

Lecture #17 – Operating Systems (OS)

**ESE 150 –
DIGITAL AUDIO BASICS**

ESE150 Spring 2021

Based on slides © 2009–2021 DeHon

ESE150 Spring 2021

MOTIVATION

- × **What things can your phone do while you are listening to an MP3?**

2

ESE150 Spring 2021

OBSERVATION

- × **We want our devices (including our phones) to do many things at once.**

3

ESE150 Spring 2021

MULTIPLE TASKS

- × **We could...**
 - + Dedicate a separate processor for every task we want to perform
- × **How many would we need?**
- × **Maybe**
 - + Need dozen processors for our Phone

4

ESE150 Spring 2021

BUT....

- × **MP3 Play**
 - + 44,000 samples per second decoded
 - + 500 cycles to decode a sample
 - + **How many instructions per second require?**
- × **What fraction of a 10^9 instruction per second processor does this use?**

5

ESE150 Spring 2021

OBSERVATION

- × **If we dedicate a processor to MP3 decoding**
 - + It will sit idle most of the time
- × **MP3 decoding (and many other things) do not consume a modern processor**
- × **Idea: Maybe we can share the processor among tasks?**

6

ESE150 Spring 2021

OUTLINE

- × Setup Need / Opportunity
- × Where are we
- × Role of Operating System
- × Virtualization

7

ESE150

COURSE MAP - WEEK 10

Music 1

Numbers correspond to course weeks

sample 2

freq 4

domain conversion 5,6

psycho-acoustics 3

compress

EULA

click OK

speaker

D/A

10101001101

MP3 Player / iPhone / Droid

8

ESE150 Spring 2021

"STORED-PROGRAM" PROCESSOR

- × By filling in memory, can program to perform any computation

9

ESE150 Spring 2021

ROLE OF OPERATING SYSTEM

10

ESE150 Spring 2021

PROGRAMMING THE PROCESSOR

- × **Think about:** How do we change the program in the memory
- × What if had to reboot machine... for every application?
 - + Change flash card
 - + Download over USB
 - × ...that is what we have to do for the Arduino...

11

ESE150 Spring 2021

MORE THAN ONE PROGRAM?

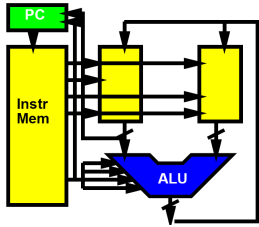
- × **How could we have multiple applications?**
 - + (just run one at a time for now)

12

ESE150 Spring 2021

COORDINATION?

- ✗ Does every program need to know about every other program?
- ✗ IoT device that runs 3 unchanging tasks?
- ✗ Smart phone that allows 3rd party apps
 - + Think: AppStore, Google Play



The diagram shows a central PC connected to a block of Instruction Memory (Instr Mem). Below this are two processors, each with its own local memory and connected to a shared ALU. Arrows indicate data flow between the PC, memory, processors, and the ALU.

13

ESE150 Spring 2021

ROLE OF OPERATING SYSTEM

- ✗ Higher-level, shared support for all programs
 - + Could put it in program, but most programs need it!
 - + Needs to be abstracted from program
- ✗ Resource sharing
 - + Processor, memory, "devices" (net, printer, audio)
- ✗ Polite sharing
 - + Isolation and protection
 - + *Fences make Good Neighbors* – R. Frost
- ✗ Idea: Expensive/limited resources can be shared in time – OS manages this sharing

14

ESE150 Spring 2021

VIRTUALIZATION

15

ESE150 Spring 2021

VIRTUALIZATION

- ✗ Providing an abstract view separate from the physical view
- ✗ Hides physical view
- ✗ Provides abstract view to software
 - + Abstract from physical resource limits

16

ESE150 Spring 2021

IDEA

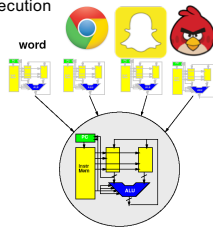
- ✗ Virtualize the processor
 - + Make it look like we have multiple processors
 - + With each program running on its own processor
- ✗ "Own" processor
 - + Can put data in memory where it wants
 - + Doesn't have to worry about another program scribbling over its memory
 - + Its state is preserved and isolated
 - + Looks like it runs all the time on the processor
 - Doesn't need to be programmed to allow other programs to run

17

ESE150 Spring 2021

TERMINOLOGY: PROCESS

- ✗ Process
 - + A virtualization of the physical processor
 - ✗ an instance of a program in execution
 - + Virtual processor



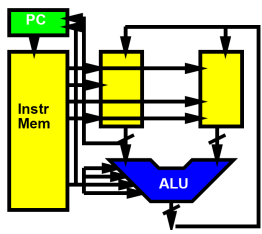
The diagram shows three application icons (Google, Snapchat, Angry Birds) with arrows pointing to a central processor diagram, illustrating how these applications are virtualized onto a single physical processor.

18

ESE150 Spring 2021

WHAT DOES OUR PROGRAM SEE?

- × **Physically**
 - + One processor
 - × One PC
 - × One data memory
 - × One instruction memory
 - + These are its **state**
 - × Terminology: context

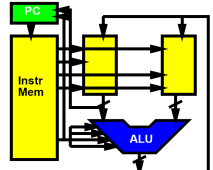


19

ESE150 Spring 2021

EXECUTING THE PROGRAM

- × **To execute program**
 - + Keep track of state of machine
 - × Value of counter (Program counter)
 - × Contents of instruction memory
 - × Contents of data memory

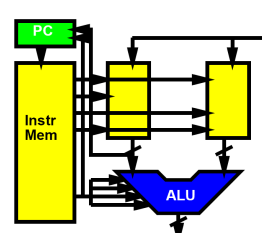


20

ESE150 Spring 2021

ONE PROCESSOR, ONE PROGRAM

- × **On the physical machine, can only run one program**
 - + **Why?**
 - × One PC
 - × One memory



21

ESE150 Spring 2021

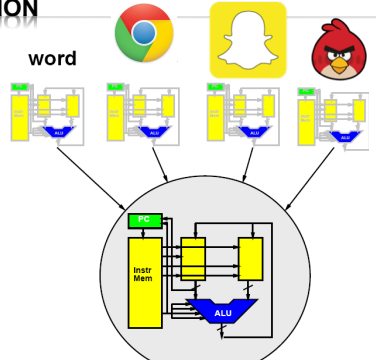
VIRTUALIZATION

- × **Make it look like we have multiple resources**
 - + Multiple processors
- × **Provide abstraction of large* number of processors**
 - + Each program gets its own processor
 - × Each program gets its own machine state
 - + * "large" enough to approximate infinite

22

ESE150 Spring 2021

VIRTUALIZATION



23

ESE150 Spring 2021

KEY IDEA

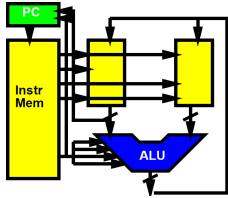
- × **Can capture state of a processor**
 - + All the information that defines the current point in the computation

24

ESE150 Spring 2021

REMEMBER

- × **State of the processor**
 - + Value of Program Counter (PC)
 - + Contents of instruction memory
 - + Contents of data memory



25

ESE150 Spring 2021

KEY IDEA

- × **Can capture state of a processor**
 - + All the information that defines the current point in the computation
 - + i.e. program counter, data and instruction memory
- × **Can save that in somewhere***
- × **Fully represents the running program**
- × **Can restore that from <where-saved> to the processor**
- × **Can save/restore without affecting the functional behavior of the program**

26

ESE150 Spring 2021

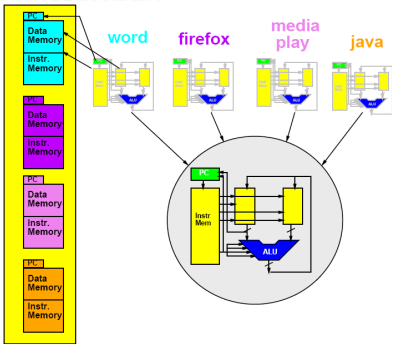
SOMEWHERE? -- MEMORIES

- × **Distinguish**
 - + Memory-seen-by-process (virtualized processor)
 - + All Memory used
 - × Physical memory available
 - × Holds state of *all* processes
- × **How might we divide up physical memory among processes?**
 - + How much each get?
 - + How define what memory goes to which process?

27

ESE150 Spring 2021

STATE IN MEMORY

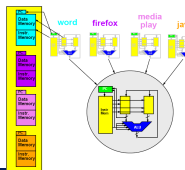


28

ESE150 Spring 2021

SHARING PROCESSOR

- × **Now that we can save/restore the state**
- × **Can share processor among processes**
 - + (Restore state; run for time; save state)
- × **Isolation: none of the processes need to know about each other**
 - + Each thinks it has the whole machine
 - + Just need to restore/save state around epochs where the process gets to run on the processor



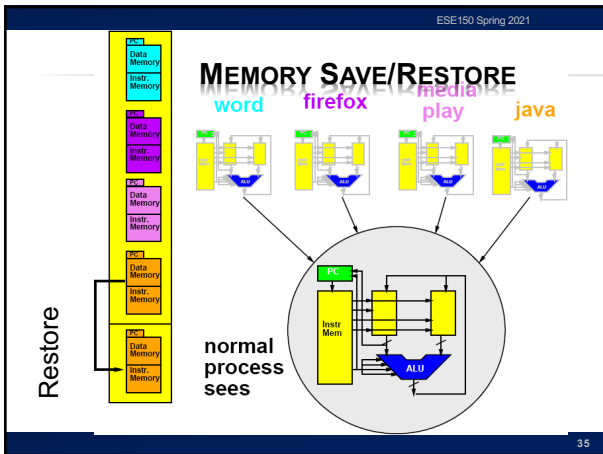
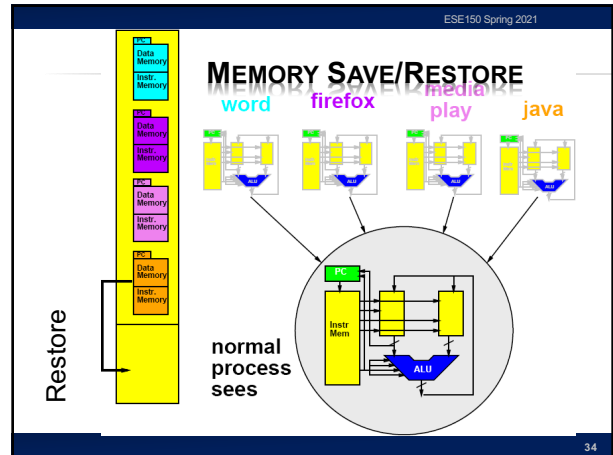
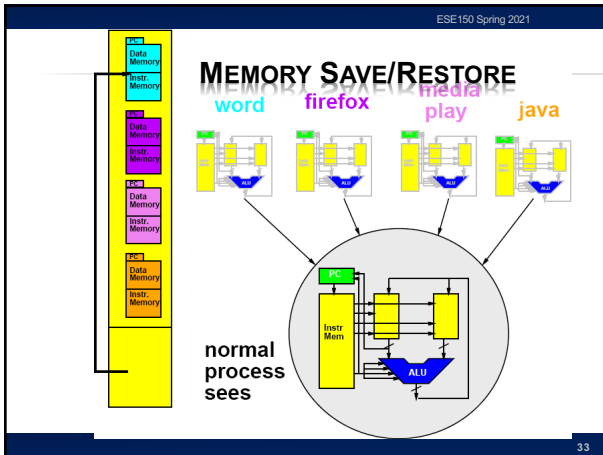
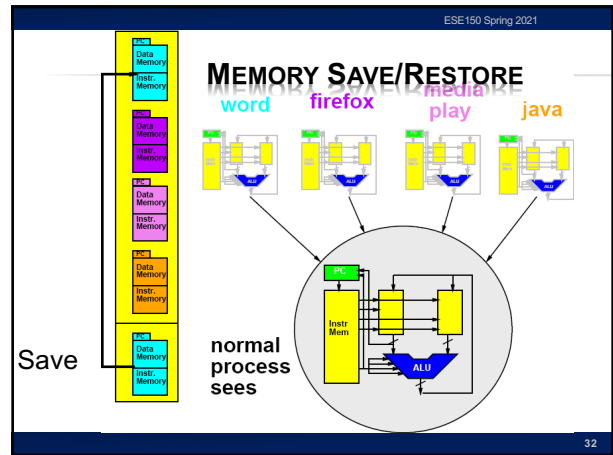
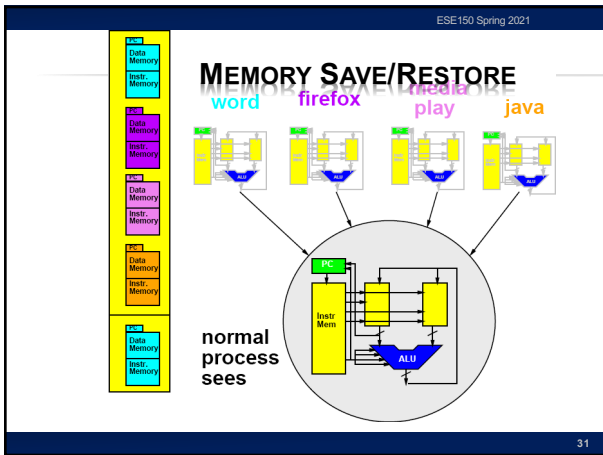
29

ESE150 Spring 2021

SAVING MEMORY?

- × **Each program has view that it owns machine**
 - + Each may put program in same place?
 - + Shouldn't have to know about other programs, where their stacks are...
- × **Could:**
 - + Have programs operate 0...max_process_mem
 - + Copy data in and out of this range
 - + Keep in larger physical memory
 - × not visible to program (process)

30



- ESE150 Spring 2021
- ### SAVING MEMORY?
- × **Each program has view of its own machine**
 - + Each may put program in same place
 - + Shouldn't have to know about other programs...
 - × where their stacks are...etc.
 - × **Can do better**
 - + Assume physical memory is larger than process memory
 - + **How could we avoid copying?**
 - + Virtualizing Memory as well
 - × Translate processor addresses
- 36

ESE150 Spring 2021

PROCESS BASE OFFSET REGISTER

- × **Add Offset Register**
 - + Holds base of memory space for process
- × **Add this to all memory references**
- × **Change memory seen by process by changing offset register**

37

ESE150 Spring 2021

MANAGEMENT PROGRAM

- × **Need another program → process**
 - + Manage swap of running processes
 - + Decide what to run next
 - + Decide when to stop a process
- × **...process manager/scheduler**

38

ESE150 Spring 2021

TIME-SLICED SHARING

- × **Simplest version:**
 - + Run each process for 10,000 cycles
 - + Then swap to next process
 - + Looks like each of n process runs on a processor $1/n$ -th the speed of the real processor
- × **More sophisticated:**
 - + Assign uneven time to processes
 - + Also change when process...
 - × waits for input
 - + **What are cases where**
 - × Uneven time appropriate?
 - × Valuable to switch on input?

39

ESE150 Spring 2021

REVIEW: KEY IDEA

- × **Can capture state of a processor**
 - + All the information that defines the current point in the computation
 - + i.e. program counter, data and instruction memory...
- × **Can save that in memory**
 - + A different memory from what the process sees
 - + (could be different range of addresses)
- × **Fully represents the running program**
- × **Can restore that from memory to the processor**
- × **Can save/restore without affecting the functional behavior of the program**

40

ESE150 Spring 2021

UPCOMING LAB

- × **Explore Linux OS and processes on Linux**
 - + See processes sharing processors
 - + Lab available now
 - × Some work (possibly installation) on prelab

41

ESE150 Spring 2021

BIG IDEAS

- × **Virtualize hardware**
 - + Identify state; save/restore from memory
- × **Program view: owns complete machine**
- × **Allows programs to share limited physical hardware (e.g. processor)**
 - + Provide illusion of unlimited hardware
- × **Operating System is the program that manages this sharing**

42

LEARN MORE

- × **CIS380 – Operating Systems**

REMEMBER

- × **Feedback**
- × **Lab 8 Due today**
- × **Lab 9 on Monday**