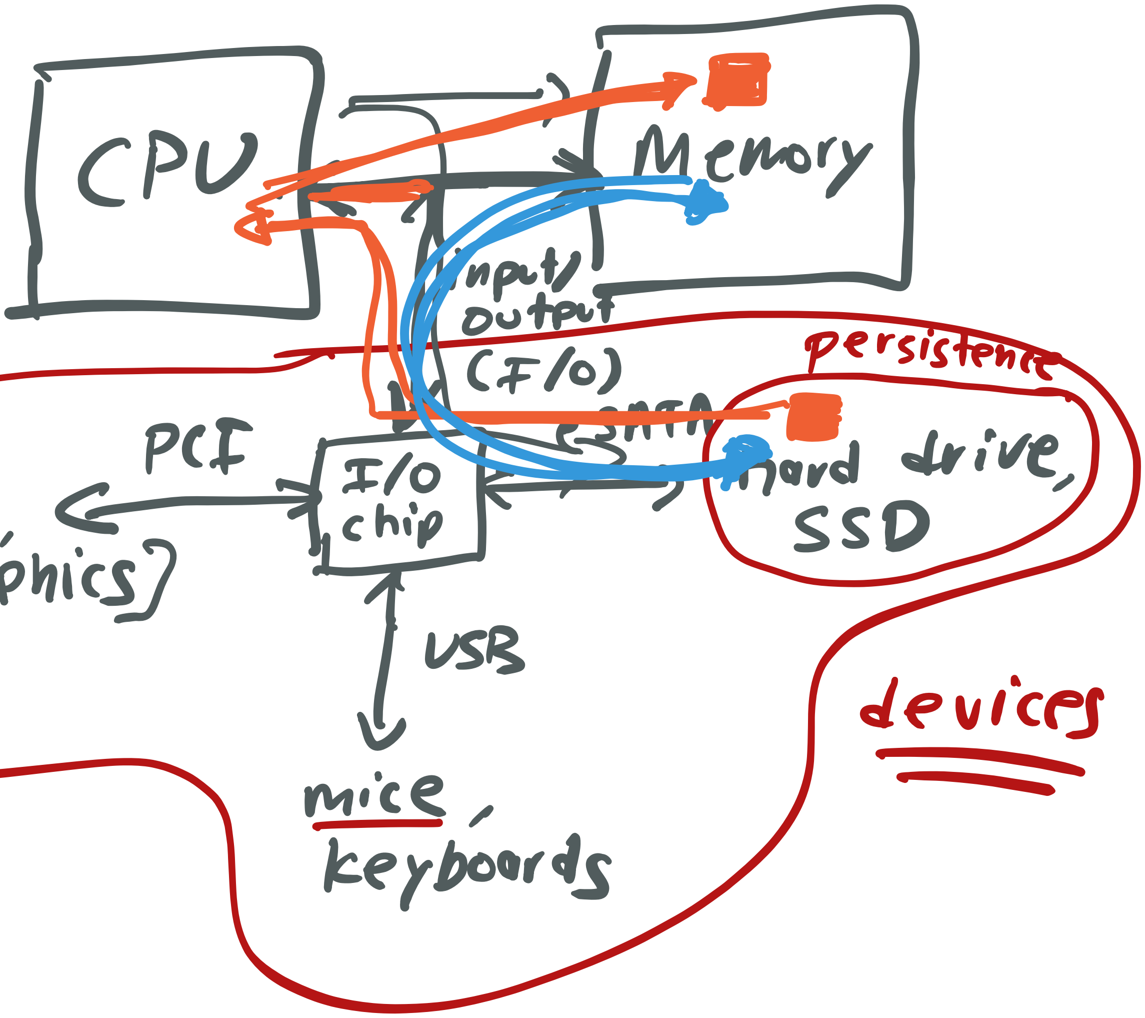
Today [We are starting!]

(volatile, not persistent)
=> power off
=> info lost
instructions
data

-> Virtualization ✓

=> Concurrency ✓

=> [Persistence]



Today:

-> Device interaction (I/O)

-> how?

-> efficient?

Specific Device : Hard Drive

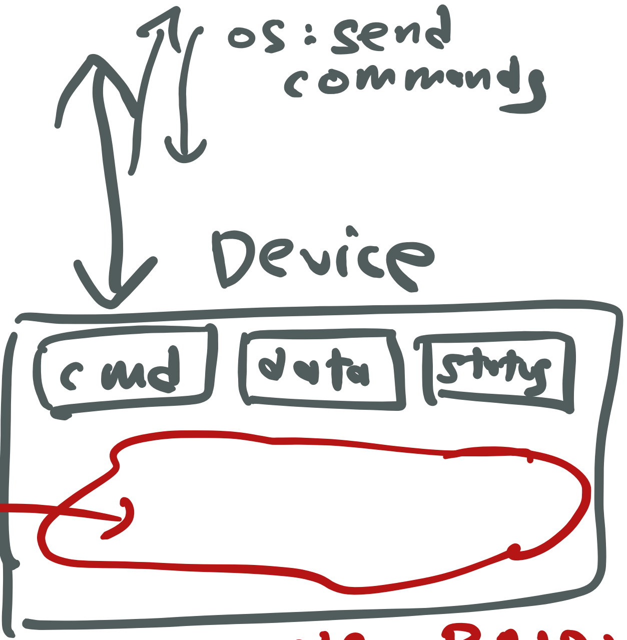
Future:

Device up => [OS : File Systems]

Generic Device:

hardware interface (generalization)

how does OS read/write? → [
→ command register
what to do
→ data register
→ status register
]



os: send commands

varying levels of complexity (simple: mouse to complex: RAID)

2 methods of access:

{ => special I/O instructions
OR
=> memory-mapped I/O }

Protocol for access:

1) check if device is available

```
while (status == BUSY)
    ; // spin
```

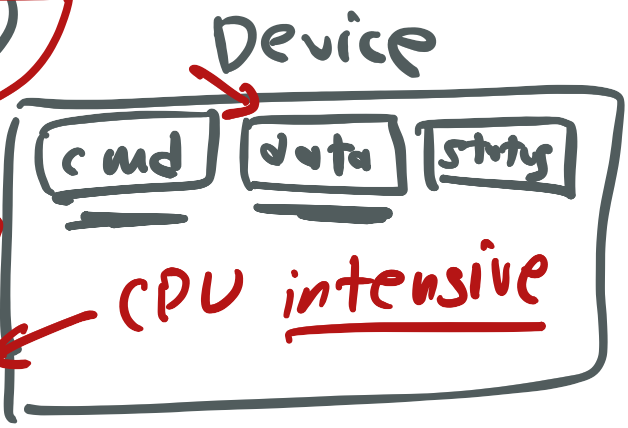
move data

2)
```
for (size of data)
    move data → data register
```

move cmd:

3) write command → cmd register

4) wait:
```
while (status == BUSY)
    ; // spin
```



Problems:

#1 → lot of spinning (polling the device)

#2 → data movement: CPU intensive

Solutions:

#1: instead of spinning, go to sleep (block)

```
while (status == BUSY)
    sem_wait(&io);
```

what wakes it up?

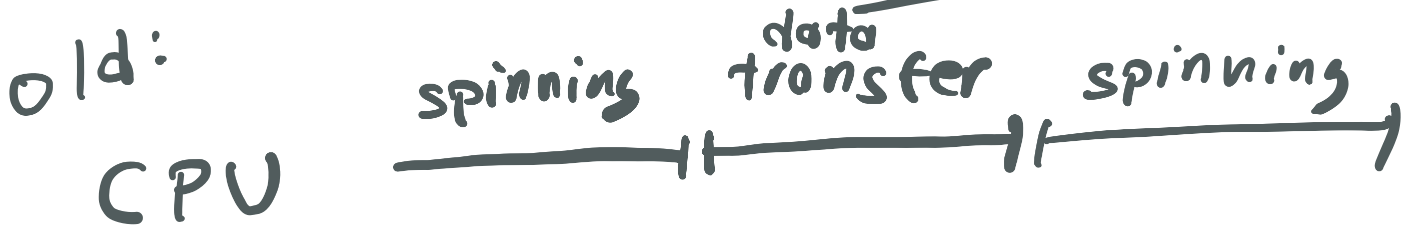
=> interrupt signal to system that device is done

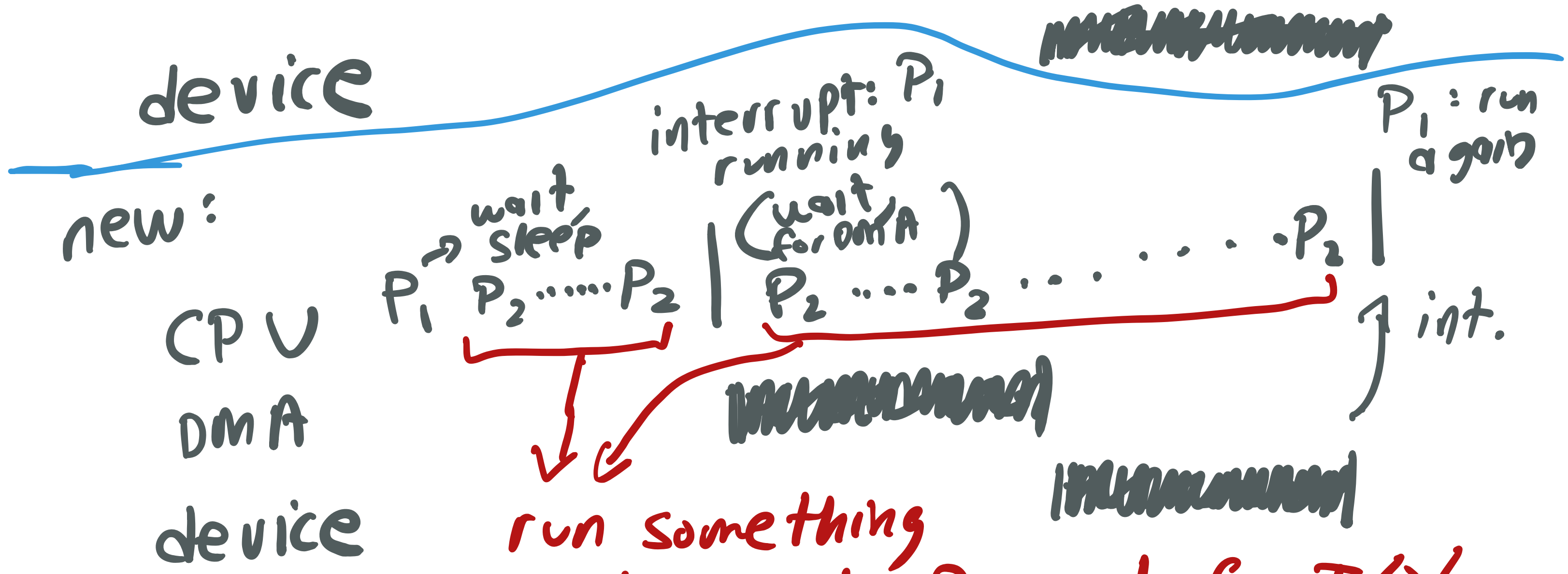
run interrupt handler

=> sem_post(&io);

#2: DMA engine : hardware

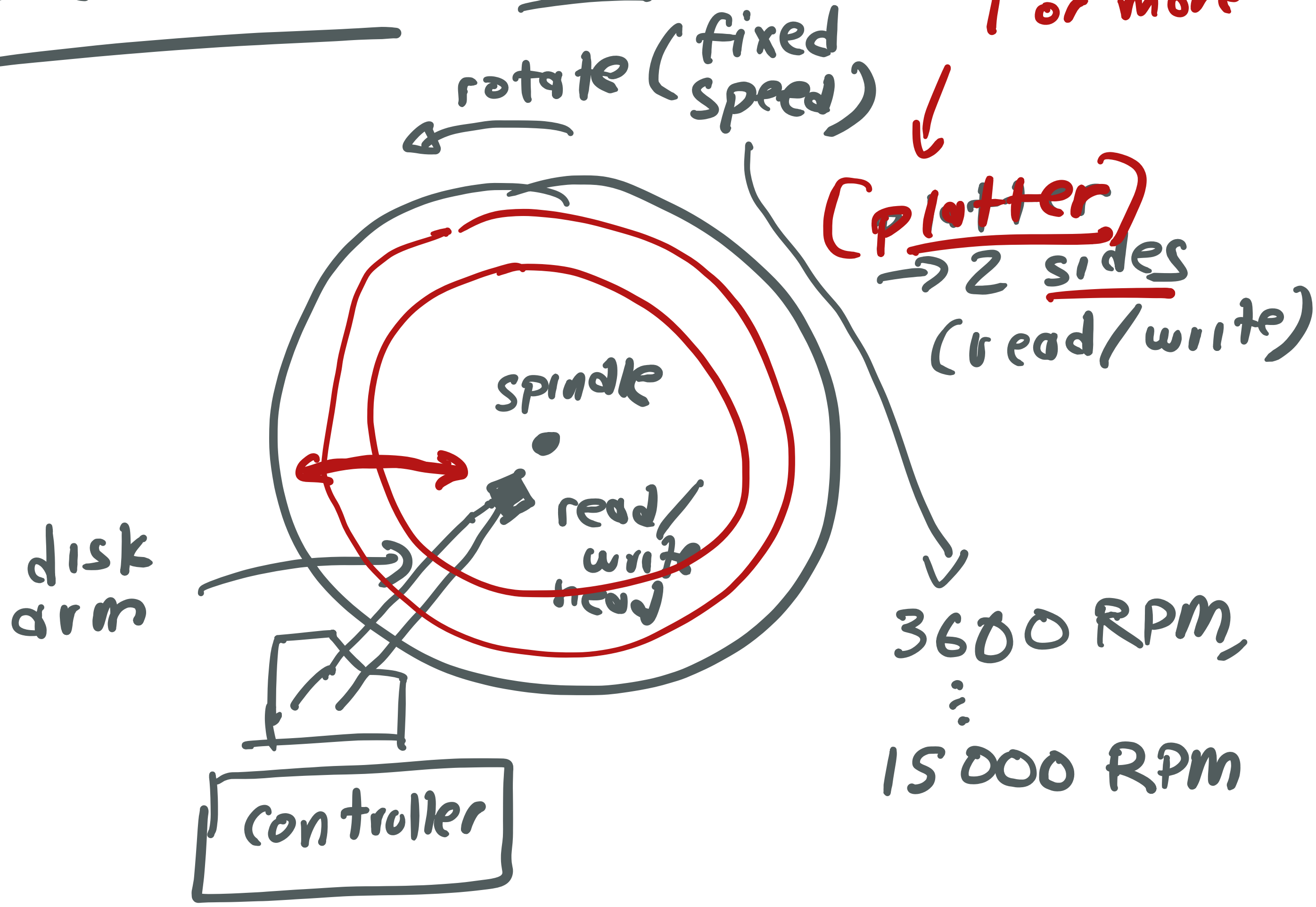
[OS can program the h/w to move data directly between memory/device]

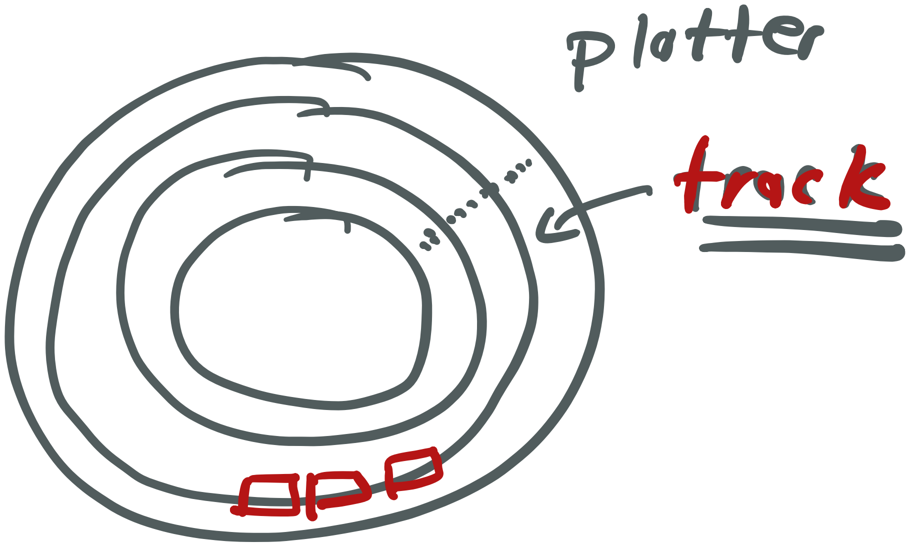




run something else while P1 waits for I/O!

Hard Drive: How?





track: consists of **sectors**
(512 byte)

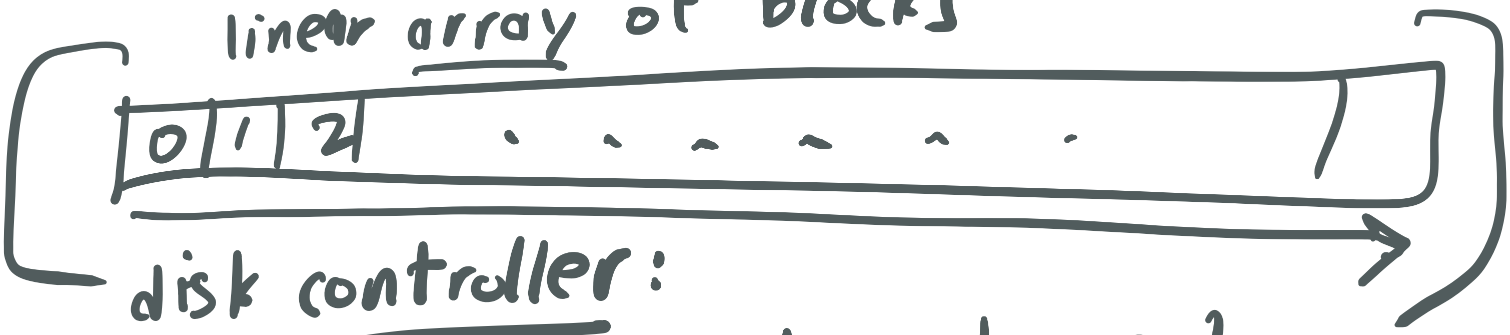
to read or write:

e.g. read a sector

how many sectors

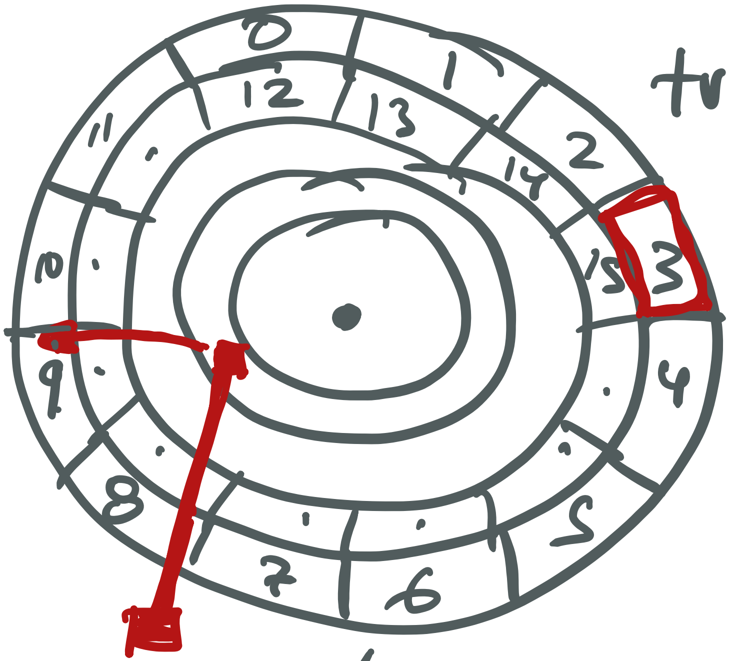
address, (size)

linear array of blocks



disk controller:

=> internal action to read or write



to do an I/O:

=> seek : moves disk arm to correct track

→ wait (rotation) : wait for sector to rotate under head

→ transfer : read/write

Example:

avg seek: ~ 7ms] 10 ms

avg rotate: ~ 3 ms

transfer: ~ 0 ms

~ 200 MB/s

512 b (½ K)

½ KB

throughput: $\frac{512\text{ b}}{10\,\cancel{\text{ms}}} \cdot \frac{1000\,\cancel{\text{ms}}}{\text{seconds}} = \frac{½\text{ MB}}{\text{s}}$

[random I/Os: dominated by seek/rotation]

random I/O is slow

[sequential I/O are "fast"
(or large I/Os)]

disk scheduling: