Math problems

- CPU scheduling processes
    - Determine which process gets cpu at given time
        - And algos
            - Priorities
    - Process time
    - Avg wait time
    - Schedule cpus to processes
        - 20 processes, 4 cpus, etc

Intro:

- Mips only program running
    - 2 programs running = access or manipulate same memory
        - Array put in stack
            - Data section, 10 million, 10 thousand
                - Puts at start
        - Other program puts an array at that same place
            - That's an issue
    - Solution – trick program
        - Program thinks running to 10 mill but really writes to 11 mill
        - Problem, still can overflow
            - Program one can still go up till area of next program
        - We need protection
            - Memory of 1 program shouldn't access other program
            - Canary
                - Is a software thing
                    - Need to check if in memory space… need to check if goes through allotted array
                        - More instructions to check if we go beyond memory space
                - Hardware solutions
                    - byte in hardware, do mem access.. check byte if action permitted
                        - Op system sets byte
                        - Limit vector on bottom and top
                    - Faster
                    - Hardware register with bounds for memory and loaded by OS
                        - 2 programs running

Only good for 1 program

- What about 2 programs running
    - Need more bytes?
        - We have lots of processes
            - More than CPUS
                - 1 process has CPU for certain period of time vs other one

- 1 program running at 1 time
    - Context switch
        - Switch from 1 program to another
            - Home vs office
        - Data related to particular process is moved into mem and out (or registers)
        - Process A running , all data loaded to registers
            - Then switch processes and the data gets moved out to memory (RAM)
                - And program B registers load
                    - Copies upper/lower limit and loads to hardware
                    - Then mem read and hardware checks right registers
- How to create feeling that a process is the only thing running on computer
    - Process = program that is running and executing at the moment
    - Single program can spawn processes (more than 1)
- Threads
    - Classic defn: lightweight process
        - Re-uses some data or code of parent process
            - Not a complete clone
                - Shared some stuff with parent process

OS goal

- Create feel that only 1 process is running on computer even though others running
- Provide security
    - Process cant mess with other process code
- Equity
    - 1 process shouldn't clog resources of whole computer

What are Resources?

- Memory
- Cpu
- Ram
- IO
    - Typing
        - Switch screen, we want what we type to go with us
- Printing
    - Paper beginning and then other process taking
    - Batch mode
        - Handles batch and only does 1st item, then finishes -> move on
        - Process tells printer to print
            - Context switch doesn't matter
        - Printer has memory
            - Puts in queue and prints

Exam mostly numbers but some definitions

Os in mips = syscall

Print to screen

- Without programming every line

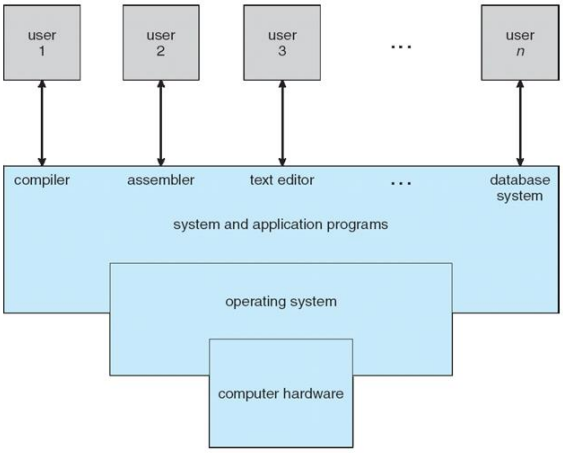
Syscall

- Agent between you and the hardware

System program

- Task manager
  - Kill processes etc.
  - Or halt them
- Not part of the OS
  - Not needed all the time
- We cant write It ourselves

Four Components of a Computer Syste



Kernel

- Essential part of OS , runs all the time
- OS is a **resource allocator**
  - Manages all resources
    - Memory
  - Decides between conflicting requests for efficient and fair resource use between processes
  - OS is a **control program**
    - Controls execution of programs to prevent errors and improper use of the computer
    - Prevents bugs/errors
    - Going to mem of other program

Definition:

No universally accepted definition

"Everything a vendor ships when you order an operating system" is good approximation

- But varies wildly

"The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.

The one program that manages, controls and provides services for all the other programs running on the computer - ND

Bootstrap program

- Firmware not on HDD
- Code run automatically
- Load OS or tell it to load first MBR – masterboot record
  - On HDD which will tell where it is

Program in set place on harddrive or in firmware itself

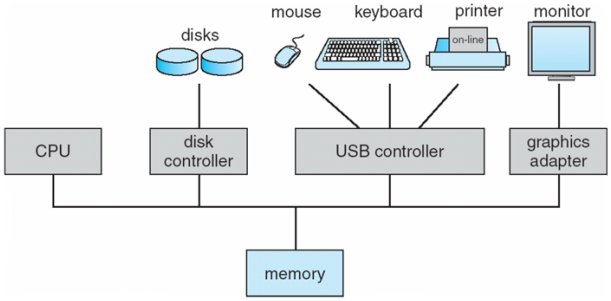
Firmware

- Between software and hardware
- Program burnt into chip on motherboard
    - Not a processer but it cant be altered
- Embedded software

Every hardware device has controller

- Internediary between device and cpu
- Cpu can talk to memory directly
- But not disk drive directly

- Computer-system operation
    - One or more CPUs, device controllers connect through common bus providing access to shared memory
    - Concurrent execution of CPUs and devices competing for memory cycles



Mem = volatile memory

- Gets wiped
- Such as RAM

Disk = hard drive

- Non volatile

Floppy drive

- Disk
- Software

Usb = disk too

Cpu talk directly

- Very fast cpu talking to slow disk
    - Would have to wait for disk to get answer
    - We have a disk controller
        - Batch
        - We want file, tell if ready
        - Then interrupt cpu
            - Have file then read it
        - Controller is hardware
            - Fast

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an interrupt

Memory isn't slow, so doesn't need controller

Interrupt

- office manager
  - Find out which worker finishes work
    - Micro-manage / **polling**
      - Ask each person, done?
        - Then move on
      - Does controller have what we need (vector)
- Another way
  - Worker tells manager when they finish work
    - **Interrupt**

Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines

Interrupt architecture must save the address of the interrupted instruction

A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request

An operating system is **interrupt driven**

Interrupt vector

- Ignorable
- Not ignorable

CPU bit lights up to say if interrupt occurred

- But which?
- Know someone called me but who?
- Interrupt but then **poll**
  - Know we get some new data

Interrupt vector

- Table with #s
- Each row
  - Interrupt occurred
  - Then address of program to run when that occurs
  - Disk is ready -> cpu does disk action

Handle interrupts

Interrupt happens

- Context switches for CPU
- Stores state of CPU – registers and Program count

Div / 0

- Interrupt and

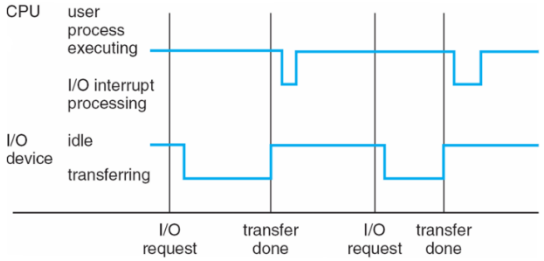
Mouse pressed

- Program runs

- The operating system preserves the state of the CPU by storing registers and the program counter
- Determines which type of interrupt has occurred:
  - **Polling:** Once the interrupt occurs, the system must determine which device, of all the devices associated with a given IPL or IRQ, actually interrupted. It does this by calling all the interrupt handlers for the designated IPL or IRQ, until one handler **claims** the interrupt.
  - **vectored** interrupt system: When the device interrupts, the system enters the **interrupt acknowledge cycle**, asking the interrupting device to identify itself. The device responds with its interrupt vector. The kernel then uses this vector to find the responsible interrupt handler.
- Separate segments of code determine what action should be taken for each type of interrupt

- Context switch to handle the mouse program
- Mips
    - Only copy S for fn calls
    - T is temp
    - But in such a case copy all, as interrupt isn't planned
        - Can be anywhere

Interrupt Timeline



Run code then go back to OG program

Device = controller in this case

Vector = faster than polling

Handle 1 interupt and another occurs?

- Only if hardware interrupt during interrupt
    - Often times its ignored
    - Other is put in a queue with 1 space
        - So only 1 more interrupt, others get lost

Two Different I/O Schemes

- After I/O starts, control returns to user program only upon I/O completion
    - Wait instruction idles the CPU until the next interrupt
    - At most one I/O request is outstanding at a time, no simultaneous I/O processing
- After I/O starts, control returns to user program without waiting for I/O completion
    - **System call** – request to the OS to allow user to wait for I/O completion
    - **Device-status table** contains entry for each I/O device indicating its type, address, and state
    - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

Syscall can call an interrupt

- Get input from user
- Print to screen
- Write to file
- Exit program
- Write to disk

Fredj

- Syscall how it works

Kernel mode and user mode

- Flag to switch modes
- Shutting down computer but make sure to do actions first
- Or memory
    - 2 processes accessing same spot
        - Lock it

Write to disk via OS

- Process a can get to file that doesn't belong to him
    - Make all access to HDD through OS and check ownership via files and processes
    - Don't want process to have full access to hdd
- What if process jumps to line of access HDD (code lines) to write to disk ignore the OS
    - Hardware bit lit
    - Solution
        - Limit of memory can access (limit base register)
            - Jump to OS is out of memory range
        - Other solution
            - Special bit says permission to access code

- OS turns it off and on
    - If I jump to space to write to file, bit off
- If called via system call
    - bit on, access codes = kernel mode
        - Then write , then shut off to access back to program

OS stops runs, let program run and write to disk

- Cant run on cpu at same time

Kernel mode bit prevents user from accessing code not allowed to access

- Syscall
    - Happens:
        - Limit and base register change to include area of code we want to do

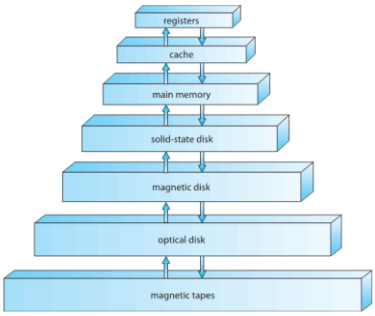
Direct Memory Access Structure

DMA

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

- CPU copies cache form memory byte after byte
- So dma does blocks of memory into cache

Storage-Device Hierarchy



cache – bring close to memory

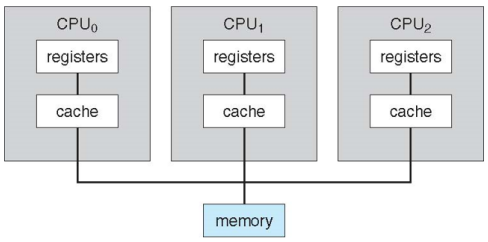
Faster

Smaller

Expensive

Multi core vs multi processer

Symmetric Multiprocessing Architecture



Originated in 1960s. Processors are connected by a separate bus. OS must support parallel processing.

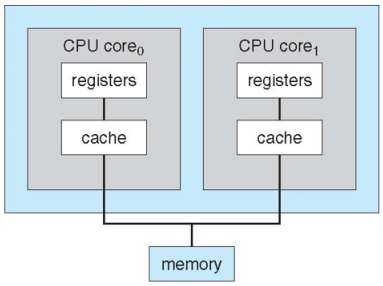
Multi processer

- 1 motherboard
- Multiple devices
- Multiple cpus on the motherboard

Vs multiple cores

- 1 cpu but more cores inside it

A Dual-Core Design



Both cores are on a single integrated circuit. Still requires OS support.

each cp has own registers and cache

Run 2 programs at same time

- Cant have 3 or more

1 program prints hello world, either runs on core 0 or 1…

- Wont gain speed

Or write program which does tasks in parallel

- OS divides work into processers t be done faster

Dual core = need to write in mind with multi threading in mind

Calculate decimal spaces

- Algo does different ops… does in parallel to save time
- Simultaneous operations

Von neuman

- Mips
- Have separate ALU, processing unit, controller (with lines), memory, external HDD, I/O
- Computer with these parts
- Computer handled store program
    - Can have calculator which si computer not programmable

Mips pipeline

- Memory – twice
    - But only once
    - 1 to access instruction fetch
    - Other for memory access
- 2 windows into memory
    - Same unit

Read from memory in IF stage

- Through bus (some kind of wires) to CPU
- Read from memory access also through bus
- In pipeline both simultaneously
    - 1 row out of memory handle each time
    - Handle 2 cars same time?
    - Or cycle long enough to have 2 memory accesses

Or 2 busses to handle read/write

2 for read , 2 for write

IF and memory

Read fist half, write second

Von Neuman architecture

- A processing unit that contains an arithmetic logic unit and processor registers
- A control unit that contains an instruction register and program counter
- Memory that stores data and instructions
- External mass storage
- Input and output mechanisms[
- The term "von Neumann architecture" has evolved to mean any stored-program computer in which an instruction fetch and a data operation cannot occur at the same time because they share a common bus. This is referred to as the von Neumann bottleneck.

- **Multiprogramming** needed for efficiency
    - **When job has to wait (for I/O for example), OS switches to another job**
    - Single user cannot keep CPU and I/O devices busy at all times
    - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
    - A subset of total jobs in system is kept in memory
    - One job selected and run via **job scheduling**
- **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
    - **Response time** should be < 1 second
    - Each user has at least one program executing in memory ⇨**process**
    - If several jobs ready to run at the same time ⇨ **CPU scheduling**
    - If processes don't fit in memory, **swapping** moves them in and out to run
    - **Virtual memory** allows execution of processes not completely in memory

Divide cpu

10 ms for process a and 10 ms for process b = timesharing / multi task

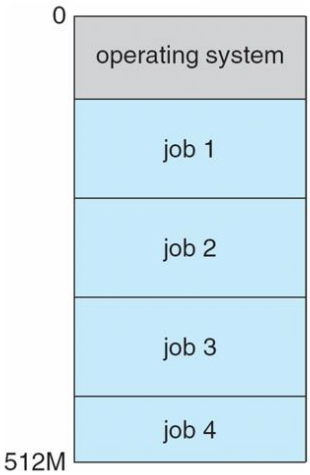
Multi programming

- Give cpu till program finishes (relinquies)
- Or going to I/O
    - Read/write to disk or keyboard
    - Slow operations
        - Please enter age: ….
        - Takes time
    - Now cpu doesn't stay with process
        - It moves to next process and waits for IO
            - Doesn't need CPU
- Run program forever till goes to I/O (then switch)

Timesharing

- Also stop at i/o
- And time limit after 10 ms or 20 ms
- 3 reasons to switch
    - End program
    - User input
    - Time limit

Swapping



- Memory/ram
    - 1/0 represents code of program sitting in memory
        - Copy to HDD and more memory space
        - Not Virtual memory
            - Swapping
- Jobs = process running
- Time sharing between 1,2,3,4,1,2,3,4
- Start job 5…
    - What do we do?
        - Say ran out of memory ?
            - Infinite processes
                - Running and takes memory
        - Or swapping
            - Take job 1 out of memory and put it job 5
            - Maybe leave 1 on disk
                - Not in ready queue
                - Not a candidate for getting CPU
        - Memory = candidate to get cpu
            - On disk, wont get cpu
                - Takes time to copy from disk to memory

    - Swap job 1 back in when the other jobs finish
        - When we have too many ppl in memory
        - Put job on hold
        - Priorities
            - Longest in running
            - LRU
            - Etc…

Virtual memory later

- Take job1,2,3,4
- Cut in half
- Put half on HDD and half in memory
    - More programs can run
    - Or what if we need swapped out code
    - Program spends 90% of time on 10% of code
        - Else if handling examples etc
        - 90% of time is loops etc
    - Use v mem and put the other stuff into HDD and out of memory
        - Figure out main section of code based on use
        - Once in a million when we need that code… slow down

Called Virtual memory

- As if the disk was actual memory but its not
- Half on disk