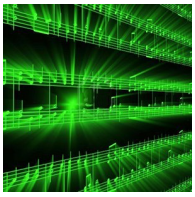
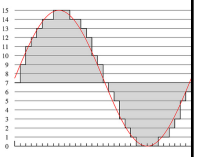
Penn Engineering ESE





Lecture #9 – Operating Systems (OS)

ESE 150 – Digital Audio Basics

ESE150 Spring 2020

Based on slides © 2009--2020 DeHon

Motivation

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

…A few things…

- **Browse web**
- **Play game**
- **Receive incoming phone calls**
- **Compose (receive, send) text messages**
- **Check weather**
- **Check/notify of incoming email**
- **Provide navigation**

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
  - 1 instruction per cycle
  - How many instructions per second require?
- **Compute: 44,000 x 500 x 1 = 22 x 10^6**
- **What fraction of a 10^9 instruction per second processor does this use?**
- **$(22 \times 10^6)/10^9 = 0.022 = 2.2\%$**

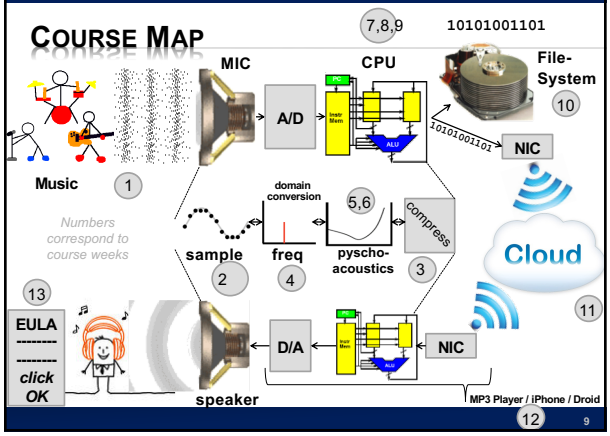
OBSERVATION

- If we dedicate a processor to MP3 decoding
  - It will sit idle most of the time
    - 97.8% in our example
- MP3 decoding (and many other things) do not consume a modern processor
- Idea: Maybe we can share the processor among tasks?

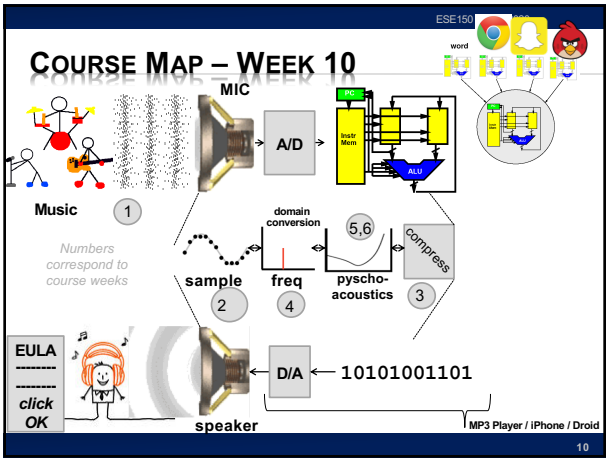
OUTLINE

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

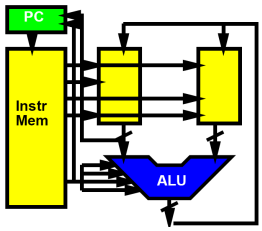
COURSE MAP



COURSE MAP – WEEK 10



"STORED-PROGRAM" PROCESSOR

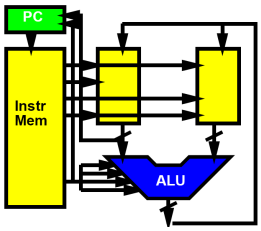


- By filling in memory, can program to perform any computation

ROLE OF OPERATING SYSTEM

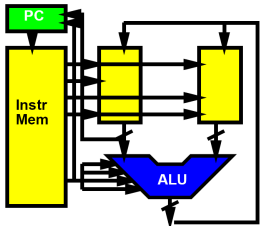
Programming the Processor

- **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…



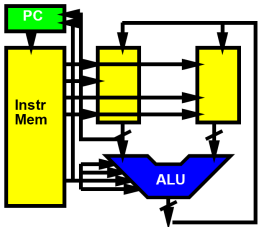
More than one Program?

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



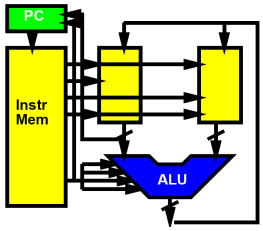
More than one Program?

- **How could we have multiple applications?**
  - (just run one at a time for now)
- **while (true)**
  - R=getResponse("Choose Progam");
  - switch (R) {
    - case 1: startProg1(); break;
    - case 2: startProg2(); break;
    - …
    - default: print("invalid Choice");
    - Break;
  - }



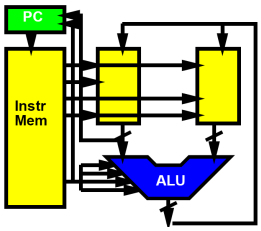
Coordination?

- **Does every program need to know about every other program?**
  - Implications and Viability?
- **IoT device that runs 3 unchanging tasks?**
  - Reasonable
  - Compile together
  - Each knows where other stores data to avoid or share



Coordination?

- **Does every program need to know about every other program?**
  - Implications and Viability?
- **IoT device that runs 3 unchanging tasks?**
- **Smart phone that allows 3rd party apps**
  - Think: AppStore, Google Play
  - Cannot know everything else will be on computer
  - Some won't have even been written when others loaded
  - Cannot place in fixed location in memory



Role of Operating System: Sharing

- **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**

VIRTUALIZATION

VIRTUALIZATION

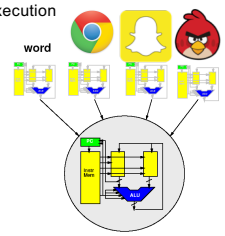
- **Providing an abstract view separate from the physical view**
- **Hides physical view**
- **Provides abstract view to software**
  - Abstract from physical resource limits

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

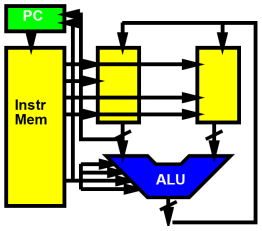
TERMINOLOGY: PROCESS

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



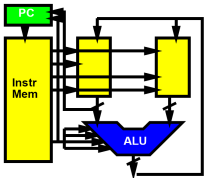
WHAT DOES OUR PROGRAM SEE?

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



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



PRECLASS: EXECUTION EXERCISE

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

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

PRECLASS EXECUTION EXERCISE

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

EXECUTION EXERCISE

- What is the state for the +12 cycle? (A, B)

A

		DMEM			
Cycle	PC	0	1	2	3
12					

		DMEM			
Cycle	PC	0	1	2	3
12	4	5	5	6	-2

EXECUTION EXERCISE

- What is the state for the +12 cycle? (A, B)

A

		DMEM			
Cycle	PC	0	1	2	3
12	4	5	5	6	-2

B

		DMEM			
Cycle	PC	0	1	2	3
12					

EXECUTION EXERCISE

- What is the state for the +12 cycle? (A, B)

A

		DMEM			
Cycle	PC	0	1	2	3
12	4	5	5	6	-2

B

		DMEM			
Cycle	PC	0	1	2	3
12	4	5	2	222	192

Execution Exercise

- What is the state for the +6 cycle? (A, B)

A

		DMEM			
Cycle	PC	0	1	2	3

B

		DMEM			
Cycle	PC	0	1	2	3

Execution Exercise

- What is the state for the +6 cycle? (A, B)

A

		DMEM			
Cycle	PC	0	1	2	3
6	2	5	2	1	-3

B

		DMEM			
Cycle	PC	0	1	2	3

Execution Exercise

- What is the state for the +6 cycle? (A, B)

A

		DMEM			
Cycle	PC	0	1	2	3
6	2	5	2	1	-3

B

		DMEM			
Cycle	PC	0	1	2	3
6	2	11	0	222	192

Full Simulation

A

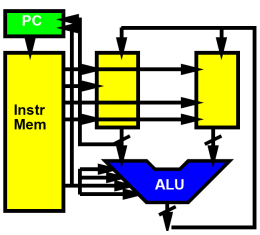
		DMEM			
Cycle	PC	0	1	2	3
Initial	0	5	35	255	66
+1	1	5	35	0	66
+2	2	5	1	0	66
+3	3	5	1	0	66
+4	4	5	2	1	66
+5	5	5	2	1	-3
+6	2	5	2	1	-3
+7	3	5	2	3	-3
+8	4	5	3	3	-3
+9	5	5	3	3	-2
+10	2	5	3	3	-2
+11	3	5	3	6	-2
+12	4	5	4	6	-2

B

		DMEM			
Cycle	PC	0	1	2	3
Initial	0	23	255	222	192
+1	1	23	0	222	192
+2	2	23	-1	222	192
+3	3	23	-1	222	192
+4	4	23	0	222	192
+5	5	11	0	222	192
+6	2	11	0	222	192
+7	3	11	0	222	192
+8	4	11	1	222	192
+9	5	5	1	222	192
+10	2	5	1	222	192
+11	3	5	1	222	192
+12	4	5	2	222	192

One Processor, One Program

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

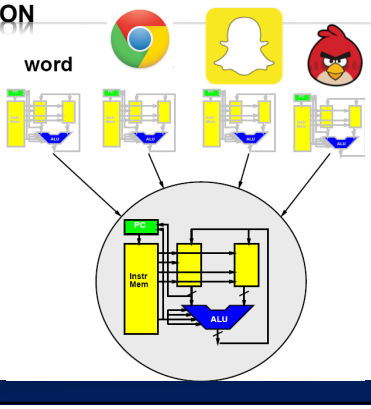


Virtualization

- **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

VIRTUALIZATION

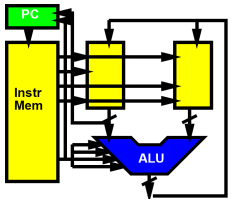


KEY IDEA

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

REMEMBER

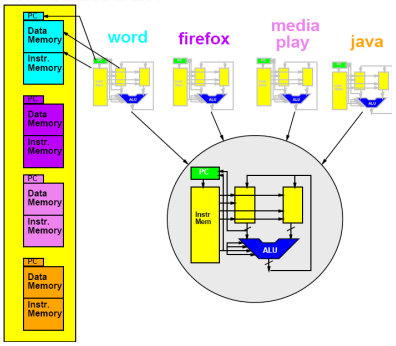
- **State of the processor**
  - Value of Program Counter (PC)
  - Contents of instruction memory
  - Contents of data memory



KEY IDEA

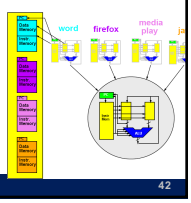
- **Can capture state of a processor**
  - All the information that defines the current point in the computation
  - i.e. program counter, data and instruction memory
- **Can save that in memory**
  - A different memory from what the process sees
  - (could be different range of addresses)
- **Fully 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 whole machine
  - Just need to restore/save state around epochs where the process gets to run on the processor



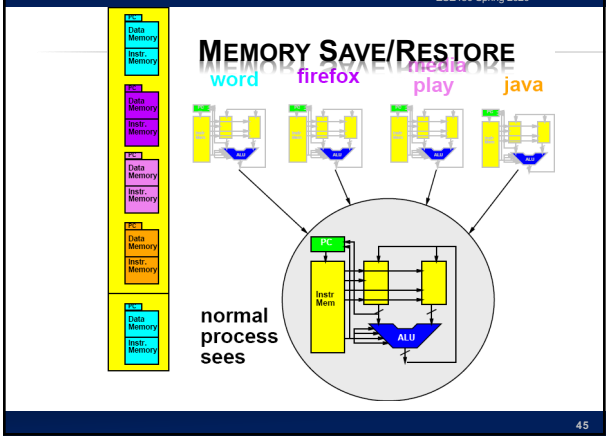
MEMORY?

- **"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

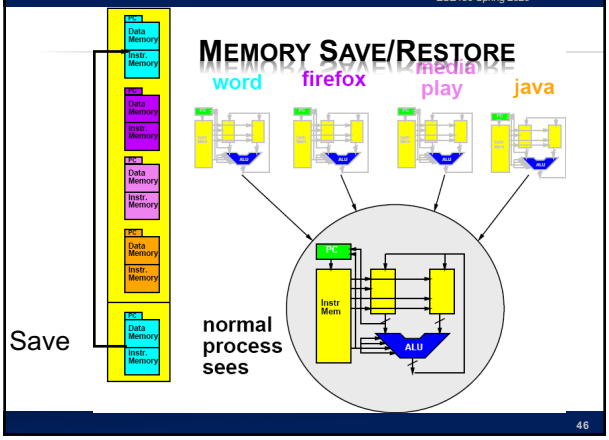
SAVING MEMORY?

- **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

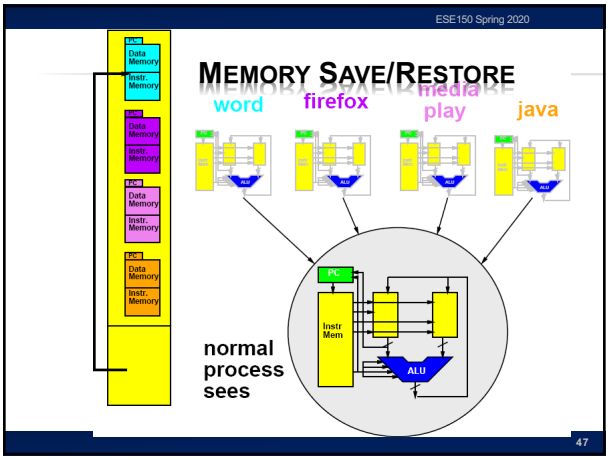
MEMORY SAVE/RESTORE



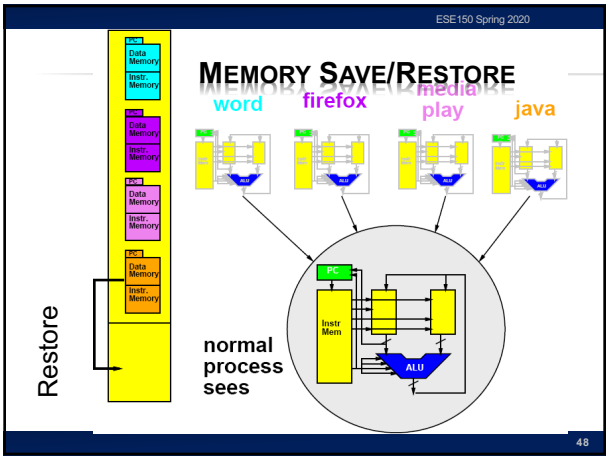
MEMORY SAVE/RESTORE



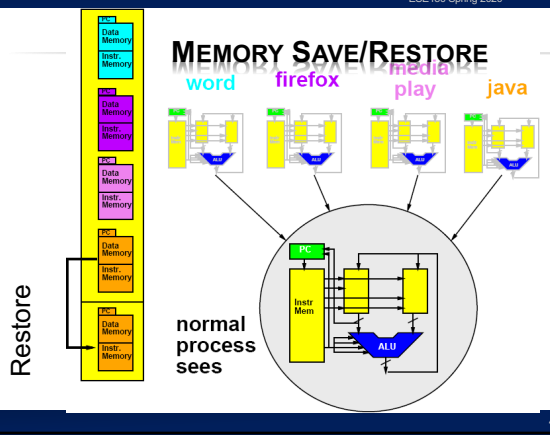
MEMORY SAVE/RESTORE



MEMORY SAVE/RESTORE



Memory Save/Restore

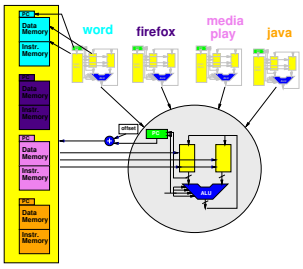


Saving Memory?

- **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

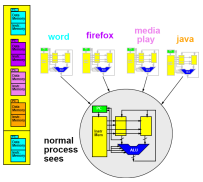
Process Base Offset Register

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



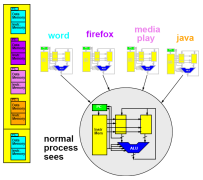
Management Program

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



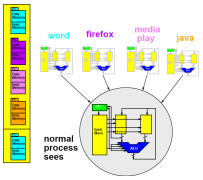
Time-Sliced Sharing

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



Time-Sliced Sharing

- **More sophisticated:**
  - Assign uneven time to processes
  - Also change when process…
    - waits for input
  - What are cases where
    - Uneven time appropriate?
      - Running game and mp3 decode
        - Mp3 decode only need 2% of time
      - Checking email, text, calls
        - While compiling program
    - Valuable to switch on input?
      - User moves mouse (touches screen)
      - Text message arrives



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**
- **What should happen? (results should we get?)**

Time Switch Exercise

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

A

		DMEM			
Cycle	PC	0	1	2	3
6	2	5	2	1	-3

B

		DMEM			
Cycle	PC	0	1	2	3
6	2	11	0	222	192

Execution Exercise

- **Simulating from +6 cycle:**
  - What is the state for the +12 cycle?

A

		DMEM			
Cycle	PC	0	1	2	3
12					

		DMEM			
Cycle	PC	0	1	2	3
12	4	5	5	6	-2

Execution Exercise

- **Simulating from +6 cycle:**
  - What is the state for the +12 cycle?

B

		DMEM			
Cycle	PC	0	1	2	3
12					

		DMEM			
Cycle	PC	0	1	2	3
12	4	5	2	222	192

Same as got earlier.

Full Simulation

A

		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
+3	3	5	1	0	66
+4	4	5	2	1	66
+5	5	5	2	1	-3
+6	2	5	2	1	-3
+7	3	5	2	3	-3
+8	4	5	3	3	-3
+9	5	5	3	3	-2
+10	2	5	3	3	-2
+11	3	5	3	6	-2
+12	4	5	4	6	-2

B

		DMEM			
Cycle	PC	0	1	2	3
Initial	0	23	255	222	192
+1	1	23	0	222	192
+2	2	23	-1	222	192
+3	3	23	-1	222	192
+4	4	23	0	222	192
+5	5	11	0	222	192
+6	2	11	0	222	192
+7	3	11	0	222	192
+8	4	11	1	222	192
+9	5	5	1	222	192
+10	2	5	1	222	192
+11	3	5	1	222	192
+12	4	5	2	222	192

Illustration

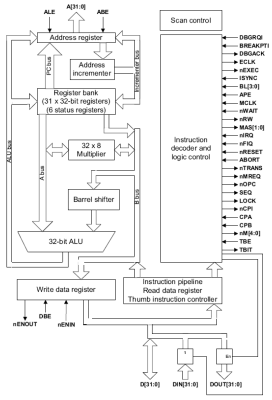
- **Did not need to know anything other than the state to complete the computation**
- **If we store the state and restore (even on a different processor) can continue the program execution.**

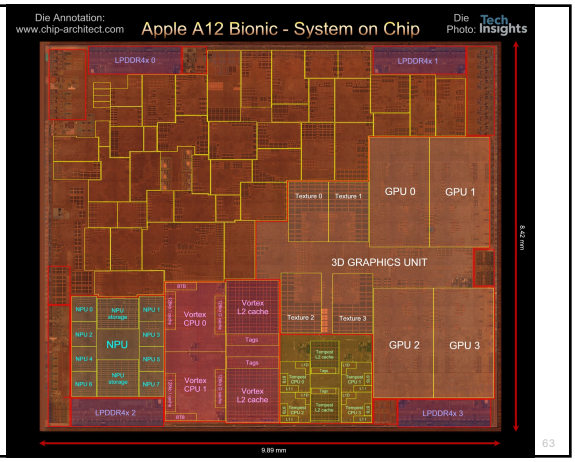
Review: Key Idea

- **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**

iPod Processor

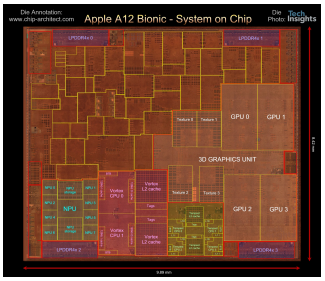
- **Early based on PortalPlayer series**
  - Two ARM7TDMI cores
  - 80MHz each
- **Guesses are ARM7 or ARM8**





Apple A12

- **84mm², 7nm**
- **7 Billion Tr.**
- **iPhone XS**
- **6 ARM cores**
  - 2 fast
  - 4 low energy
- **4 custom GPUs**
- **Neural Engine**
  - 5 Trillion ops/s?



Upcoming Lab

- **Explore Linux OS and processes on Linux**
  - See processes sharing processors
  - Lab out
  - No new software to install…this time!
- **Remember feedback form for lecture, lab**

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**