

CS 370: OPERATING SYSTEMS

[INTRODUCTION]

Shrideep Pallickara
Computer Science
Colorado State University

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

1

Frequently asked questions from the previous class survey

- Kernel and the OS: *Nota bene* the terms can be used interchangeably
 - Useful analogy: engine (kernel) and car (OS)
- Is the User Interface considered part of the OS?
- Where does the OS fit in the computer architecture?
- Masking hardware limitations?
- What is a Raspberry Pi?
- Readings in the course from different textbooks?
- Who creates the abstractions?
- Lots of technical terms: scheduling, virtualizing, resource allocations, memory management, etc.
- Echo360 recordings of Lectures?



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

2

Topics covered in this lecture

- CPU, Caches and main memory
- Secondary storage
- Relative speeds of the memory hierarchy
- The Kernel Abstraction



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

3

Over the past 50 years ...

- The most striking aspect has been Moore's Law and comparable advances in related technologies, such as memory and disk storage
- The cost of processing and memory has decreased by almost 10^6 over this period; the cost of disk capacity has decreased by 10^7
 - ▣ Disk latency has improved, but at a much slower rate than disk capacity
- These relative changes have **radically altered** both the use of computers and the **tradeoffs** faced by operating system designers



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

4

Operating systems tend to be huge, complex and long-lived

- Source code of an OS like Linux or Windows?
 - ▣ Order of 5 million lines of code (for kernel)
 - 50 lines page, 1000 pages/volume = 100 volumes
- Application programs such as GUI, libraries and application software?
 - ▣ 10-20 times that



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

5

Why do operating systems live for a long time?

- Hard to write and folks are loath to throw it out
- Typically **evolve** over long periods of time
 - ▣ Windows 95/98/Me is one OS
 - ▣ Windows NT/2000/XP/Vista/7/8/10/11 is another
 - ▣ System V, Solaris, BSD derived from original UNIX
 - ▣ Linux is a fresh code base
 - Closely modeled on UNIX and highly compatible with it
 - ▣ Apple OS X based on XNU (X is not Unix) which is based on the Mach microkernel and BSD's POSIX API



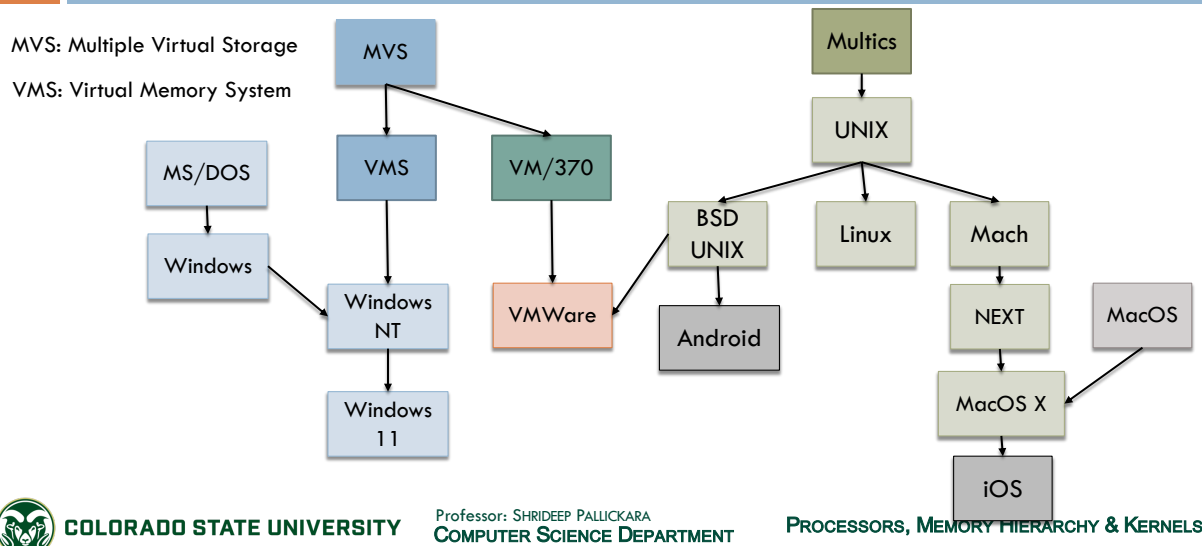
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

6

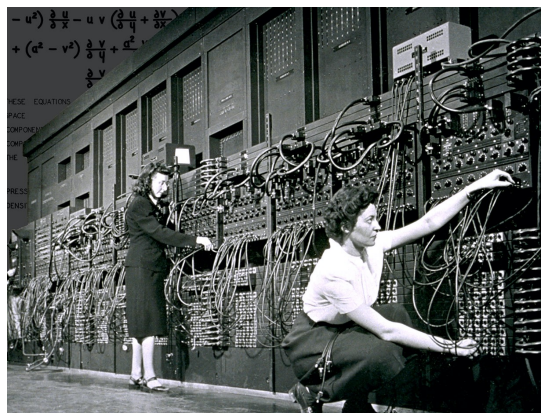
Genealogy of modern operating systems



7

Modern computer systems design

- Has its origins dating back to 1947 as part of work on ENIAC
- Breaks up a computing machine into three main subsystems:
 - ▣ The central processing unit (**CPU**) for performing arithmetic operations
 - ▣ The **memory** for storage of instructions and data
 - ▣ The input-output (**I/O**) unit for communicating with the external world
- This way of organizing a machine became known as the “**von Neumann architecture**”



8

Modern computers: The secret sauce?

- At its core modern computers harnesses the **movement of electrons** in circuits to carry out computations
- Computer circuits deal only with voltages, currents, switches, and malleable materials
 - ▣ Internally the computer does not process numbers and symbols



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

9

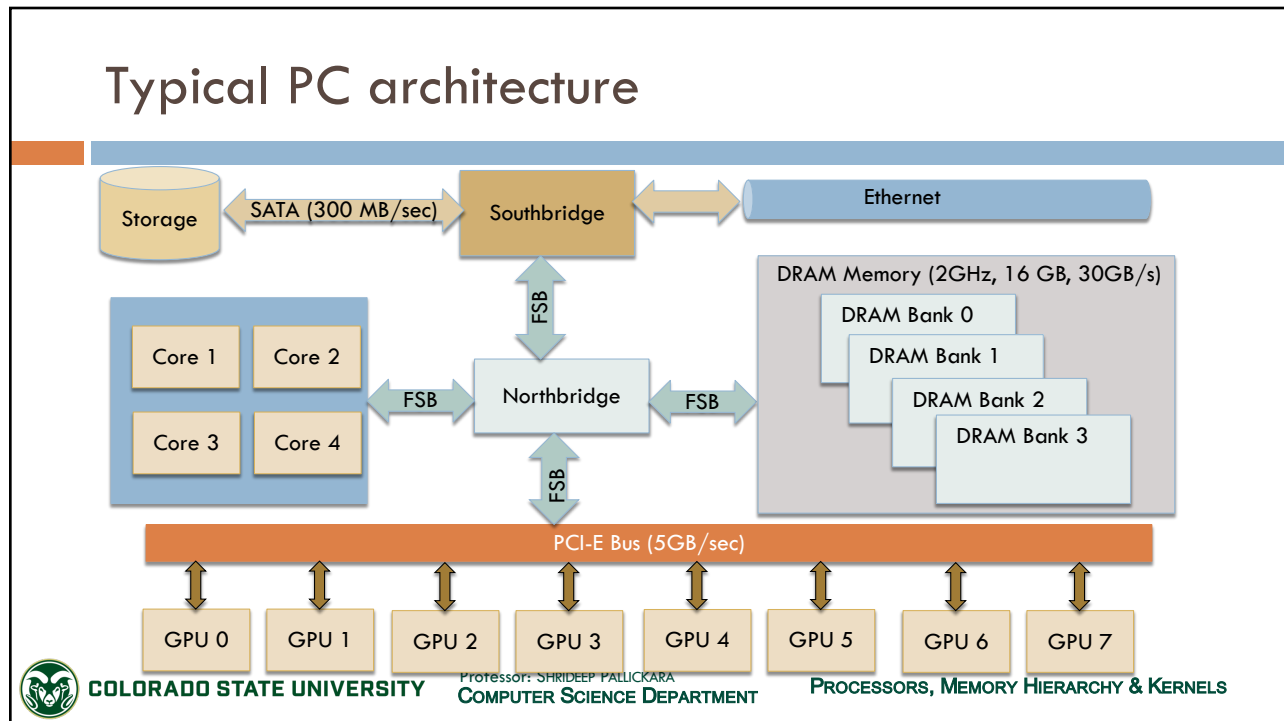
COMPONENTS OF A COMPUTER

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

10



11

Processors

- **Brain** of the computer
- Each CPU has a specific set of instructions that it can execute
 - Pentium cannot execute SPARC and vice versa

* The instruction set architecture (ISA) is how software interacts with the hardware.

COLORADO STATE UNIVERSITY
Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT
PROCESSORS, MEMORY HIERARCHY & KERNELS

12

Rationale for registers inside the CPU

- Accessing memory to get instruction or data
 - ▣ **Much longer** than executing the instruction
- Registers hold:
 - ▣ Key variables
 - ▣ Temporary results



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

13

What the instruction set looks like

- Load a word from memory into register
 - ▣ And, from register into memory
- Combine two operands from register, memory, or both into a result
 - ▣ E.g., add two words and store result in a register or in memory



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

14

Besides the registers to hold variable and temporary results there are special registers

- **Program Counter**
 - Contains the memory address of the next instruction to be fetched
- **Stack pointer**
 - Points to the top of the current stack in memory
- Program Status Word
 - Stores condition code bits and other control code bits
 - Plays an important role in system calls and I/O



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

15

Memory hierarchy:
Cache memory

- Mostly *controlled by hardware*
- Main memory divided up into **cache lines**
 - Usually 64 bytes
 - Addresses 0-63 in cache line 1, 64-127 in cache line 2, and so on
- Most heavily used cache lines are stored in a high-speed cache



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

16

When a program needs to read a memory word

- Cache hardware checks if the needed line is in the cache
- If it is, that's a **cache hit**
 - Request satisfied from cache in about *2 clock cycles*
 - No memory access needed
- If needed line is not present in cache
 - **Cache miss**, and must access memory
 - **Substantial** time penalty



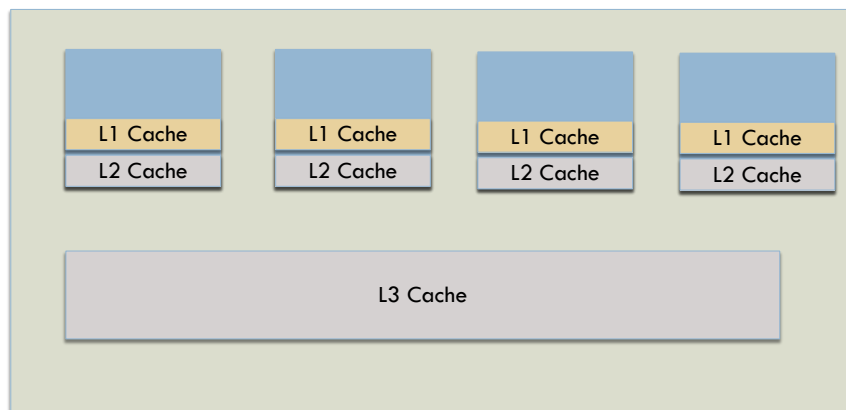
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

17

Logical view of the Processor, Cores, and Caches



L1: Typically in the order of 16 KB

L2, L3: Typically in the order of MBs

18

Caching is a powerful concept used elsewhere too. Let's see when ...

- ① Large resource *can be divided* into pieces
 - ② Some pieces *used more heavily* than others
- OS caching examples:
 - Pieces of heavily used files in main memory
 - Reduce disk accesses
 - Conversion of file names to disk addresses
 - Addresses of Web pages (URLs) as hosts



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

19

Main Memory

- Usually called **RAM** (Random Access Memory)
- Cache misses go to the main memory
- **Volatile**
 - Contents lost when power is turned off
- Memory size is in the order of several GB in most modern desktops



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

20

Memory

- Ideally the memory should be
 - Extremely **fast**: Faster than executing an instruction
 - CPU should not be held up by the memory
 - Abundantly **large**
 - Dirt **cheap**

- No current technology satisfies all these goals



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

21

Computers run most of their programs from (rewriteable) main memory

- Typically implemented in a technology called DRAM (dynamic random access memory)

- Ideal Scenario: Programs and data reside permanently in main memory. BUT ...
 - Space is *limited*
 - Main memory is *volatile* storage



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

22

Secondary storage is needed to hold large quantities of data permanently

- Programs use the disk as the source and destination of processing
- Seek time 7 ms
- SPIN: 7200 – 15000 RPM
- Transfer rate
 - ▣ Disk-to-buffer: 70 MB/sec (SATA)
 - ▣ Buffer-to-Computer: 300 MB/sec
- Mean time between failures
 - ▣ 600,000 hours
- 1 TB capacity for less than \$75



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

23

Improvements in hard disk capacity

- 1980 - 5 MB
- 1991 - 100 MB
- 1995 - 2 GB
- 1997 - 10 GB



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

24

Improvements in hard disk capacity

- 2002 - 128 GB addressing space barrier [28 bits]
 - Old IDE/ATA interface: 28-bit addressing
 - $2^{28} \times 512 = 2^{28} \times 2^9 = 2^{37} = 128 \text{ GB} = 137,438,953,472 \text{ bytes}$
- 2003 – Serial ATA introduced
 - Bus interface providing connections to mass storage devices
- 2005 – 500 GB hard drives
- 2008 – 1 TB hard drives



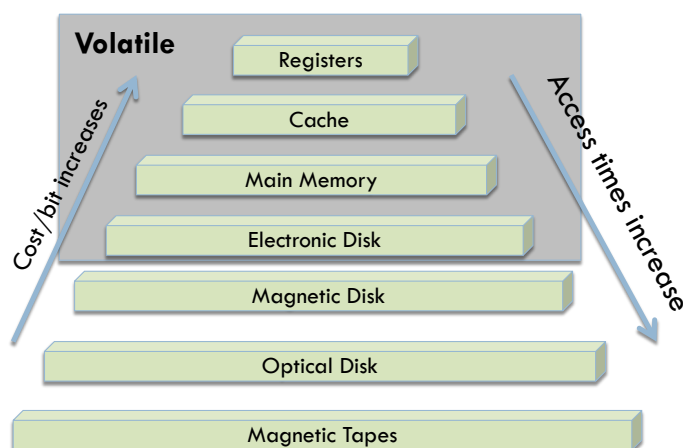
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

25

Storage system hierarchy based on speed, cost, size and volatility



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

26

Characteristics of peripheral devices & their speed relative to the CPU

Item	time	Scaled time in human terms (2 billion times slower)
Processor cycle	0.5 ns (2 GHz)	1 second
Cache access	1 ns (1 GHz)	2 seconds
Memory access	70 ns	140 seconds
Context switch	5,000 ns (5 μ s)	167 minutes
Disk access	7,000,000 ns (7 ms)	162 days
Quantum	100,000,000 ns (100 ms)	6.3 years



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

27

Mechanical nature of disks limits their performance

- Disk access times *have not* decreased exponentially
 - Processor speeds are growing *exponentially*
- Disparity between processor and disk access times continues to grow
 - 1:14,000,000



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

28

RELATIVE SPEEDS OF THE MEMORY HIERARCHY

COMPUTER SCIENCE DEPARTMENT



29

Since caches have limited size, cache management is critical

Level	1	2	3	4
Name	registers	cache	Main memory	Disk Storage
Typical Size	< 1 KB	< 16 MB	< 64 GB	> 100 GB
Implementation Technology	Custom memory, CMOS	On/off chip CMOS SRAM	CMOS DRAM	Magnetic disk
Access times	0.25 ns	0.5-25 ns	80-250 ns	> 5 ms
Bandwidth (MB/sec)	20,000 – 100,000	5000-10,000	1000-5000	80-300
Managed by	compiler	hardware	OS	OS
Backed by	cache	Main memory	Disk	CD/Tape



COLORADO STATE UNIVERSITY

OS & KERNELS

30

ONTOGENY RECAPITULATES PHYLOGENY

COMPUTER SCIENCE DEPARTMENT



31

After Charles Darwin's book ON THE ORIGIN OF SPECIES was published

□ German zoologist Ernst Haeckl stated

□ **Ontogeny** recapitulates **Phylogeny**

Development of an embryo

repeats

the evolution of the species

- i.e. human egg goes through stages of being a fish, ... , before becoming human baby
- Modern biologists think this is a gross simplification!



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

32

Something vaguely similar has happened in the computer industry

- Each new species (*type of computer*) goes through the development its ancestors did
 - ▣ Both in hardware and software
 - ▣ Mainframe, mini computers, PC, handheld, etc



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

33

Much of what happens in computing and other fields is technology driven

- Ancient Romans lacked cars not because they liked walking
 - ▣ It is because they didn't know to build cars
- PCs exist not because people have a centuries-old pent-up desire to own one
 - ▣ It is now possible to manufacture them cheaply



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

34

Technology affects our view of systems

- A **change** in technology renders some idea *obsolete*
 - Another change could *revive* it
- Especially true when change has to do with **relative performance**
 - Of different parts of the system



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

35

Let's look at this relative performance

- When CPUs become faster than memories?
 - Caches become important to speed-up slow memory
- If new memory technology makes memories much faster than CPUs?
 - Caches will vanish!
- In biology extinction is forever
 - In computer science, it is sometimes only for a few years



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

36

Historical developments

Large Memories

- IBM 7090/7094 1959-1964
 - 128 KB of memory
 - Programmed in assembly language (even the OS)
 - With time FORTRAN/COBOL and assembly was dead
- PDP-1 had only 4096 18-bit words of memory
 - Assembly is back!
 - Over time memory increases, assembly is out
- Microcomputers in 1980s
 - 4 KB memory and assembly is back again



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

37

Other places where such a cycle has gone on?

- Protection hardware
- Disks
- Virtual memory



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

38



Good fences make good neighbors
17th century proverb

THE KERNEL ABSTRACTION

COMPUTER SCIENCE DEPARTMENT



39

A central role of the OS is **protection** — the isolation of potentially misbehaving applications and users

- Implementing protection is the job of the OS **kernel**
- The kernel has full access to all of the machine hardware
 - ▣ The lowest level of software running on the system
 - ▣ Necessarily trusted to do anything with the hardware
- Everything other than the kernel — that is, the untrusted software running on the system — is run in a restricted environment
 - Less than complete access to the full power of the hardware



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

40

What hardware is needed to let the kernel provide isolation?

- At a minimum, the hardware must support **three** things:
- **Privileged Instructions:** All potentially unsafe instructions are prohibited when executing in user mode
- **Memory Protection:** All memory accesses outside of a process's valid memory region are prohibited when executing in user mode
- **Timer Interrupts:** Regardless of what the process does, the kernel must have a way to periodically regain control from the current process



Conceptually, the kernel/user mode is a one-bit register

- When set to **1**, the processor is in kernel mode and can do anything
- When set to **0**, the processor is in user mode and is restricted



Privileged Instructions

- Instructions available in kernel mode, but not in user mode, are called **privileged instructions**
- To do its work, the kernel must be able to execute these instructions
 - ▣ Change privilege levels, adjust memory access, and disable and enable interrupts, set/reset timers
- If these instructions were available to applications?
 - ▣ A rogue application would in effect have the power of the kernel



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

43

Making memory sharing safe

- The kernel must be able to configure the hardware so that each application process can read and write **only its own memory**
 - ▣ Not the memory of the operating system or any other application
- While it might seem that read-only access to memory is harmless, the OS needs to provide both security and privacy
 - ▣ For example, user passwords may be stored in kernel memory while they are being verified



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

44

Hardware timers

- Timers can be set to **interrupt** the processor after a specified delay
 - Either in time or after some number of instructions have been executed
- Each timer interrupts one processor ... separate timer for each CPU
 - The kernel might set each timer to expire every few milliseconds
- Resetting the timer is a privileged operation
 - User-level process cannot inadvertently or maliciously disable the timer
- How does the kernel know if an application is in an infinite loop?
 - It doesn't!



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

45

Mode transitions

- The kernel places a user process in a carefully constructed **sandbox**
 - The next question is how to safely transition from executing a user process to executing the kernel, and vice versa
- These transitions are **not rare events**
 - E.g.: A web server might switch between user mode and kernel mode thousands of times per second
- Transitions must be both fast and safe



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

46

There are three reasons for the kernel to take control from a user process

- Reasons: interrupts, processor exceptions, and system calls
- Asynchronous events
 - **Interrupts** are triggered by an *external event* and can cause a transfer to kernel mode after any user-mode instruction
- Synchronous events
 - **Processor exceptions** and **system calls** are triggered by *process execution*
 - The term **trap** refers to any synchronous transfer of control from user mode to the kernel



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

47

Interrupts are also used to inform the kernel of the completion of I/O requests

- Mouse device hardware triggers an interrupt every time the user moves or clicks on the mouse
 - The kernel, in turn, notifies the appropriate user process — the one the user was “mousing” across
- Virtually **every I/O device generates an interrupt** whenever some input arrives for the processor and whenever a request completes
 - E.g.: the Ethernet, WiFi, hard disk, thumb drive, keyboard, mouse, etc.



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

48

As the processor executes instructions, it checks for whether an interrupt has arrived

- If so, it completes or stalls any instructions that are in progress
 - ▣ Instead of fetching the next instruction, the processor hardware saves the current execution state
 - ▣ Starts executing at a specially designated interrupt handler in the kernel



Processor exceptions

- A processor exception is a **hardware event** caused by user program behavior that causes a transfer of control to the kernel
- A processor exception occurs whenever a process
 - ▣ Attempts to perform a privileged instruction
 - ▣ Accesses memory outside of its own memory region
 - ▣ Causes an arithmetic overflow. E.g., divide-by-zero
 - ▣ Accesses a word of memory with a non-aligned address
 - ▣ Attempts to write to read-only memory



User processes can also transition into the kernel voluntarily

- To request that the kernel perform an operation on the user's behalf
- A **system call** is any procedure provided by the kernel that can be called from the user level
 - Examples include system calls to establish a connection to a web server, to send or receive packets over the network, to create or delete files, to read or write data into files, and to create a new user process



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

51

To protect the kernel from misbehaving user programs

- It is key that the hardware transfers control on a system call to a pre-defined address
 - User processes cannot be allowed to jump to arbitrary places in the kernel
- The kernel handles the details of:
 - Checking and copying arguments
 - Performing the operation, and
 - Copying return values back into the process's memory



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

52

System calls provide the illusion that the kernel is simply a set of library routines available to users

- Implementing system calls requires the operating system to define a **calling convention**
- Once the arguments are in the correct format, the user-level program can issue a system call by executing the trap instruction to transfer control to the kernel
- The kernel implement its system calls in a way that **protects itself** from all errors and attacks that might be launched
 - Extreme version of defensive programming: *always* assume that system call parameters are intentionally designed to be as malicious as possible!



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

53

When an interrupt, processor exception or system call trap occurs ...

- How does the processor know what code to run?
- The processor has a special register that points to an area of kernel memory called the **interrupt vector table**
- The hardware determines which device caused the interrupt, if the trap instruction was executed, or what exception condition occurred
 - Thus, the hardware can select the right entry from the interrupt vector table and invoke the appropriate handler
- The format of the interrupt vector table is processor-specific



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

54

The interrupt vector table on the x86

- Entries 0 – 31: are for different types of processor exceptions
 - E.g: divide-by-zero anything related to arithmetic overflow
- Entries 32 – 255 are for different types of interrupts
 - Timer, keyboard, etc.
 - By convention, entry 64 points to the system call trap handler



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

55

What about kernel to user mode transitions? When do these happen?

- New process
- Resume after an interrupt, processor exception, or system call
- Switch to a different process
- User-level upcall
 - Most OS provide user programs with the ability to receive asynchronous notification of events



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

56



PERFORMANCE

COMPUTER SCIENCE DEPARTMENT




COLORADO STATE UNIVERSITY

57

There are two approaches to improving performance

- Determine component **bottlenecks**
 - Replicate
 - Improve



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

58

To replicate or improve?

"If one ox could not do the job, they [pioneers] did not grow a bigger ox, but used two oxen."

-- Admiral Grace Murray Hopper
Computer Software pioneer

"If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?"

-- Seymour Cray
Computer Hardware pioneer



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

59

Multiprocessor systems have 2-or-more processors in close communications

- The processors **share** the bus, and *may* share clock, memory and peripheral devices
- Advantages:
 - Increased throughput
 - Reliability



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

60

Multiprocessor systems fall in two categories based on control

- Asymmetric multiprocessing:
 - Controller processor manages the system
 - Workers **rely on controller** for instructions

- Symmetric multiprocessing
 - Processors are **peers** and perform all OS tasks
 - Have own set of registers and local cache
 - Share physical memory
 - **Supported by virtually all modern OS**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

61

Recent trend has been towards adding multiple cores

- *Raison d'être*
 - On chip communications are much faster
 - Uses less power than multiple single-core chips
 - Cope with heat dissipations
 - Improve Thread level parallelism

- Number of cores **doubling** every year
 - Each core also gets more execution pipelines
 - Gartner Projection: 1024 cores soon!

- Challenge: Re-engineering programs daunting



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS

62

The contents of this slide-set are based on the following references

- *Thomas Anderson and Michael Dahlin. Operating Systems: Principles and Practice, 2nd Edition. Recursive Books. ISBN: 0985673524/978-0985673529. [Chapter 2]*
- *Andrew S Tanenbaum and Herbert Bos. Modern Operating Systems. 4th Edition, 2014. Prentice Hall. ISBN: 013359162X/ 978-0133591620. [Chapter 1]*
- *Avi Silberschatz, Peter Galvin, Greg Gagne. Operating Systems Concepts, 9th edition. John Wiley & Sons, Inc. ISBN-13: 978-1118063330. [Chapter 1, 2]*
- *Kay Robbins & Steve Robbins. Unix Systems Programming, 2nd edition, Prentice Hall ISBN-13: 978-0-13-042411-2. [Chapter 1]*
- *Peter J. Denning and Matti Tedre. Computational Thinking. MIT Press. ISBN-13: 978-0262536561. [Chapter 3]*



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

PROCESSORS, MEMORY HIERARCHY & KERNELS