CSE 134 - Embedded OS

Elijah Hantman

Operating Systems

Why Operating System?

- Lots of types of hardware
- Different Instruction sets
- Possibly multiple programs on the same computer
- timesharing
- Concurrency
- Input/output devices
  - Mice
  - keyboards
  - speakers
  - monitors
  - Networking
  - Printers
  - Disk

An Operating System's job is to give a common layer for multiple types of programs to run on top of.

The part of an Operating System which manages hardware access is called the Resource Manager

What does an operating system do?

They ensure the hardware is connected, they translate the raw output of hardware into understandable format.

Where is the Software?

- On top we have User Mode. This includes all applications, games, etc.
- User Mode talks to Kernel mode software to interface with hardware. Kernel mode is where the operating system lives, and is responsible for directly handling hardware. This is also where device drivers live, and any programs which directly edit memory, set permissions, etc.

Fun fact, for desktop systems, generally they have hardware support for kernel mode. ie: They allow editing special memory like what code is run when specific hardware things happen. as well as editing things like the virtual page tables used by the cpu to virtualize memory.

Resource Manager

- Time(cpu) and space(memory) sharing
- Split CPU time between multiple concurrent tasks
- Split memory between multiple concurrent processes
- Split I/O between multiple processes

Operating System is a Hardware Abstraction and Resource Manager

Computer Hardware Review

- Two busses, memory and control
- Bus connects all hardware
- CPU has memory management unit to control the bus and manage the CPU cache and i/o.
- I/O covers all components except memory. CPU only has Memory management unit since the I/O is mapped into memory from the perspective of a program.
- One of the jobs of the MMU is to allow the CPU to interpret I/O as volitile memory
- CPU only interacts with I/O via memory as an intermediary

CPU Pipelining

- Pipelines allow for temporal parallelism on a hardware level.
- Can multiply the amount of work a CPU can do
- Increases Throughput without changing latency
- Pipelines can be linear or they can be "superscalar" and include spatial parallelism in addition to the temporal parallelism inherent to pipelines.

> Intel and AMD have a concept of ports which can perform different types of instructions. Instructions are also decomposed into micro instructions which are then split between ports and executed.

Memory Hierarchy

- We use a hierarchy because memory speed is inversely related to its density. To get advantages of fast memory like SRAM and the large size of DRAM, Magnetic Disk, Magnetic Tape, we use both kinds and move the memory we need into the fast memory and move memory out to disk to save space.
- RAM $\rightarrow$ Random Access Memory

> Can be read anywhere at all time, can be written to and read from. Usually volitile.

- ROM $\rightarrow$ Read Only Memory
- EEPROM $\rightarrow$ Eletronically Enabled Programable Read Only Memory

Multithreaded and Multicore Chips

- Often cores share L2 and L3 cache.
- On GPUs groups of 16-32 cores will share memory, as well as the global memory shared by all cores.

> Fun fact, Intel initially used a shared L2 cache design, and AMD originally used a split L2 design.
>
> AMD usually turned out faster. But Intel had the benefit of using a large L2 cache which reduced the complexity of synchronizing the caches and ensuring memory was not corrupted.
>
> MESI, MSI, etc.
>
> AMD is faster for many different programs which do not have shared memory. Intel is faster for using multiple cores working on the same memory.

Disk

- Magnetic disks, come in a large stack. Single write head for each surface, ie: 2 per disk.
- Segment on the first surface is where the hardware will look for instructions on start up.
- Supernode controls manages mapping each disk to a number for reads and writes.

Device Drive

- Software which interfaces with some hardware.
- Complicated usually

> Converts read commands to commands the specific hardware understands.
>
> For a HDD it converts memory locations into arm movements and disk number.

- Different Modes
  - Polling

    > Write information into memory and occasionally check. Works both ways, software might write memory and hardware occasionally reads.
    >
    > This is often used for Games to read Input, every frame the game will check for input and then update it's internal tracking.

  - Interrupt

    > Send special signal to the CPU which forces CPU to execute specific code to handle input.

  - DMA

    > Directly map device information into memory. Have some kind of mapping between memory addresses and the device directly.
    >
    > This is for things like GPUs or sometimes memory mapped I/O which is a Userland abstraction.

- I/O via Polling
  - Driver issues commands
  - Driver keeps checking on device until it is ready
  - Big Use of CPU
  - Called Programmed I/O, not common anymore
- Interrupt I/O

  - CPU is working and doesn't check
  - Device sends signal to CPU which causes it to run special Interrupt handler
  - Driver then Sends commands and handles input before returning to normal execution
  - Requires CPU hardware support, usually the return address is kept in special register
  - Requires special handling to store register state
- DMA I/O
  - The Device directly writes into memory
  - CPU interaction is kept to minimum