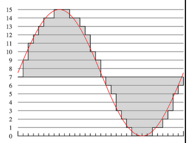
Penn Engineering ESE



Lecture #9 – Operating Systems (OS)

ESE 150 – Digital Audio Basics

Based on slides © 2009--2018 DeHon
Additional Material © 2014 Farmer

Preclass 1

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

Observation

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

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

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?

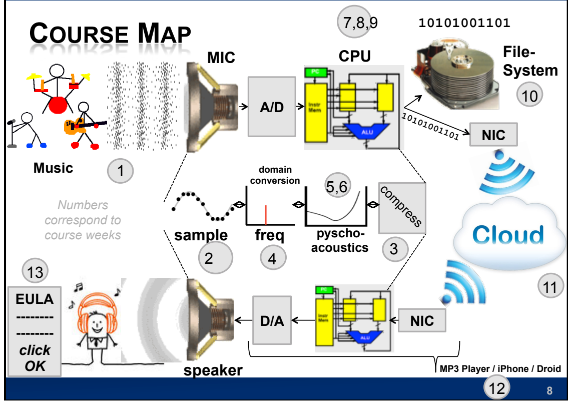
Observation

- 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 the processor
- Maybe we can share the processor among tasks?

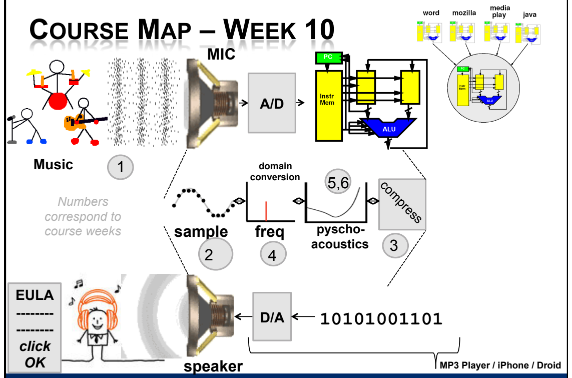
Outline

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

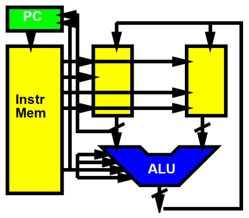
Course Map



Course Map – Week 10



"Stored-Program" Processor

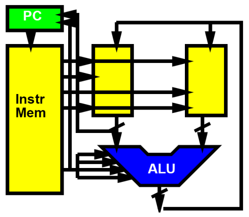


- By filling in memory, can program to perform any computation

Role of Operating System

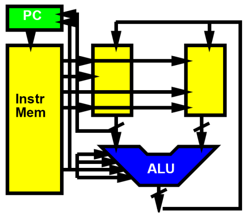
Programming the Processor

- How can we change the program/app?
  - How do we gets the bits into memory?
- What if had to reboot machine (change flash card) for every application?



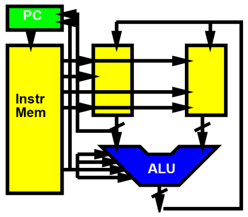
More than one Program?

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



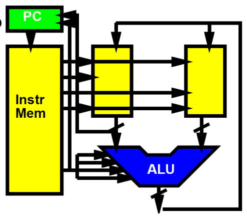
Multiple Running Programs?

- How can multiple applications run simultaneously on this?



Coordination?

- Does every program need to know about every other program?
  - Implications?
- Where acceptable?
  - Proprietary system with small set of applications all developed in-house.
- Where unworkable?
  - Any upgradeable platform (e.g. laptop, iPhone)
  - Any platform integrating non-source applications from variety of sources



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
- **Idea: Expensive/limited resources can be shared in time – OS manages this sharing**

Shared Support

- **What software support do most programs need?**
- **Examples:**
  - Memory allocation/deallocation
  - Handle I/O: keyboard/screen
  - Draw pretty boxes/menus/selections

Devices

- **Displays**
- **Input (keyboard, mouse)**
- **Storage (hard drive, USB drive, CDROM)**
- **Network (ethernet, wifi, bluetooth)**
- **Microphone, speakers**
- **GPS**
- **Printer**

Device Coordination

- Coordinate among multiple users
  - Don't want programs accessing hardware directly ignorant of other users
- Exclusively allocate to one application at a time?
  - Speaker
  - Printer
  - Screen? (portion of screen?)
- Allow interleaved use?
  - Network
  - Hard disk

19

Virtualization

20

Virtualization

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

21

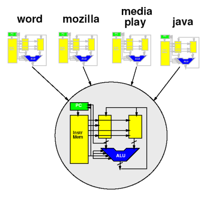
Idea

- Virtualize the processor
  - Make it look like we have multiple processors
  - With each program running on its own processor
- Abstraction
  - Programs see hardware as simple blocks
    - Ex: USB/Display/I/O all seen as a "file"
    - Programmer View:
      - calls function: "FGETC()" to read character from keyboard
    - OS View:
      - Transfers data along databus from keyboard into memory
      - Loads data from memory to regfile, returns to user
      - Programmer/User never knows how complex things are!

22

Terminology: Process

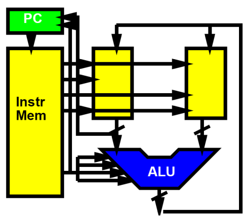
- Process
  - A virtualization of the physical processor
    - an instance of a program in execution
  - Virtual processor



23

What does our program see?

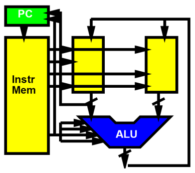
- Physically
  - One processor
    - One PC
    - One data memory
    - One instruction memory
  - These are its state
    - Terminology: context



24

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



25

Execution Exercise

- Simulate one of the 2 cases (as indicated on your worksheet) for the 12 cycles shown.

26

Execution: Getting Started

		DMEM			
Cycle	PC	0	1	2	3
Initial	0	5	35	255	66
+1	1	5	35	0	66
+2	2	5	1	0	66

A

IMEM	0	DMEM[2]=DMEM[2]-DMEM[2]; PC=PC+1
	1	DMEM[1]=DMEM[2]+1; PC=PC+1

27

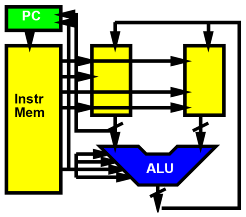
Execution Exercise

- What is the state for the +12 cycle?
- What is the state for the +6 cycle?

28

One Processor, One Program

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



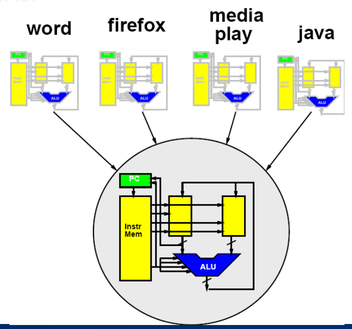
29

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

30

VIRTUALIZATION

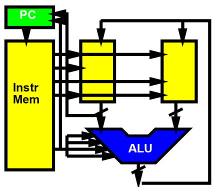


KEY IDEA

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

EXECUTING THE PROGRAM

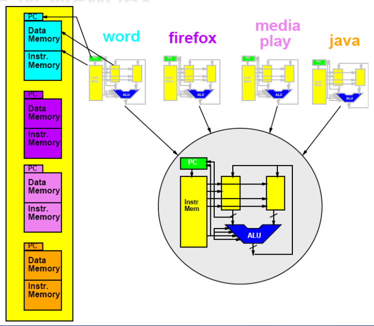
- What is the state of the processor?
  - Value of Program Counter (PC)
  - Contents of instruction memory
  - Contents of data memory



KEY IDEA

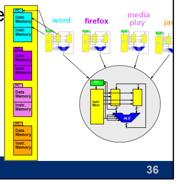
- Can capture state of a processor
  - All the information that defines the current point in the computation
  - i.e. program counter, data and instruction memory
- Can save that in memory
  - A different memory from what the process sees
  - (could be different range of addresses)
- Fully represents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

STATE IN MEMORY



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 a whole machine



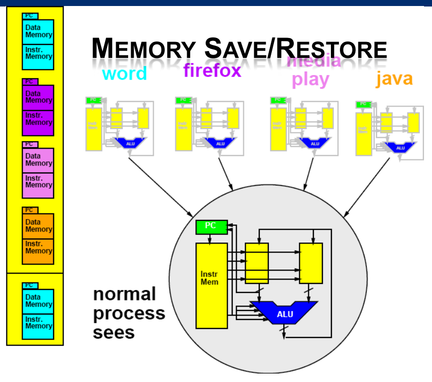
Memory?

- "save all of memory" ?
  - Must have more memory
  - Enough to hold all the memory of all the running programs == all the processes
- **Each program has view that it owns machine**
  - Each may put program in same place?
  - Shouldn't have to know about other programs, where they use memory

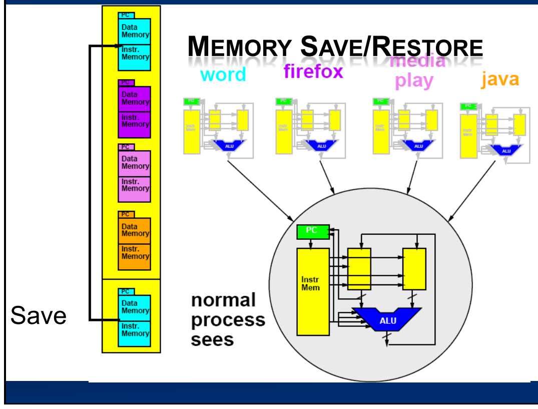
Saving Memory?

- **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 elsewhere
    - more memory not visible to program
    - On disk

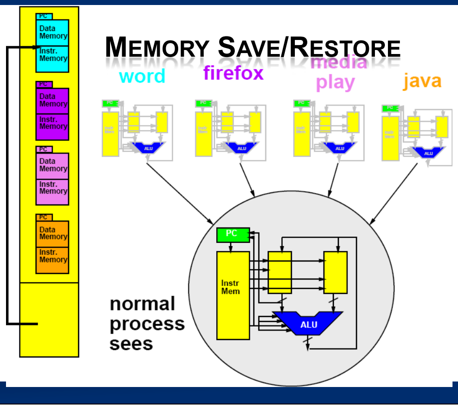
Memory Save/Restore



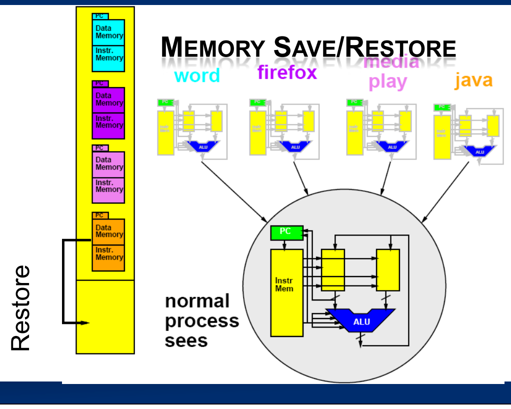
Memory Save/Restore



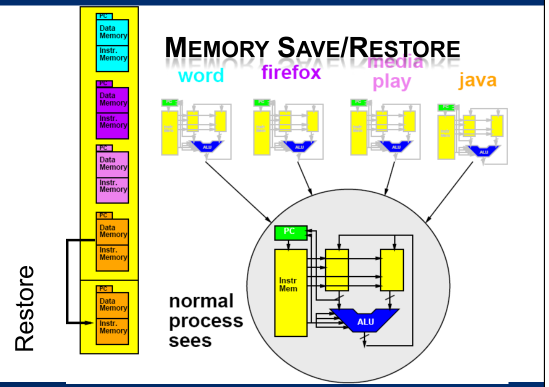
Memory Save/Restore



Memory Save/Restore



Memory Save/Restore

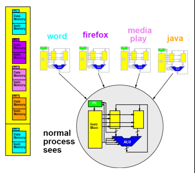


Saving Memory?

- 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
  - Avoid copying
  - Virtualizing Memory as well
    - Translate processor addresses

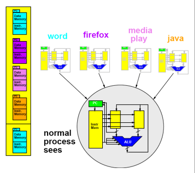
Management Program

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



Time-Sliced Sharing

- Simplest version:
  - Run each process for 10,000 cycles
  - Then swap to next process
  - Looks like each 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 this is
    - appropriate?



Time Switch Exercise

- Write down the +6 cycle state from the opposite case
  - This is your "swap back in" of task

Time Switch Exercise

- Simulate from +6 cycles
- What is the state for the +12 cycle?
- Compare earlier solutions

Time Switch Exercise Demo

- Simulating a case:
- Processor runs A for 6 cycles
  - Then stores off to memory.
- Processor runs B for 6 cycles
  - Then stores off to memory
- Processor reads A state from memory and runs for another 6 cycles
- Processor reads B state from memory and runs for another 6 cycles

Device Virtualization

- Similar concept
  - Identify state of device
  - Save/restore state as use "virtual" device
- Window as virtualization for screen
  - May not even be visible (e.g. minimized)

Upcoming Lab

- Explore linux and processes on linux
- Monday after next (4/9)

- This Monday (4/2) – Processor Lab

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

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

Learn More

- CIS380 – Operating Systems