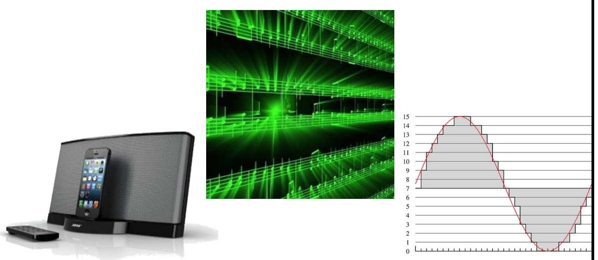
Penn Engineering

ESE



Lecture #18 – Operating Systems (OS)

ESE 150 – Digital Audio Basics

ESE150 Spring 2022

Based on slides © 2009--2022 DeHon

1

Observation

- **We want our phones (and computers) to do many things at once.**
- **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?**

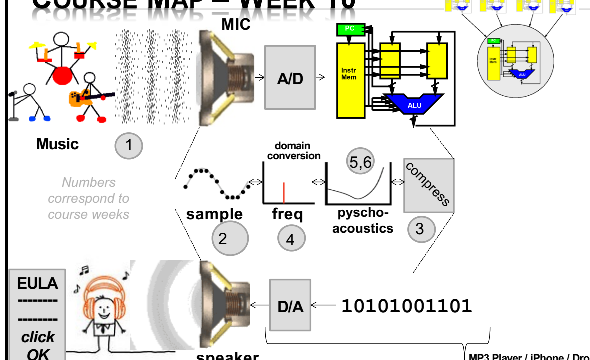
2

Outline

- **Review**
- **Worksheet: Virtualization In Action**

3

Course Map – Week 10



4

Virtualization

5

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

6

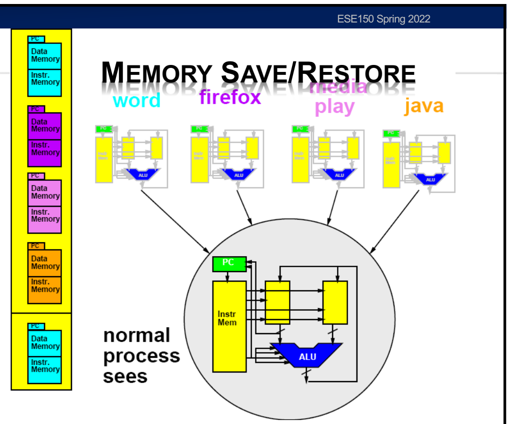
Strategy

- Time-share single processor in time
  - Store all process state in memory
    - Process = virtual processor
  - Iterate through processes
    - Restore process state
    - Run for a number of cycles
    - Save process state

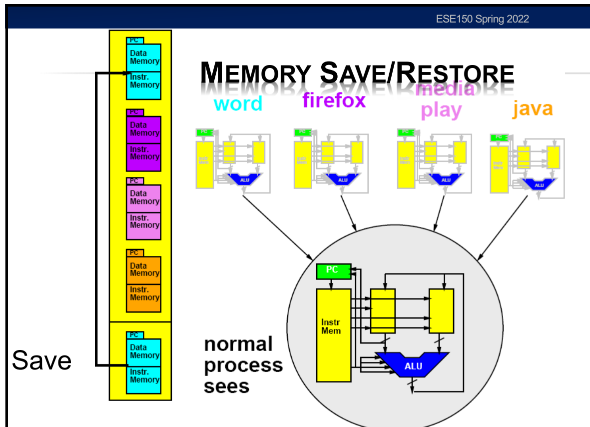
Idea Refined

- 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

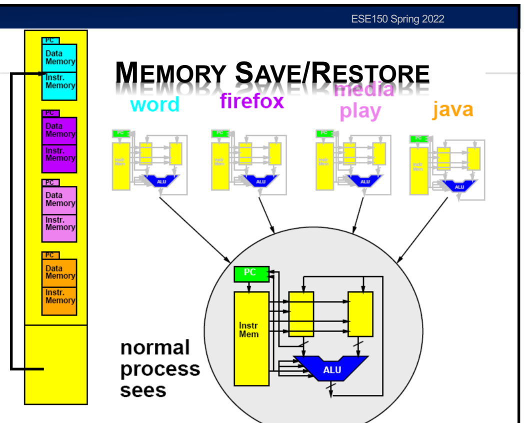
Memory Save/Restore



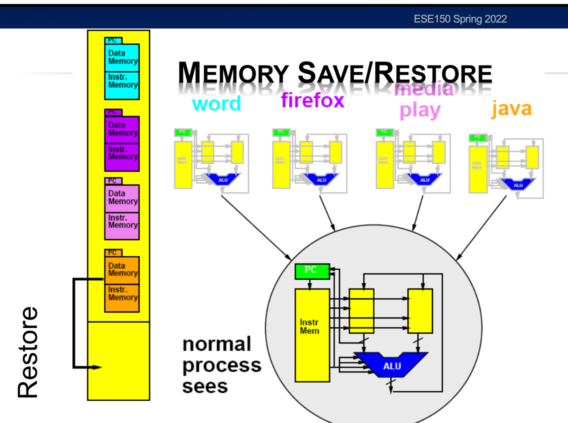
Memory Save/Restore



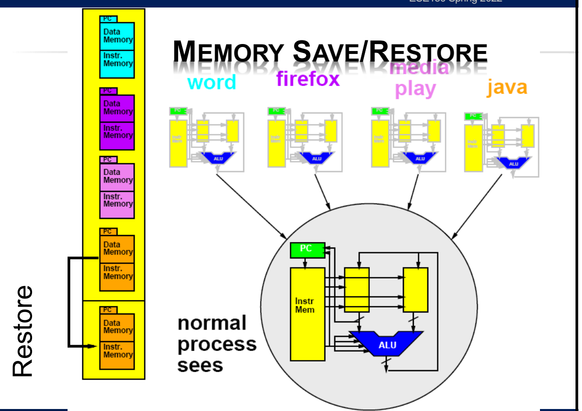
Memory Save/Restore



Memory Save/Restore



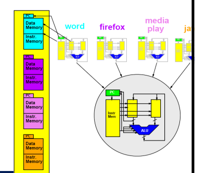
Memory Save/Restore



13

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



14

Worksheet Exercise

Demonstration

15

Worksheet: Execution Exercise

- We're going to simulate the computer and watch the processor state

16

Execution Exercise

- Simulate A for 12 cycles
  - Work together as class

18

Simulate Swapping

- Imagine we ran A for 6 cycles (and saved state)
- Swap and Run B for 6 cycles
- Swap and Run A for next 6 cycles
  - What should we get?
- Swap and Run B for next 6 cycles
- Swap and Run A for next 6 cycles (time permit)

19

Simulate Swapping

- Simulate B for 6 cycles
  - Individually
- What get for +6 line?

Simulate Swapping

- Swap in A+6 and Simulate for 6 cycles (to 12)
  - individually
- What get for +12 line?
- Compare to what we got on A simulation?

Simulate Swapping

- Swap in B+6 and Simulate for 6 cycles (to 12)
  - individually
- What get on +12 line?

Simulate Swapping (time permitting)

- Swap in A+6 and Simulate for 6 cycles (to 18)
  - individually
- What get on +18 line?

Simulate Swapping

- Imagine we ran A for 6 cycles (and saved state)
- Swap and Run B for 6 cycles
- Swap and Run A for next 6 cycles
- Swap and Run B for next 6 cycles
- Swap and Run A for next 6 cycles (time permit)

Conclude

- Can Time Share Processor
- Key is saving/restoring state of computation
- Interleave computation of all the processes

Review: Key Idea

- Can capture state of a processor
  - All the information that defines the current point in the computation (PC, data and instruction mem)
- 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
- Time-share processor → pretend have unlimited number

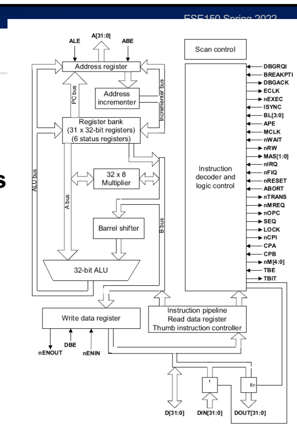
26

Media Processors

27

iPod Processor

- Early based on PortalPlayer series
  - Two ARM7TDMI cores
  - 80MHz each
- Guesses are ARM7 or ARM8



28

Apple A14 Bionic

- 88mm^2, 5nm
- 11.8 Billion Tr.
- iPhone 12
- 6 ARM cores
  - 2 fast (2.9--3GHz)
  - 4 low energy
- 4 custom GPUs
- 16 Neural Engines
  - 11 Trillion ops/s?

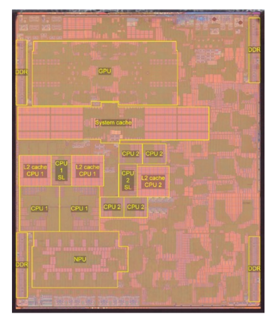


Image from https://www.extremetech.com/computing/318715-comparison-of-apple-m1-a14-shows-differences-in-s
details: https://www.tomshardware.com/news/apple-a14-bionic-revealed
https://www.anandtech.com/show/16226/apple-silicon-m1-a14-deep-dive/2

29

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

30

Learn More

- CIS380 – Operating Systems

31

Reminders

- Feedback including Lab
- Lab8 writeup due today
- Lab9 on Wednesday