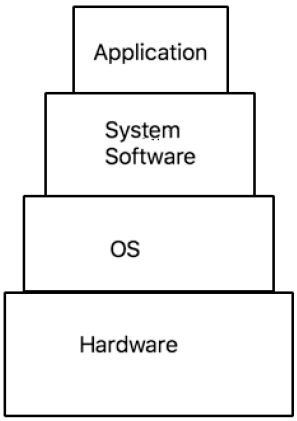
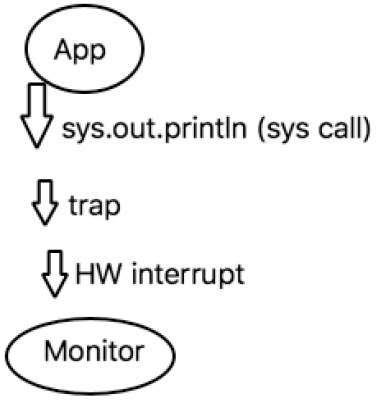
Barrier synchronization
Other synchronization (flag checks and return integers, etc)
You have to provide it will not work: mutual exclusion, progress, bounded waiting
→ no deadlock if all 3 are there
Mutual exclusion: no process will have exclusive access to a resource
Bounded waiting: how long (quantitative measure) process/thread has been waiting.
Starvation

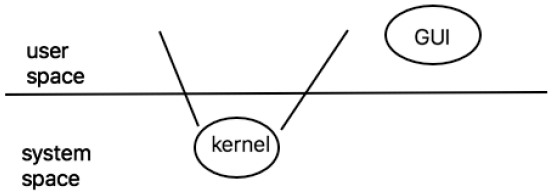
Review throughput assignment and excel file might be in midterm
Non preemptive system = let all process go through
If we make it a preemptive system= what would be changed.



User mode vs system mode:
Why do we need a system mode? To avoid programs using hardware resources.
What could happen? You could write one program that could erase everything. To make sure access to hardware resource is privileged.
Application wants to display on a monitor (hardware resource)
sys.out.println (system call)
That generates a software trap
That generates a hardware interrupt
OS comes in and transmit the command to Monitor
How do we implement this interrupt hardware mechanism?
Mode Bit : switch between 0 and 1.

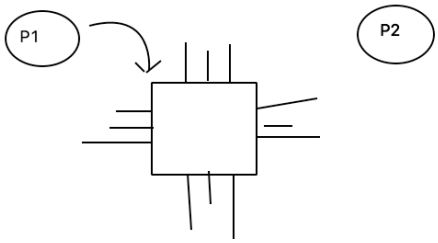


OS Architecture
<u>Batch</u>
→ 0 → 0 →
<u>Microkernel</u>
We take core OS piece .

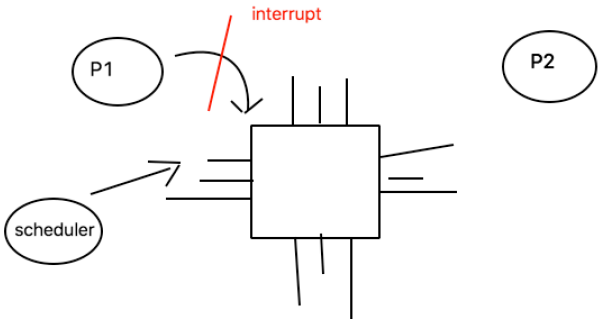


Multitasking: You can load multiple programs to memory

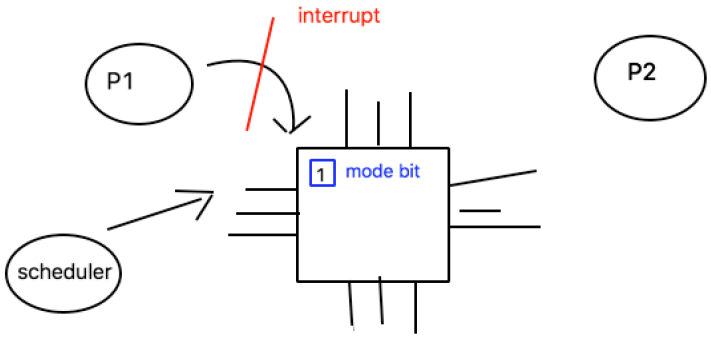
Context Switching



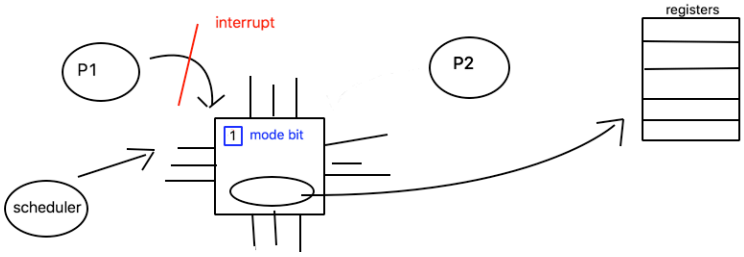
1. P1 is running and hit timeslice
2. Hardware Interrupt



3. OS Scheduler loaded on CPU
4. Scheduler Change the Mode bit (to 1, to be in syst mode) to access resources



5. Save register value in memory except program counter (PC)



6. Then take all data of P2 load them on register except program counter (PC)
7. Change back the Mode bit, Stop the scheduler.