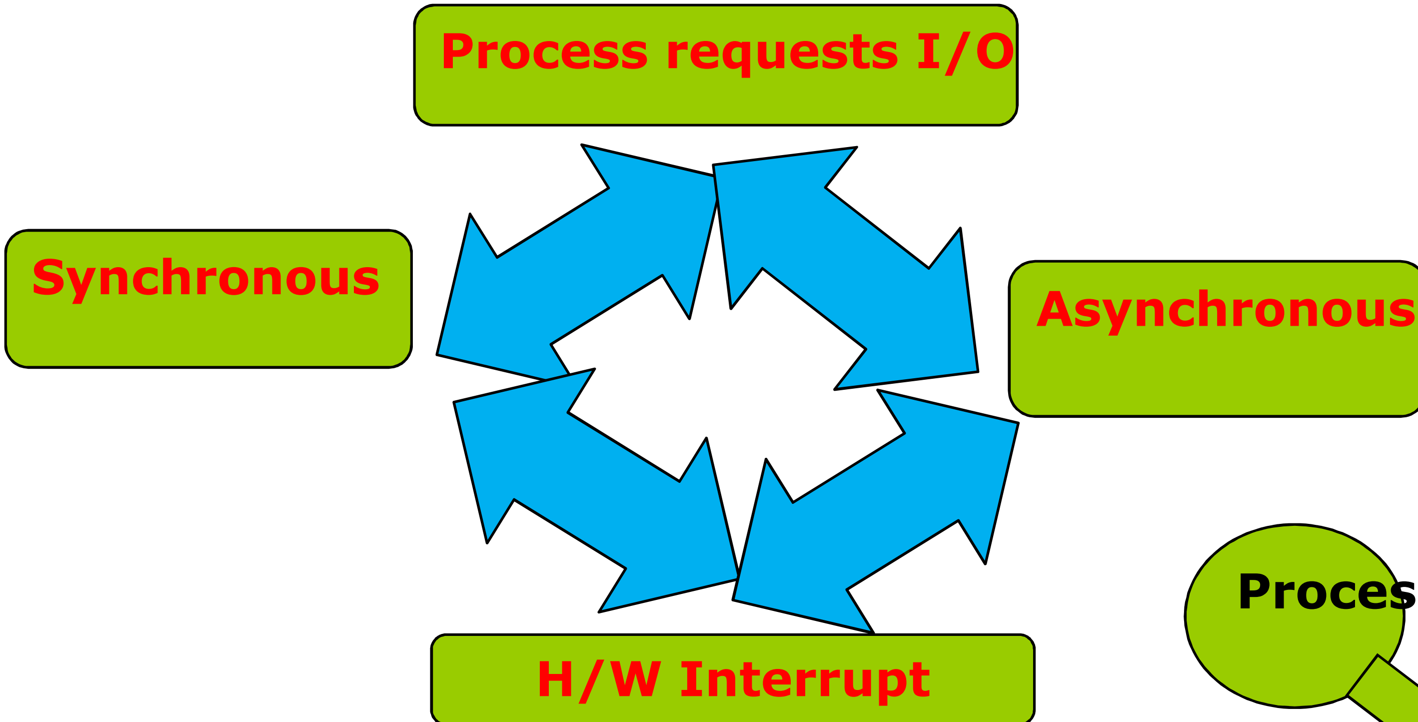
	RAM
0150	systemcall (SWI)
0160	
0170	store $v0, 0($t0)
0058	Device driver (SWI)
0090	I/O command, data control Communication with H/W
	GOTO RA=0160

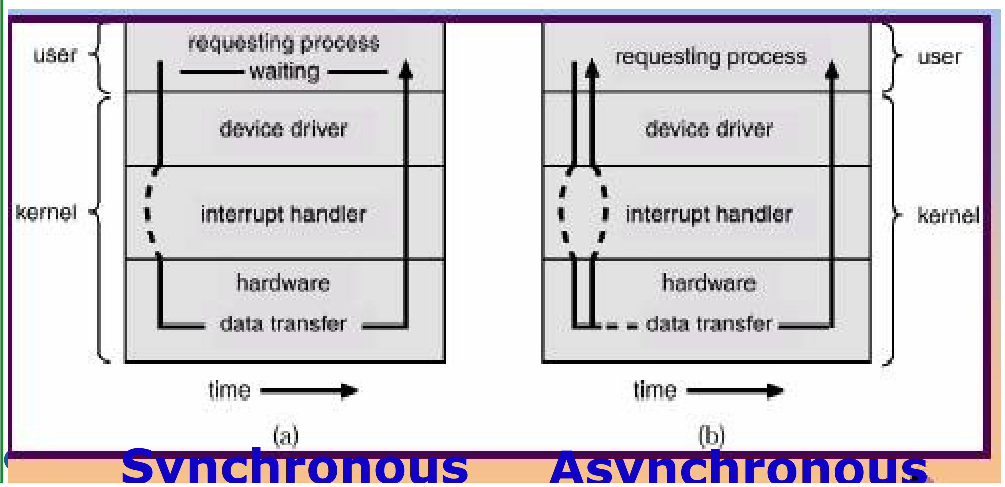
IVT- Interrupt Vector Table @ RAM-0000:0000H

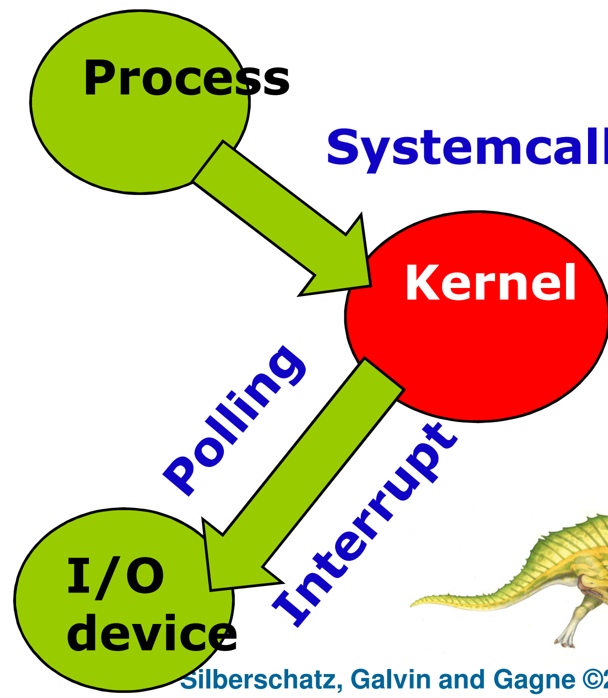
Service #	Code Address
….	…..
005	0058
019	0090

Modified figure-13.10 @ galvin

How does H/W communicate with Kernel/vice versa?







POLLING

"Polling is like picking up your phone every few seconds to see if you have a call. Interrupts are like waiting for the phone to ring."

- Interrupts win if processor has other work to do and event response time is not critical
- Polling can be better if processor has to respond to an event ASAP
  - May be used in device controller that contains dedicated secondary processor

CPU POLL

Poll a device takes three (3) CPU cycles

-read device register
-do logical-and to extract a status bit
-bit =0 then submit job to device

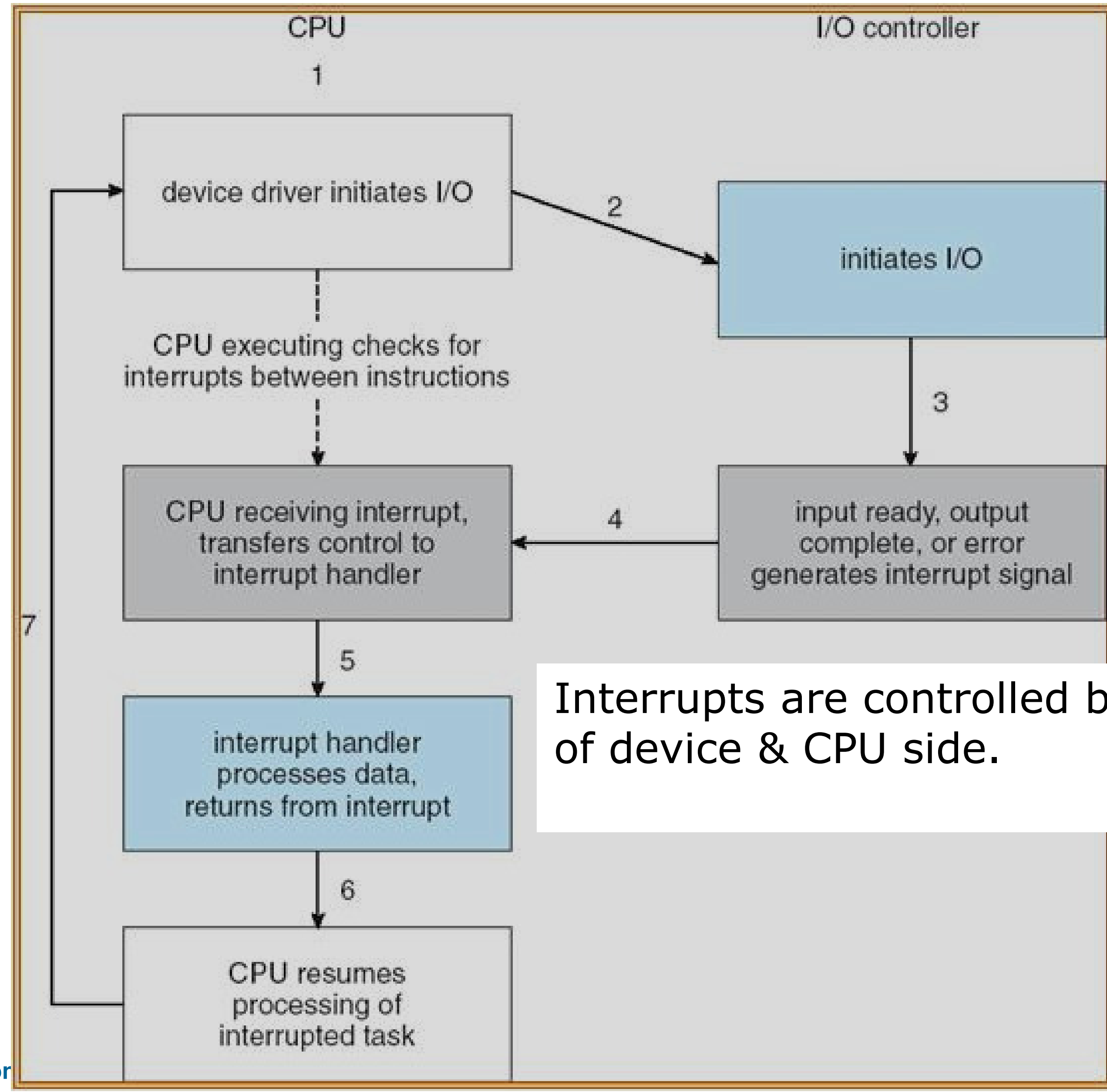
WHY Interrupt?

- People like connecting devices
  - A computer is much more than the CPU
    - Keyboard, mouse, screen, disk drives
    - Scanner, printer, sound card, camera, etc.
- These devices occasionally need CPU service
  - But we can't predict when
- External events typically occur on a macroscopic timescale
  - we want to keep the CPU busy between events

☞ Need a way for CPU to find out devices need attention

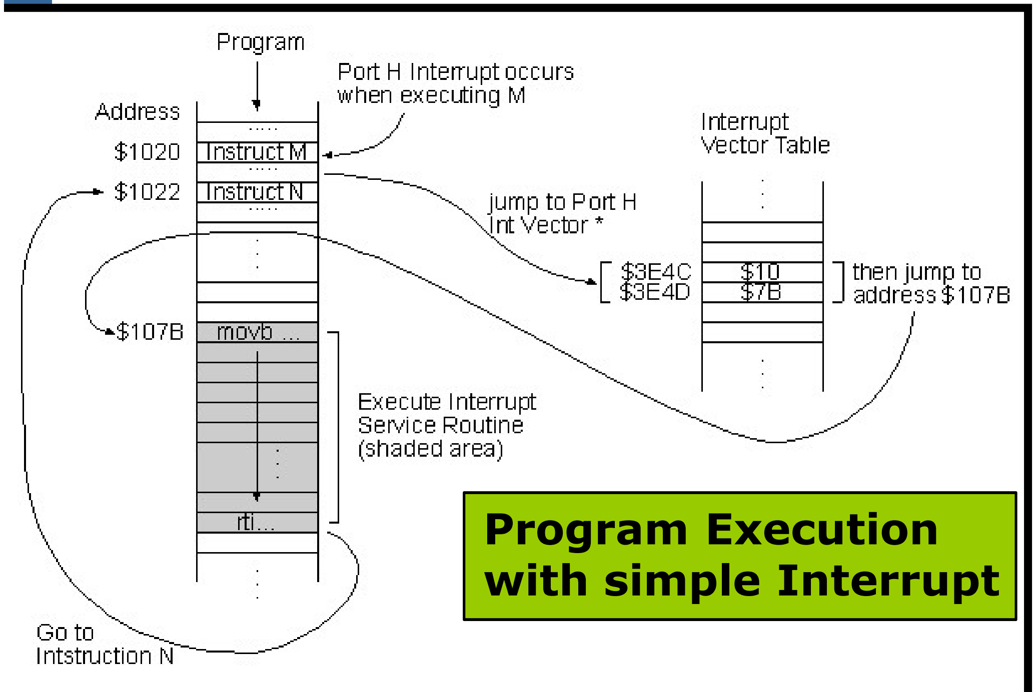
How does Interrupt work?

Interrupt-driven I/O Cycle
Chapter 13 Figure 13.3

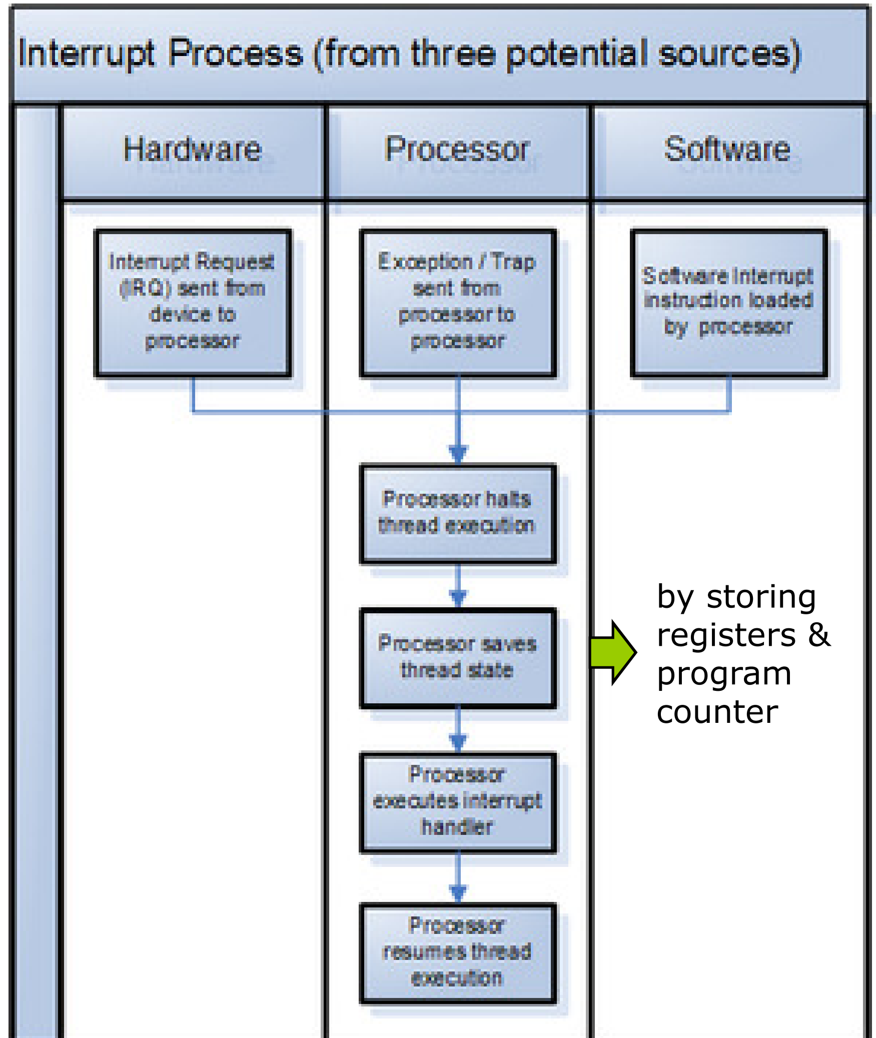


Interrupts are controlled by status bits of device & CPU side.

Program Execution with Interrupt



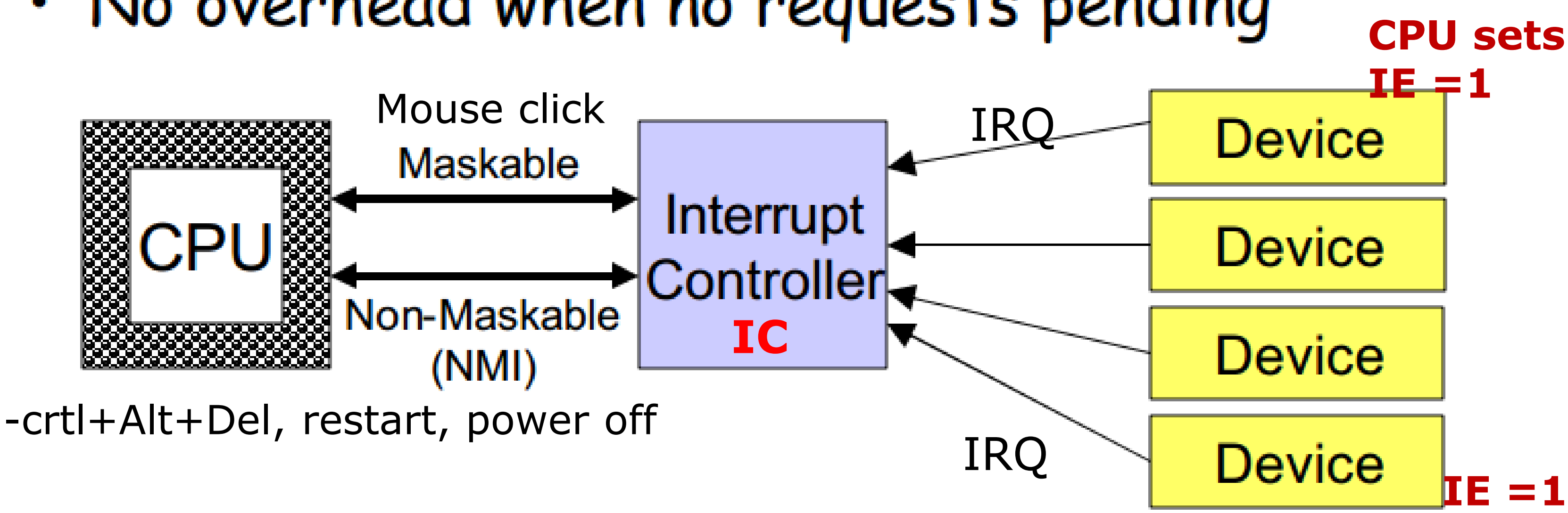
Program Execution with simple Interrupt



by storing registers & program counter

Alternative: Interrupt Vector

- Give each device a wire (interrupt line) that it can use to signal the processor
  - When interrupt signaled, processor executes a routine called an interrupt handler to deal with the interrupt
  - No overhead when no requests pending



CPU sets IE =1, Device can send interrupt (Not Masked)

IC checks interrupt mask register(IMR) bit is masked (disable) or not, if IMR sets to 1,IRQ will not be send to processor, otherwise IMR sets to 0.

Interrupt Vector Table

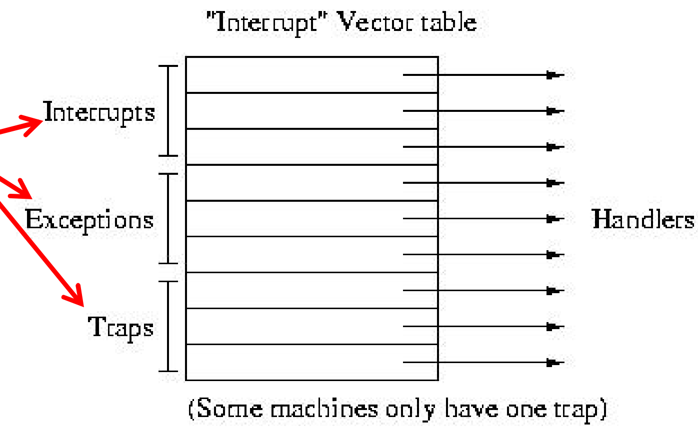
vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Nonmaskable

Priority will be set
Lower has Higher
priority

Different Interrupts

An operating system is interrupt driven



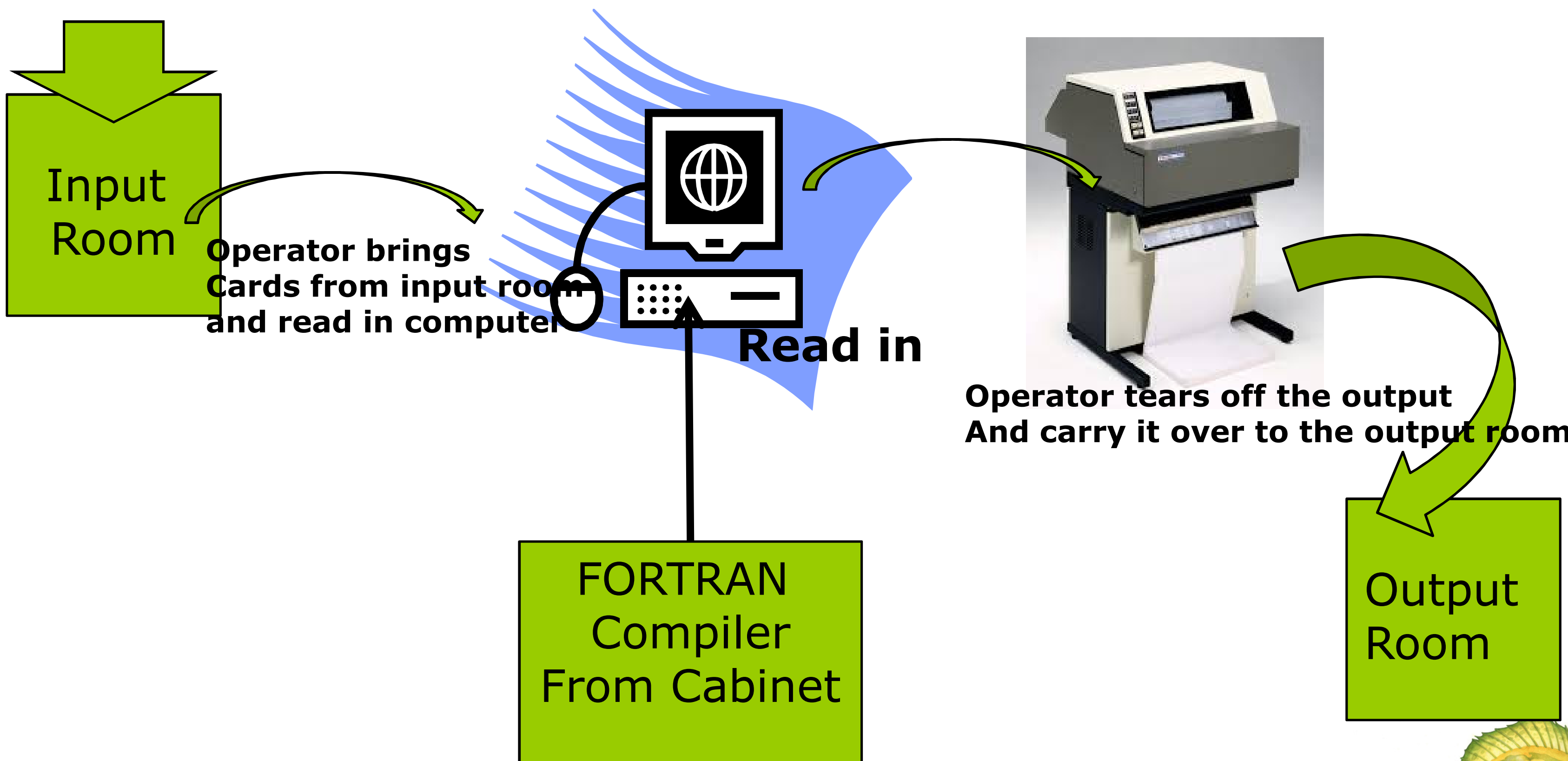
Interrupts (Hardware events)	• **external hardware event (external to the CPU).** • **Higher priority**
Exception (Software events)	• **automatically generated unexpected events occur in response to some exceptional condition.** • **illegal program actions that generate an interrupt.**
Traps (Software Interrupts)	• a programmer initiated and expected transfer of control to a special handler routine. • **user program can ask for an operating system service** • **unconditional –control always transfer to pre-defined procedure.**
System Call (Software Interrupts)	• simply sets up some registers with the needed system call/ OS service number, and execute the Trap/software interrupt.

Lecture 2: OS Evaluation

Computers were in Universities / Large companies

Programmers wrote programs (FORTRAN/ Assembly)on paper.

Punched the program into CARDS

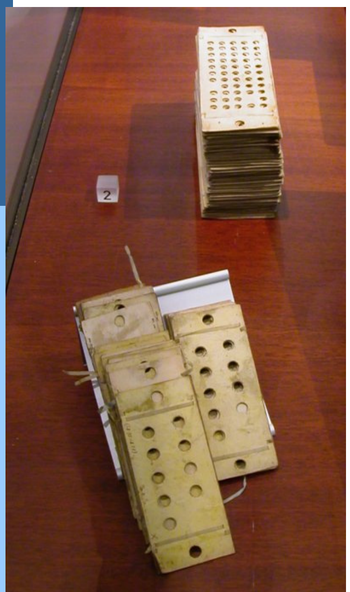


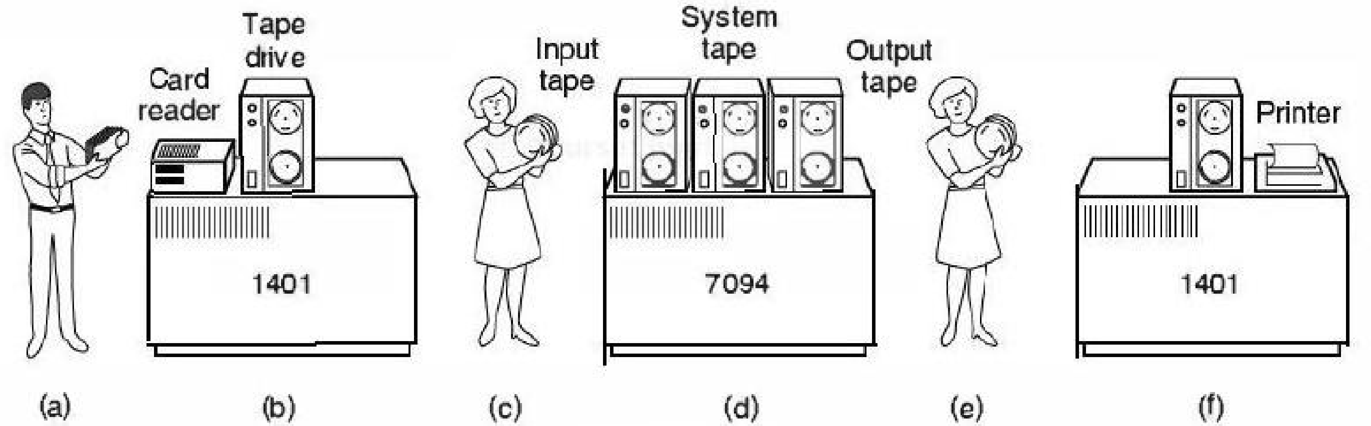
Wasting time while operators were walking around the machine room

Supervisor/Operator Control

CR ----> MT MT ----> CPU CPU ----> LP

IBM 7094





An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

- Problems
  - Long turnaround time - up to 2 DAYS!!!
    - Debugging…
  - Low CPU utilization
    - I/O and CPU could not overlap;
    - slow mechanical devices.

From John Ousterhout slides

Tanenbaum@Modern OS

BATCH Processing

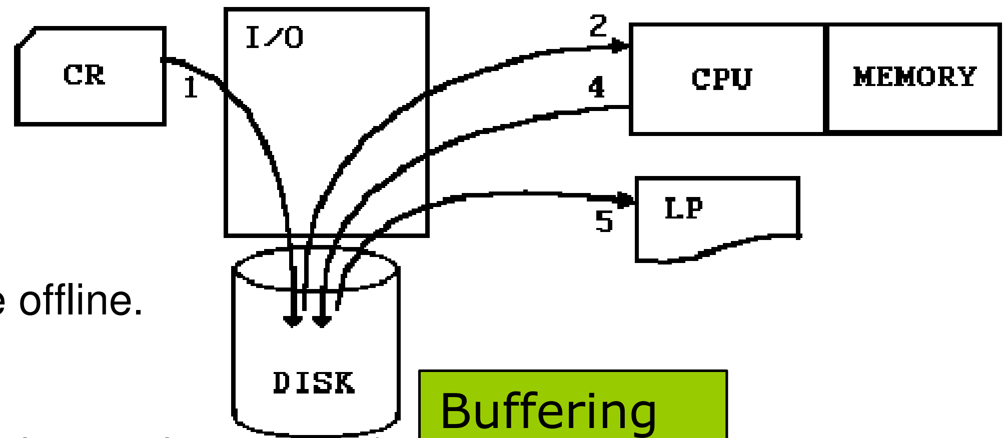
Process 1			Process 2			Process 3			
I1			I2			I3			
	C1			C2			C3		
		O1			O2			O3	
2	4	2	2	3	2	2	7	3	27

I= Input
C=Computation
O=Output

OS task was simple
To transfer control from one job to another

Batch Systems - Issues

- **Solutions to speed up I/O:**

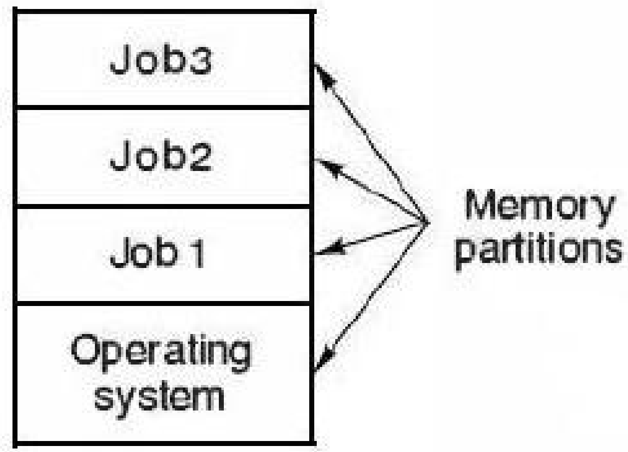


Buffering

- Offline Processing (Previous)
  - load jobs into memory from tapes,
  - card reading and line printing are done offline.
- Online Spooling (*simultaneous peripheral operations on-line*)
  - A type of buffering, to disk (present)
    - Random access device, unlike tape
    - Modern: print spooler
  - Use disk as large storage for reading as many input files as possible and storing output files until output devices are ready to accept them.
  - Allows overlap - I/O of one job with computation of another.
  - Introduces : job pool (Queues-job queue, ready queue)
    - allows OS to choose next job to run so as to increase CPU utilization

Multiprogramming

- Use interrupts to run multiple programs simultaneously
  - When a program performs I/O,
    - instead of polling, execute another program till interrupt is received.
  - Efficiency

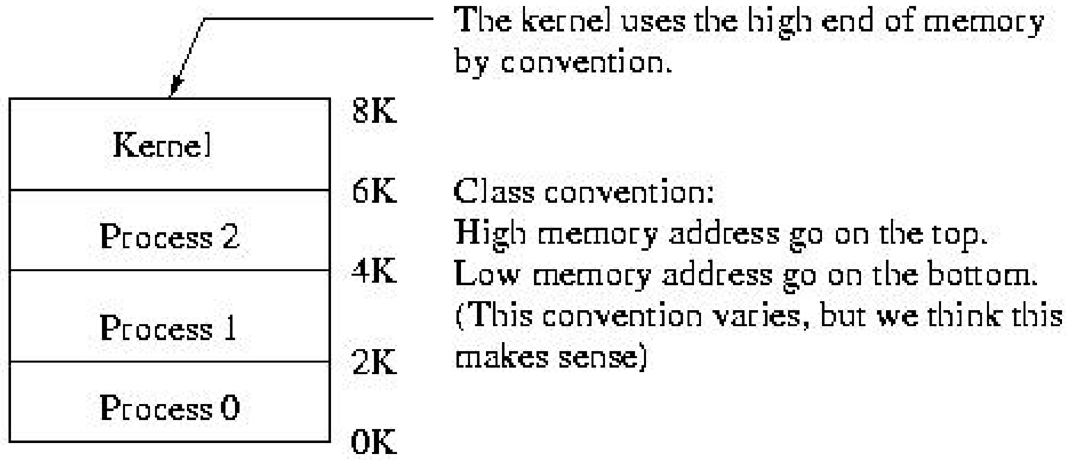


A multiprogramming system with three jobs in memory.

- Requires:
  - **secure memory and** I/O for each program.
  - **intervention, if program loops indefinitely**.
  - CPU scheduling to choose the next job to run.

Convey effect!

Simple Model for Memory Protection



To obtain true protection, need more hardware support.

- The goal is to make sure that, for example, process 2 can't go outside of the 4K-6K region.
  - This requires hardware support -- we can't do it in software alone.
  - The simplest model is to add two registers: BASE & LIMIT (assigned by OS)
  - This is the simplest model, most computers are more complex.

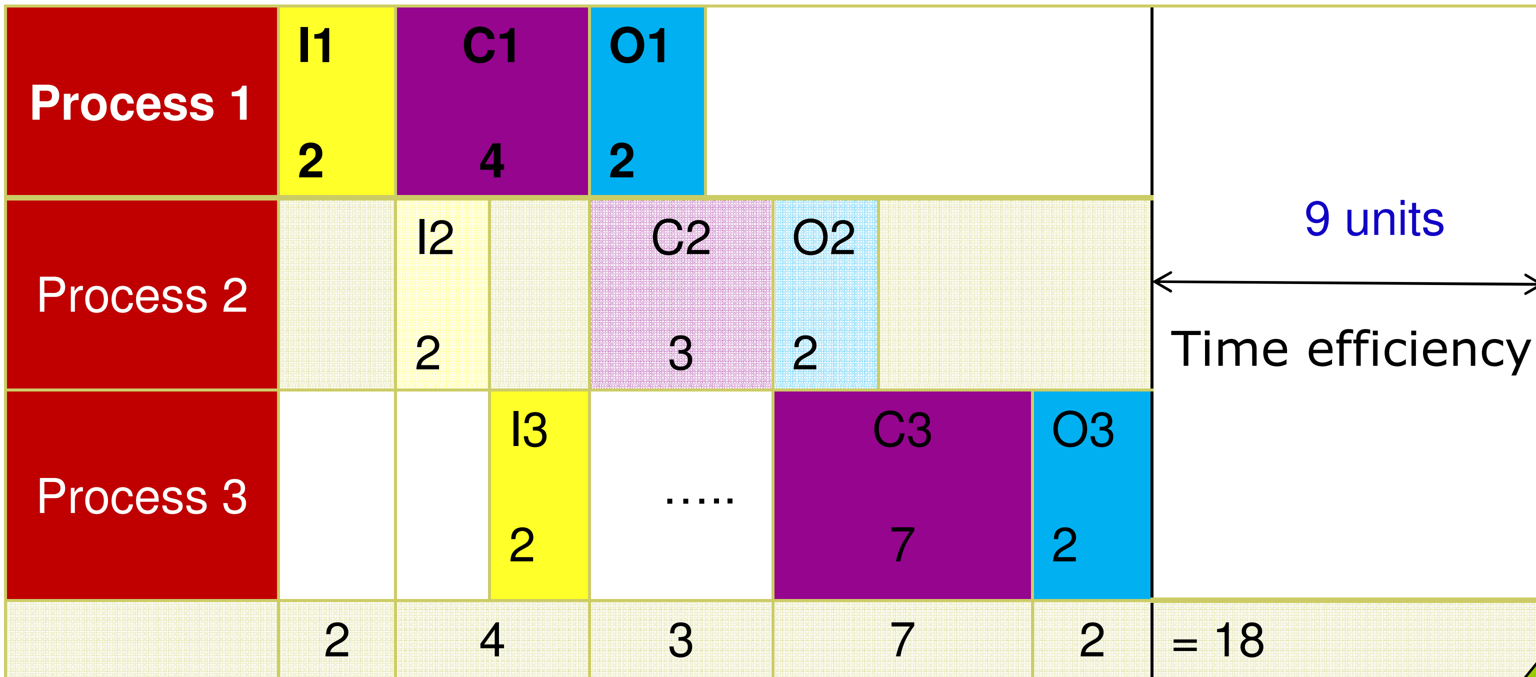
In order to keep process 2 within it's 2K box, we can do one of two things:

Option #1:
Set BASE = 4K
Set LIMIT=2K
A memory access is legal,
iff BASE <= X < BASE + LIMIT,
where X is the actual memory address

Option #2:
Set BASE = 4K
LIMIT = 2K
more frequent scheme
A memory access is legal,
iff 0 <= X < LIMIT,
where X is the actual memory address = BASE + X

Multiprogrammed Processing

I= Input
C=Computation
O=Output



Problem: Convey effect!

OS supports
- -SPOOL
- -Overlapping I/O and CPU
- - Interleaving – mixed of CPU and I/O operations and alternating between them in a program

Timesharing

- A variant of multiprogramming.
- Supports multiple users at the same time
- Programs queued for execution in FIFO order.
- A technique to protect CPU.

- **Like multiprogramming, but timer device interrupts** after a quantum (timeslice).
  - Interrupted program is returned to end of FIFO
  - Next program is taken from head of FIFO

Time Shared Processing

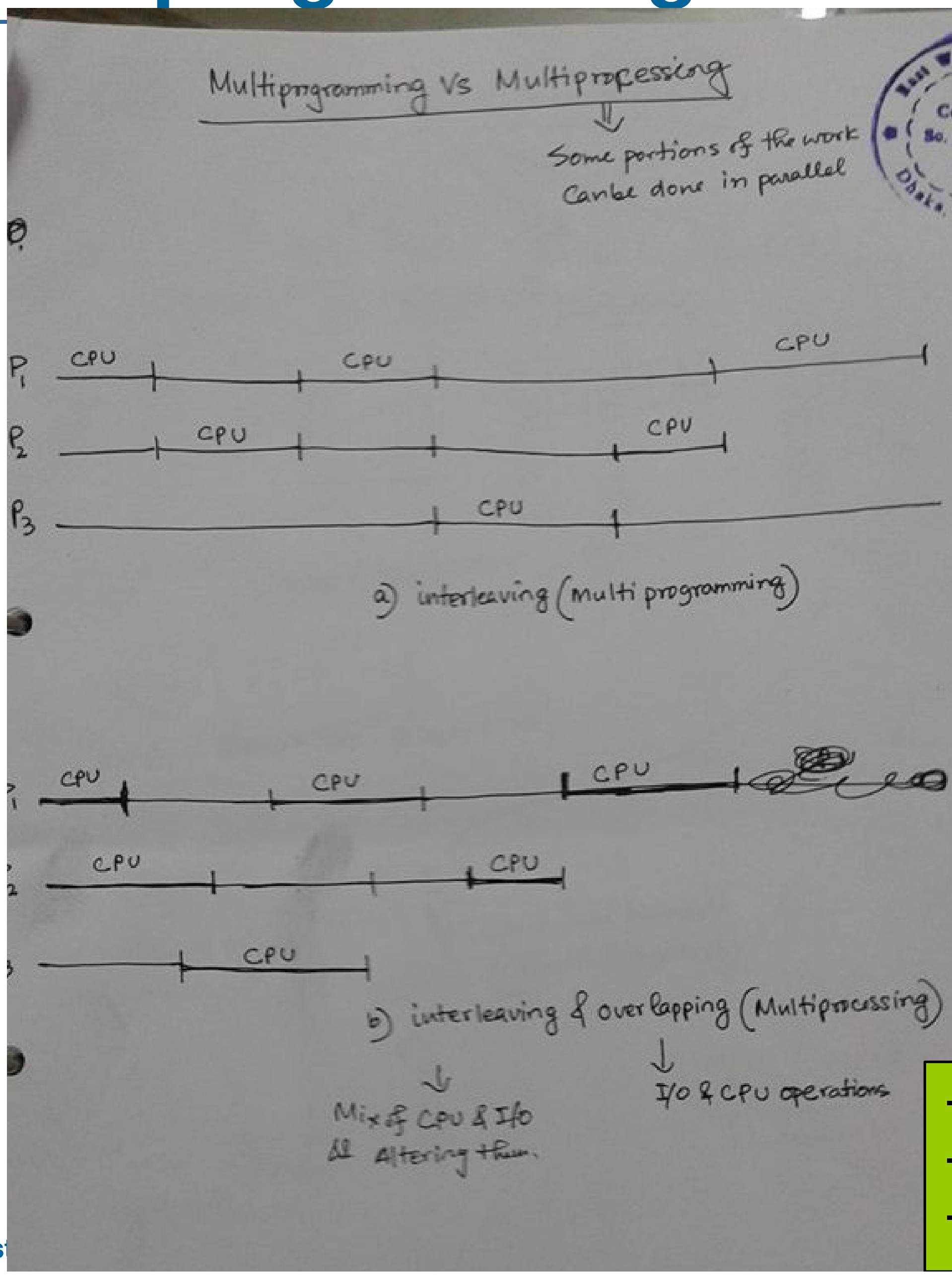
Process 1	**I1** 2	**C1** 2	…	….	**C1** 2	**O1** 2				
Process 2		I2 2	C2 2	…	…	C2 1 …	O2 2			
Process 3			I3 2	C3 2	…	… C3 2	C3 2	C3 1	O2 2	
	2	2	2	2	2	3	2	1	2	=18

Concept of Virtual Processors
Time Slice =2

Timesharing (cont.)

- Interactive (action/response)
  - when OS finishes execution of one command, it seeks the next control statement from user.
- File systems
  - online file system is required for users to access data and code.
- Virtual memory
  - Job is swapped in and out of memory to disk.

Multiprogramming vs Multiprocessing



- Interleaving CPU & I/O
- Overlapping CPU & I/O
- No overlapping CPU operations

- Interleaving CPU & I/O
- Overlapping CPU & I/O
- Overlapping CPU operations

Reading Materials

- Galvin : 1.1, 1.2, 2.1, 2.2, 2.5, 3.3, 3.5.3,
- Galvin: 13, 13.2.1,13.2.2,13.2.3
- Stallings: 1.1 – 1.4

End of Lecture 1-3