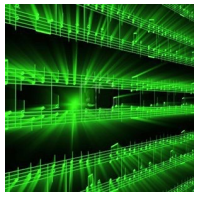
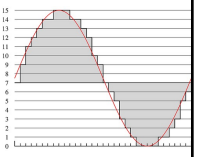
Penn Engineering ESE





Lecture #17 – Operating Systems (OS)

ESE 1500 – Digital Audio Basics

ESE1500 Spring 2023

1

Motivation

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

2

Observation

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

3

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

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

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?**

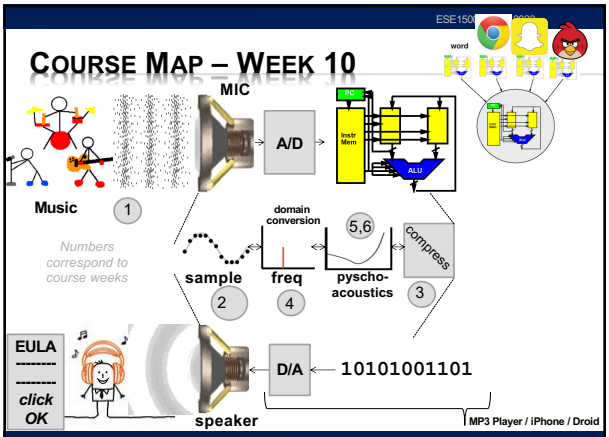
6

OUTLINE

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

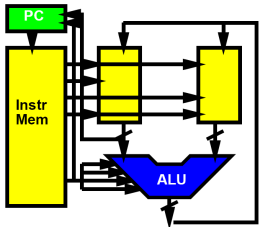
7

COURSE MAP – WEEK 10



8

"STORED-PROGRAM" PROCESSOR



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

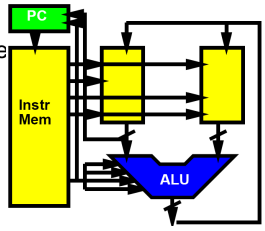
9

ROLE OF OPERATING SYSTEM

10

PROGRAMMING THE PROCESSOR

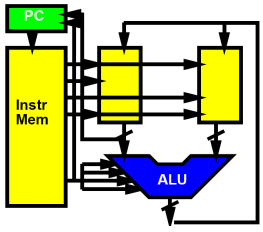
- **Think about:** How do we change the program in the memory
- What if had to reboot machine… for every application?
  - Change flash card
  - Download over USB
    - …that is what we have to do for the Arduino…



11

MORE THAN ONE PROGRAM?

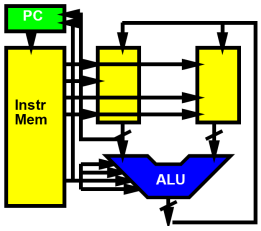
- **How could we have multiple applications?**
  - (just run one at a time for now)



12

More than one Program Actively Running?

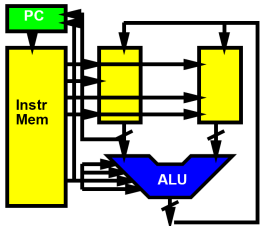
- **How could we run multiple applications concurrently?**
  - Start some programs/apps before earlier ones end
  - Have all started applications able to respond and make progress



13

More than one Program Actively Running

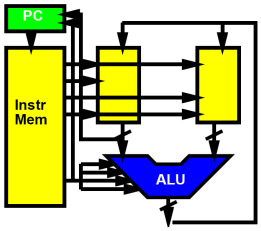
- **Idea:** share processor in time
- Programs take turns running instructions on processor
- Break program into pieces
  - Run piece of program then change to next
- **How could we define pieces?**



14

Coordination?

- **Does every program need to know about every other program?**
- **Is that viable?**
- **IoT device that runs 3 unchanging tasks?**
- **Smart phone that allows 3rd party apps**
  - Think: AppStore, Google Play
    - Millions of apps available



15

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

16

Virtualization

17

Virtualization

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

18

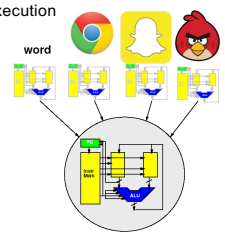
IDEA

- **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

19

TERMINOLOGY: PROCESS

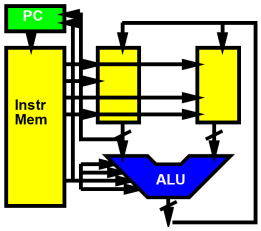
- **Process**
  - A virtualization of the physical processor
    - an instance of a program in execution
  - Virtual processor



20

WHAT DOES OUR PROGRAM SEE?

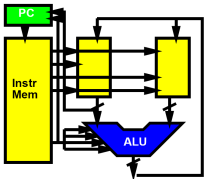
- **Physically**
  - One processor
    - One PC
    - One data memory
    - One instruction memory
  - These are its **state**
    - Terminology: context



21

EXECUTING THE PROGRAM

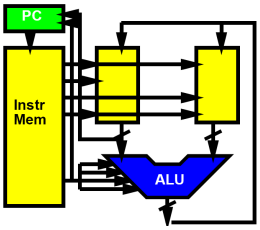
- **To execute program**
  - Keep track of state of machine
    - Value of counter (Program counter)
    - Contents of instruction memory
    - Contents of data memory



22

ONE PROCESSOR, ONE PROGRAM

- **On the physical machine, can only run one program**
  - One PC
  - One memory
- **Why limit when want to run multiple programs?**



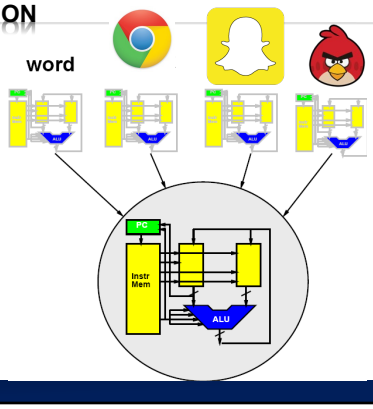
23

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

24

VIRTUALIZATION



25

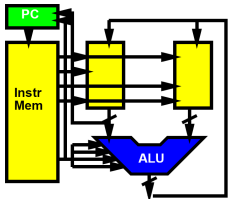
KEY IDEA

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

26

REMEMBER

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



27

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 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**

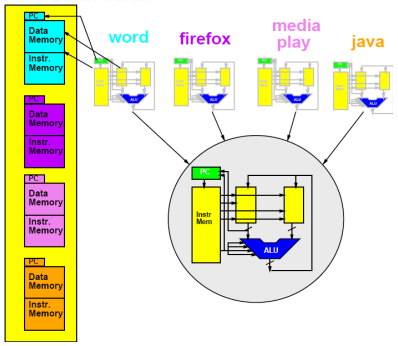
28

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?

29

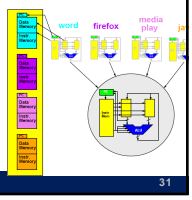
STATE IN MEMORY



30

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



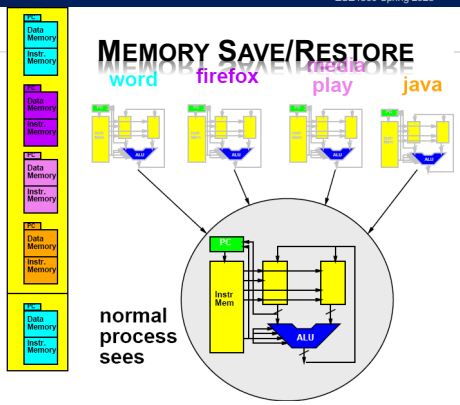
31

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)

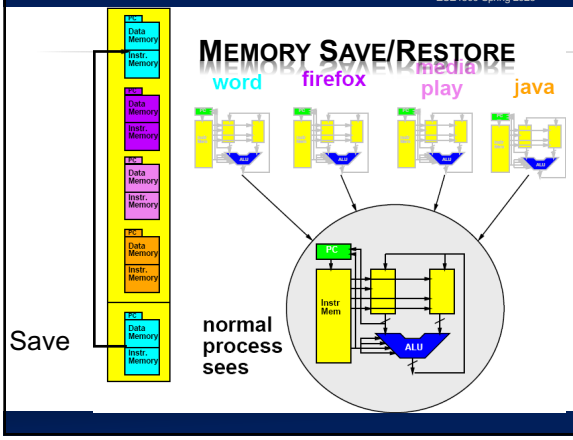
32

Memory Save/Restore



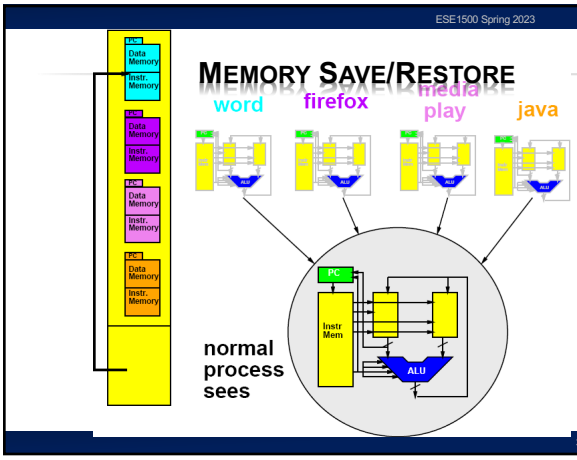
33

Memory Save/Restore



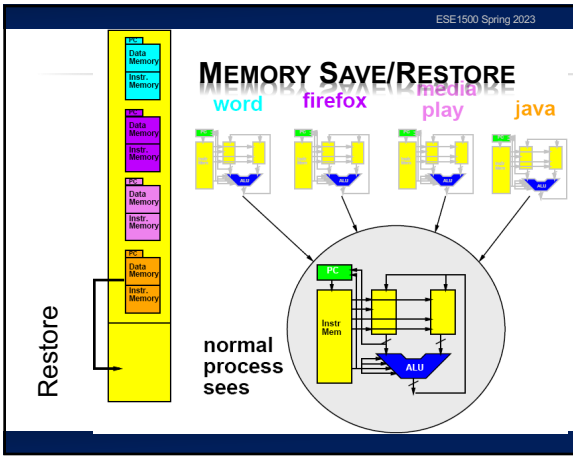
34

Memory Save/Restore



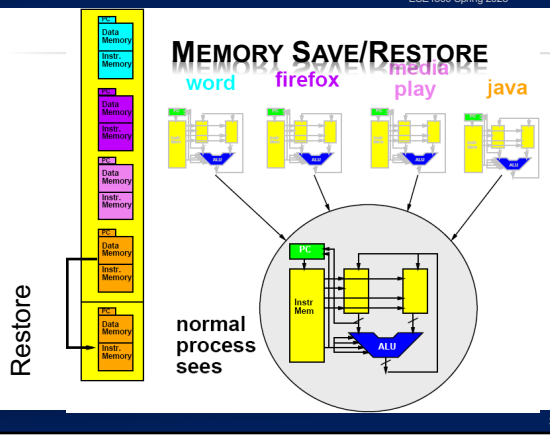
35

Memory Save/Restore



36

Memory Save/Restore



37

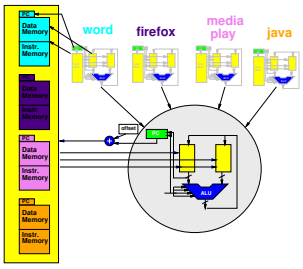
Saving Memory?

- **Each program has view of it owns 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

38

Process Base Offset Register

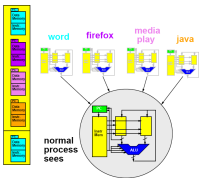
- **Add Offset Register**
  - Holds base of memory space for process
- **Add this to all memory references**
- **Change memory seen by process by changing offset register**



39

Management Program

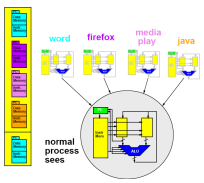
- **Need another program → process**
  - Manage swap of running processes
  - Decide what to run next
  - Decide when to stop a process
- **…process manager/scheduler**



40

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?



41

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)
- **State fully represents the running program**
- **Can restore that from memory to the processor**
- **Can save/restore without affecting the functional behavior of the program**

42

LAB 9

- **Explore Linux OS and processes on Linux**
  - See processes sharing processors
  - Lab available now

43

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**

44

LEARN MORE

- **CIS3800 – Operating Systems**

45

REMEMBER

- **Feedback, including lab**
- **Lab 9 on Monday**

46