Operating Systems
I/0 devices

Nipun Batra

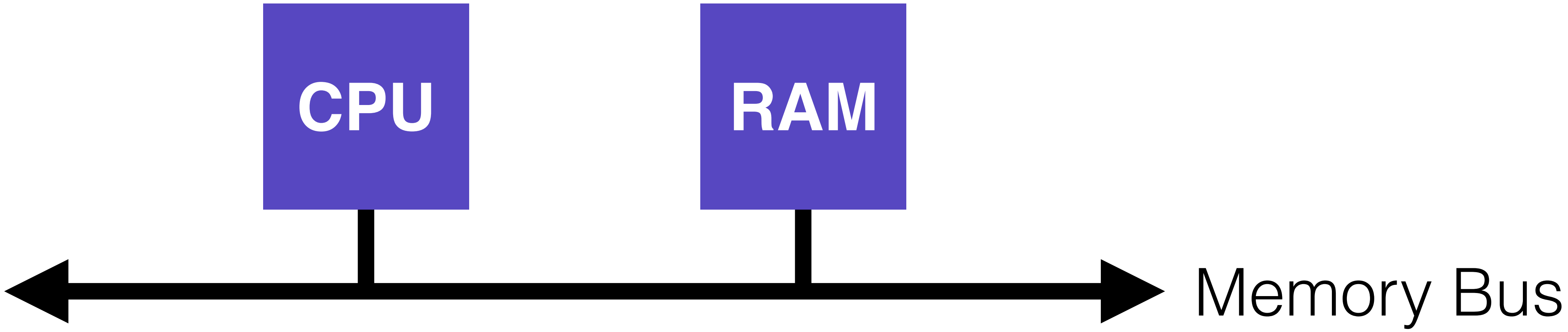
Motivation

What good is a computer without any I/O devices?
- keyboard, display, disks

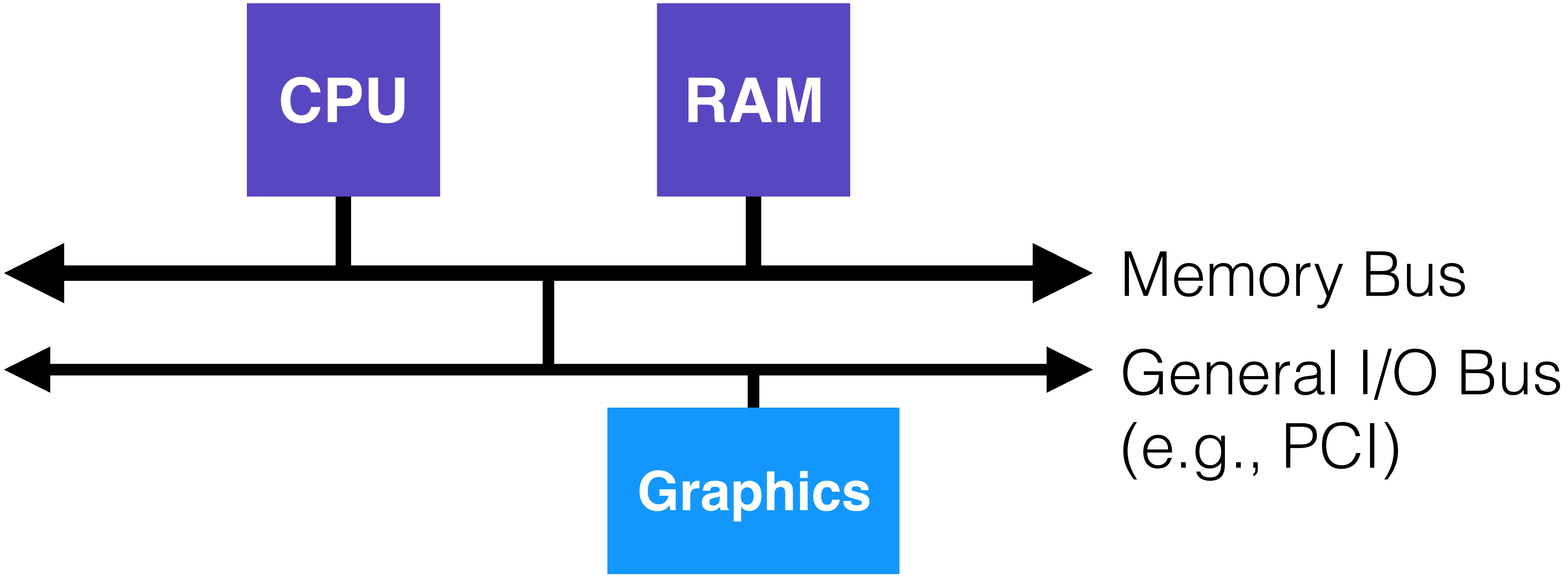
We want:
- **H/W** that will let us plug in different devices
- **OS** that can interact with different combinations

Largely a communication problem…

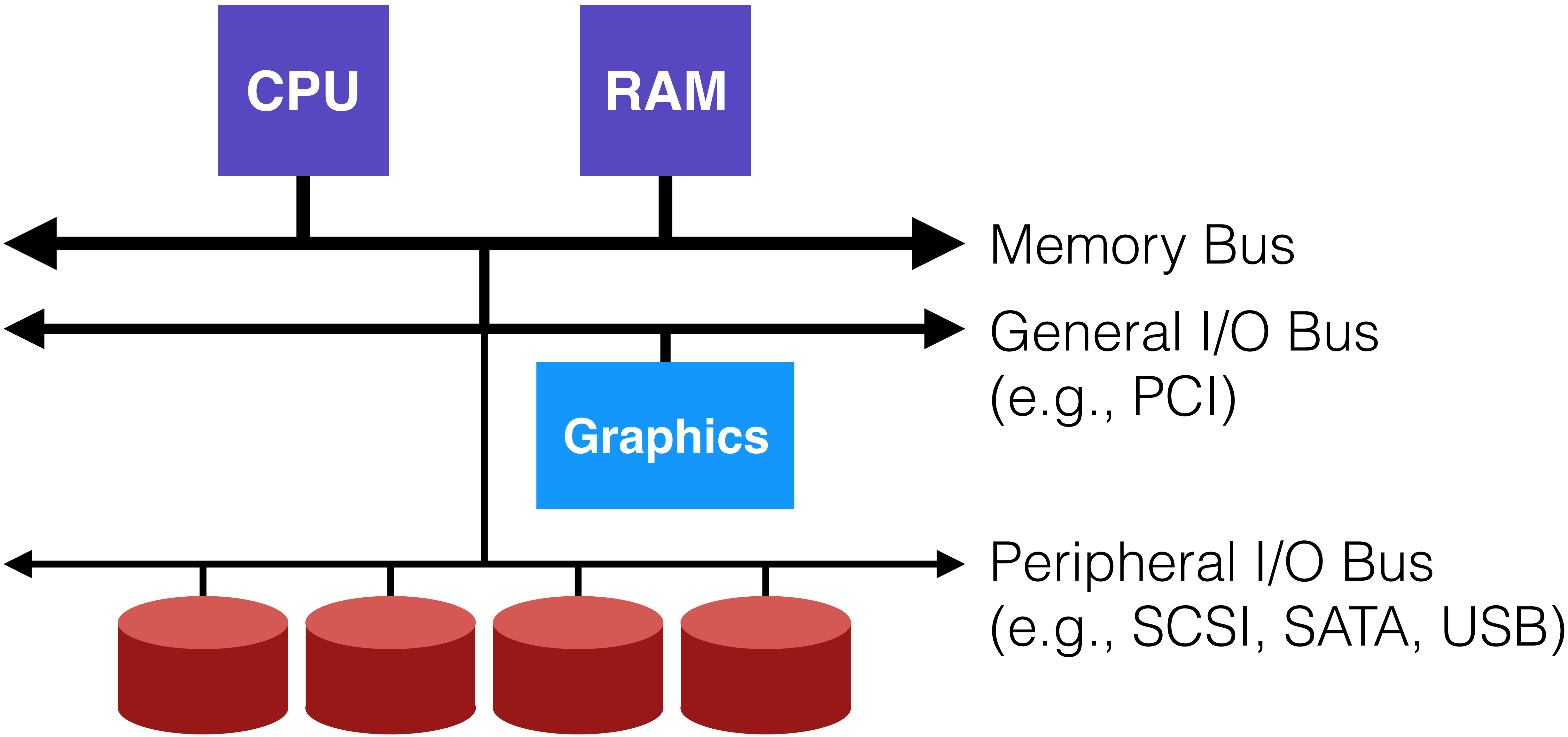
System Architecture



System Architecture



System Architecture



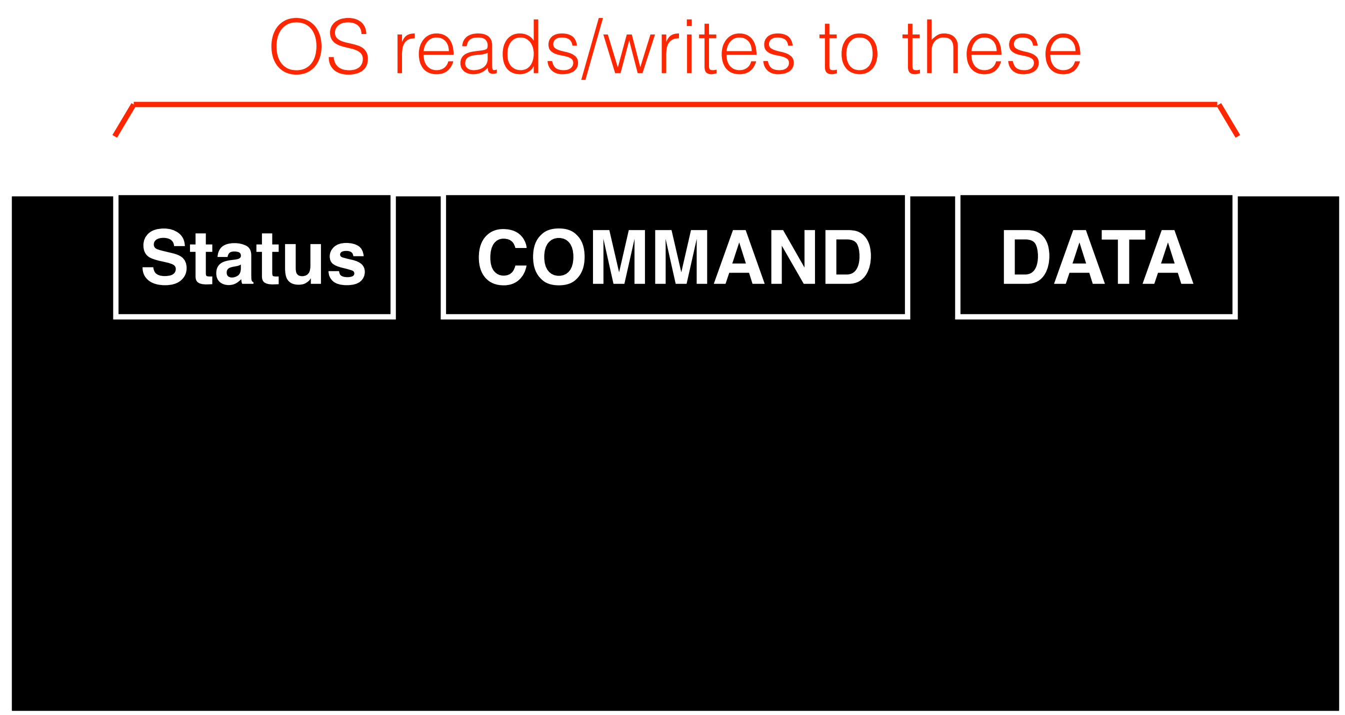
Why use hierarchical buses?

Canonical Device

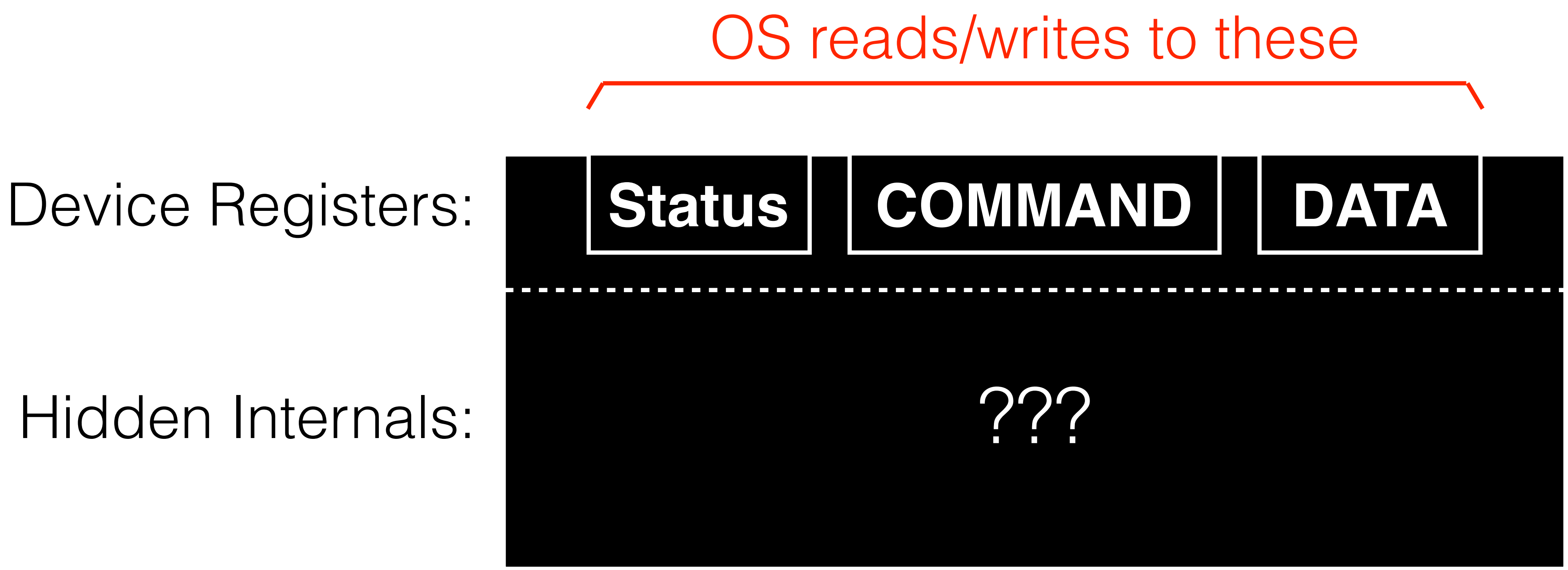
Device Registers:

Canonical Device

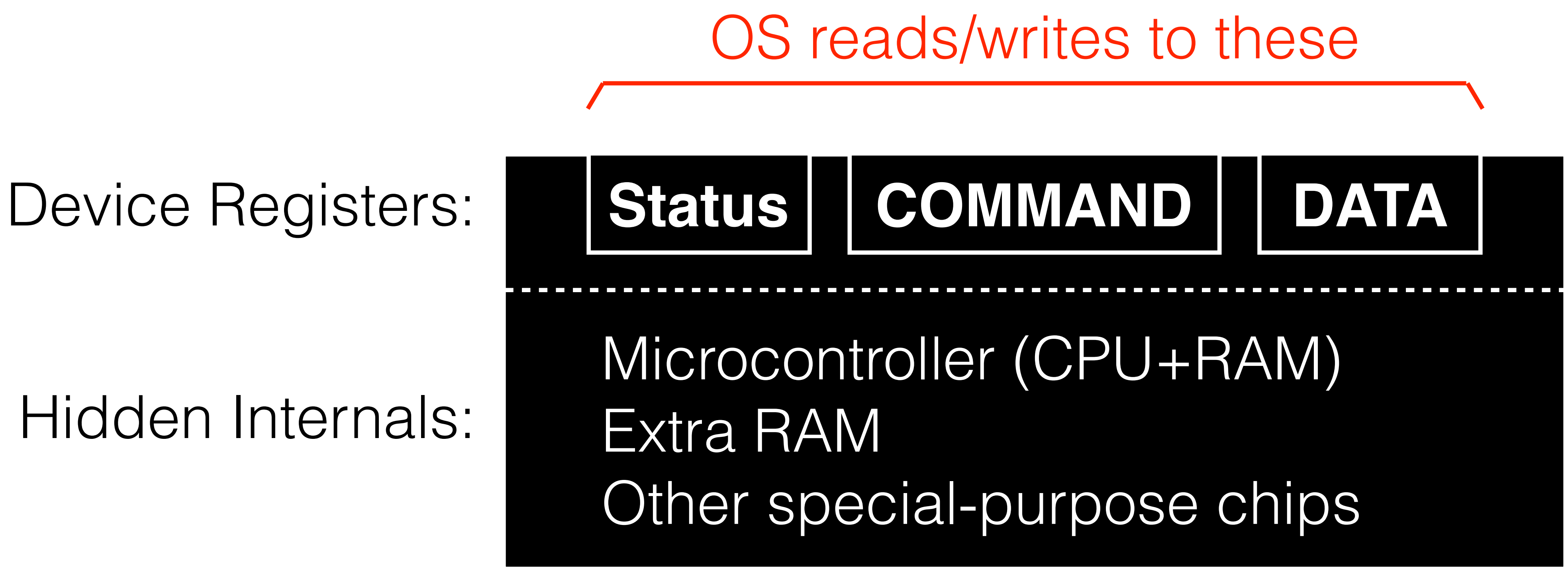
Device Registers:



Canonical Device



Canonical Device



Example Protocol

```
while (STATUS == BUSY)
   ; // spin
Write data to DATA register
Write command to COMMAND register
while (STATUS == BUSY)
   ; // spin
```

Example

CPU:

Disk:

```
while (STATUS == BUSY)                  // 1
   ;
Write data to DATA register             // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
   ;
```

Example

CPU: **A**

Disk: **C**

```
while (STATUS == BUSY)                // 1
   ;
Write data to DATA register           // 2
Write command to COMMAND register     // 3
while (STATUS == BUSY)                // 4
   ;
```

Example

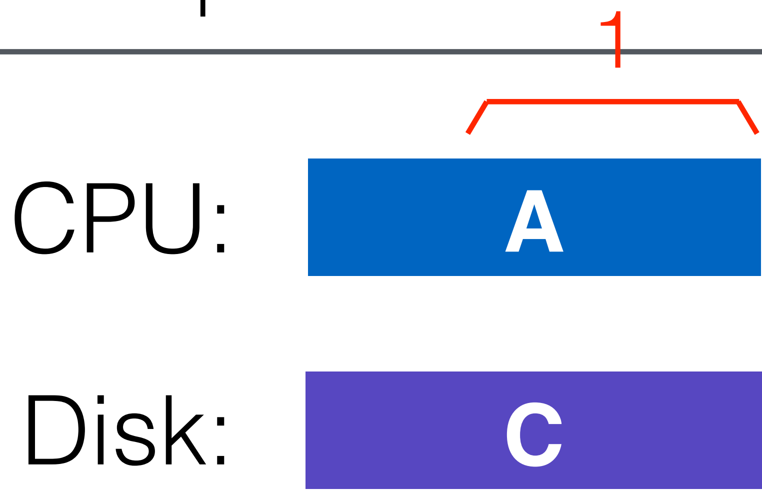
A wants to do I/O

CPU: A

Disk: C

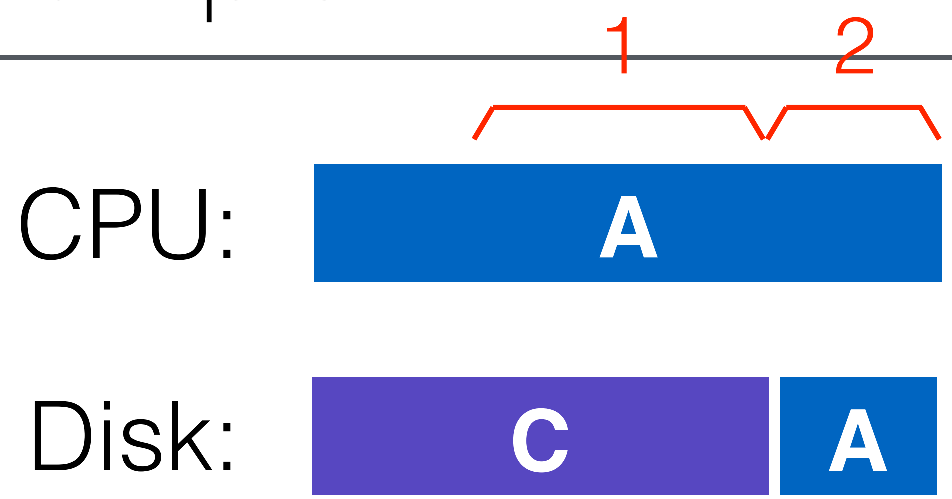
```
while (STATUS == BUSY)                    // 1
   ;
Write data to DATA register               // 2
Write command to COMMAND register         // 3
while (STATUS == BUSY)                    // 4
   ;
```

Example



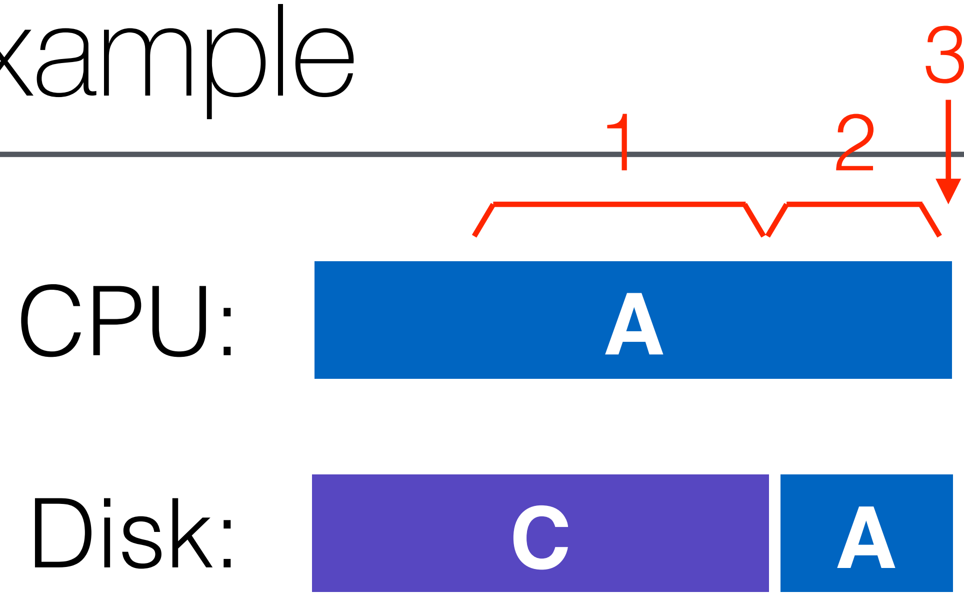

```
while (STATUS == BUSY)                  // 1
   ;
Write data to DATA register             // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
   ;
```

Example



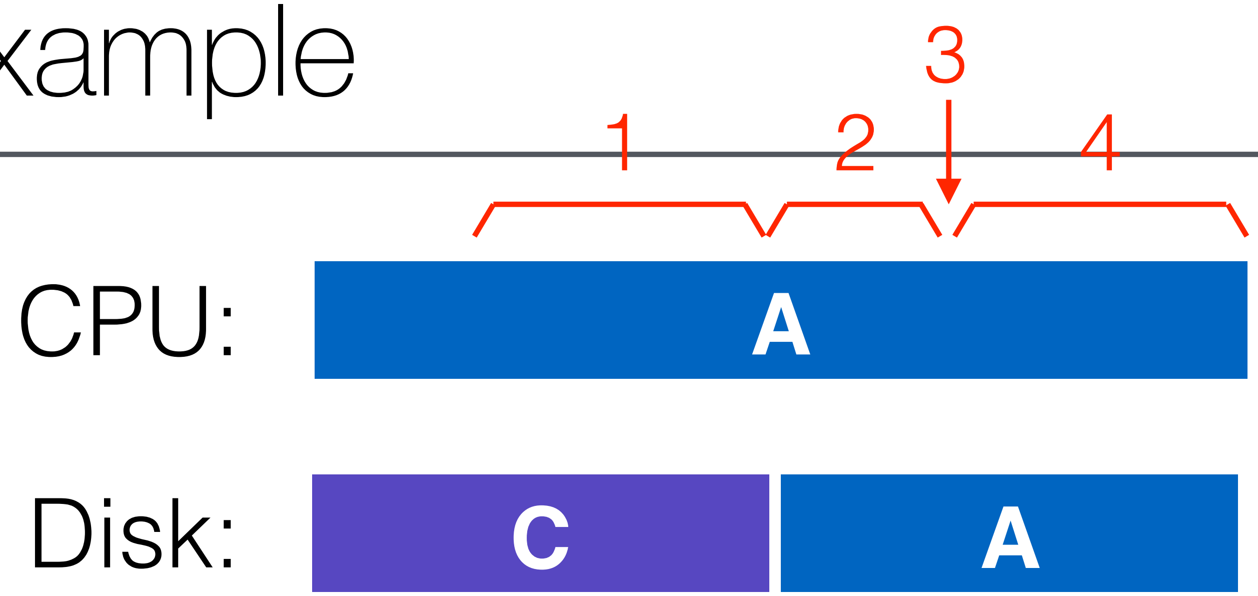

```
while (STATUS == BUSY)                // 1
   ;
Write data to DATA register           // 2
Write command to COMMAND register     // 3
while (STATUS == BUSY)                // 4
   ;
```

Example



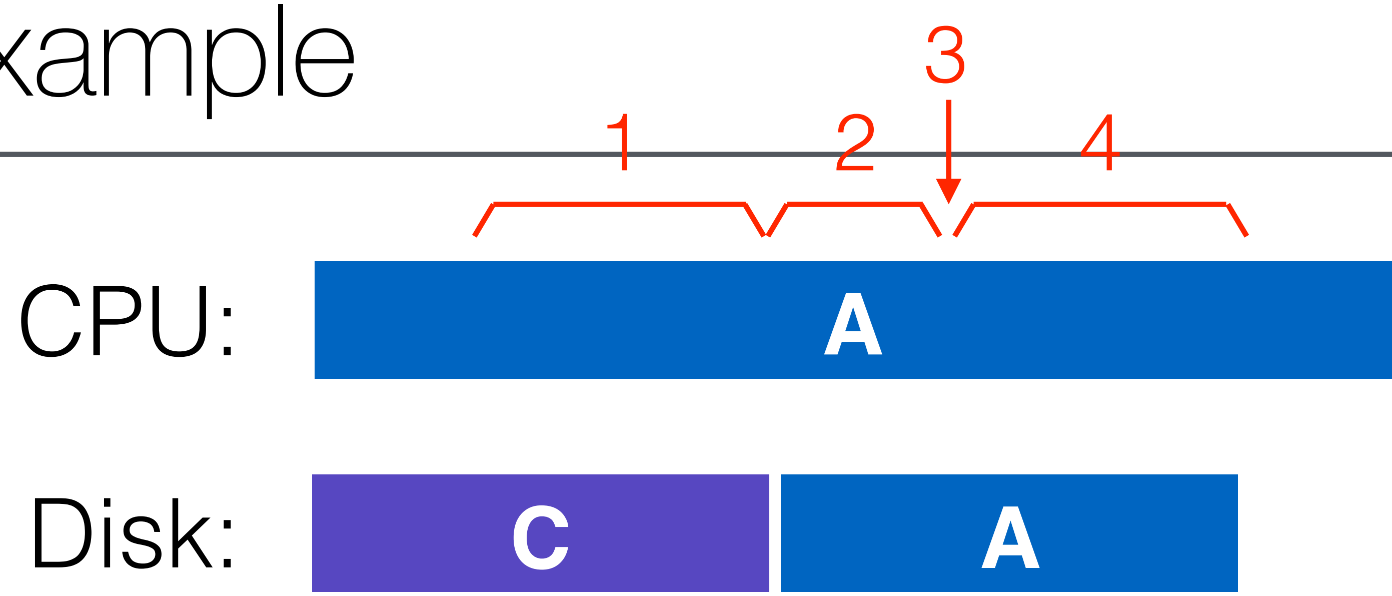

```
while (STATUS == BUSY)                // 1
   ;
Write data to DATA register           // 2
Write command to COMMAND register     // 3
while (STATUS == BUSY)                // 4
   ;
```

Example



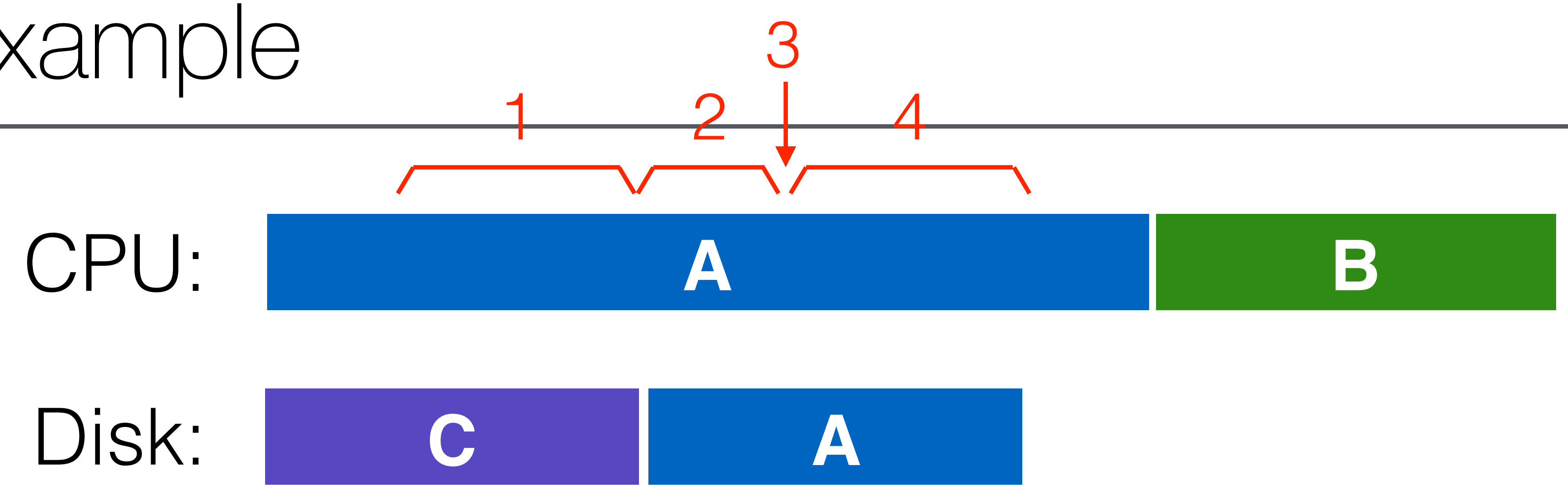

```
while (STATUS == BUSY)                // 1
   ;
Write data to DATA register           // 2
Write command to COMMAND register     // 3
while (STATUS == BUSY)                // 4
   ;
```

Example



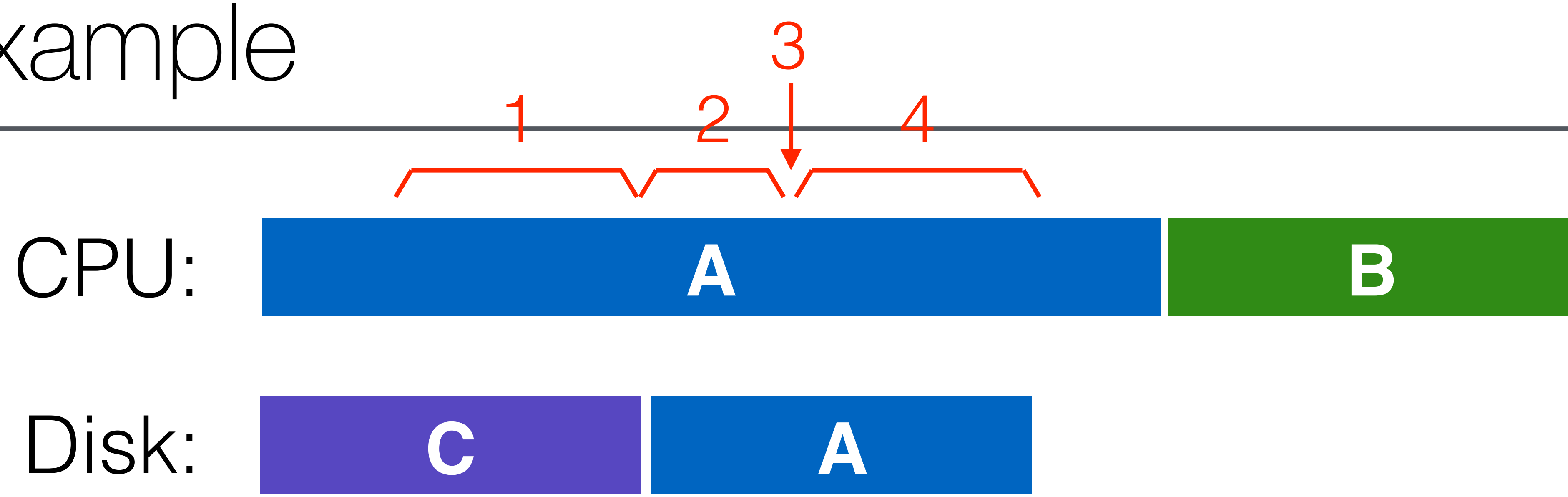

```
while (STATUS == BUSY)                  // 1
   ;
Write data to DATA register             // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
   ;
```

Example




```
while (STATUS == BUSY)                // 1
   ;
Write data to DATA register           // 2
Write command to COMMAND register     // 3
while (STATUS == BUSY)                // 4
   ;
```

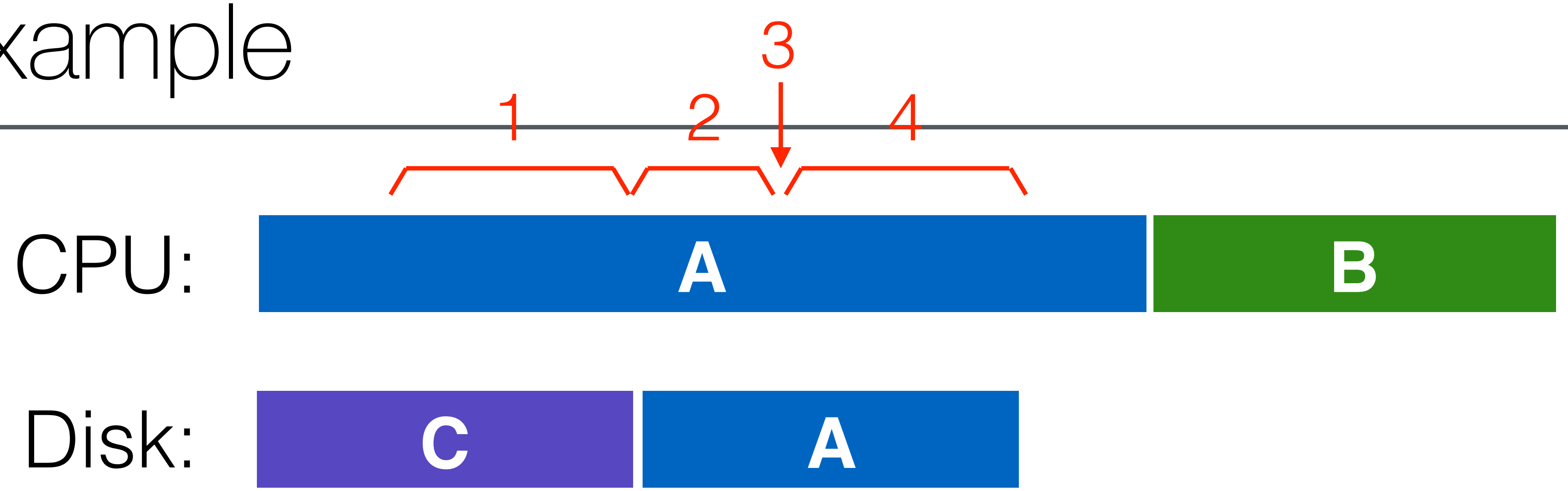
Example




```
while (STATUS == BUSY)                // 1
   ;
Write data to DATA register           // 2
Write command to COMMAND register     // 3
while (STATUS == BUSY)                // 4
   ;
```

How to avoid spinning?

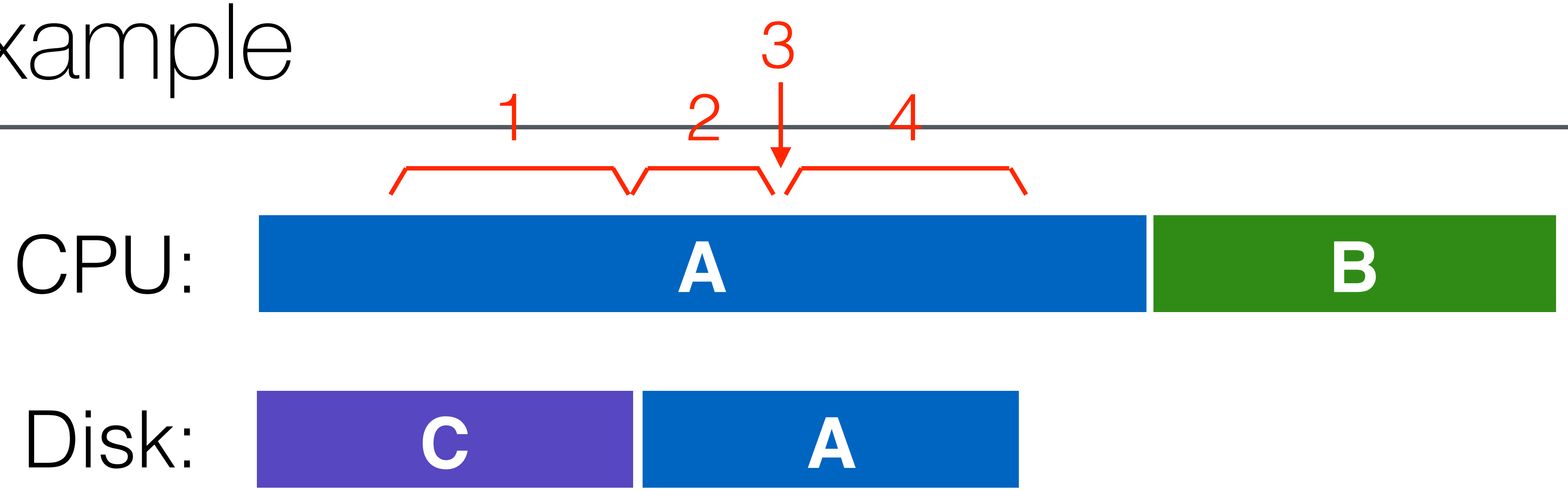
Example




```
while (STATUS == BUSY)                     // 1
   ;
Write data to DATA register                // 2
Write command to COMMAND register          // 3
while (STATUS == BUSY)                     // 4
   ;
```

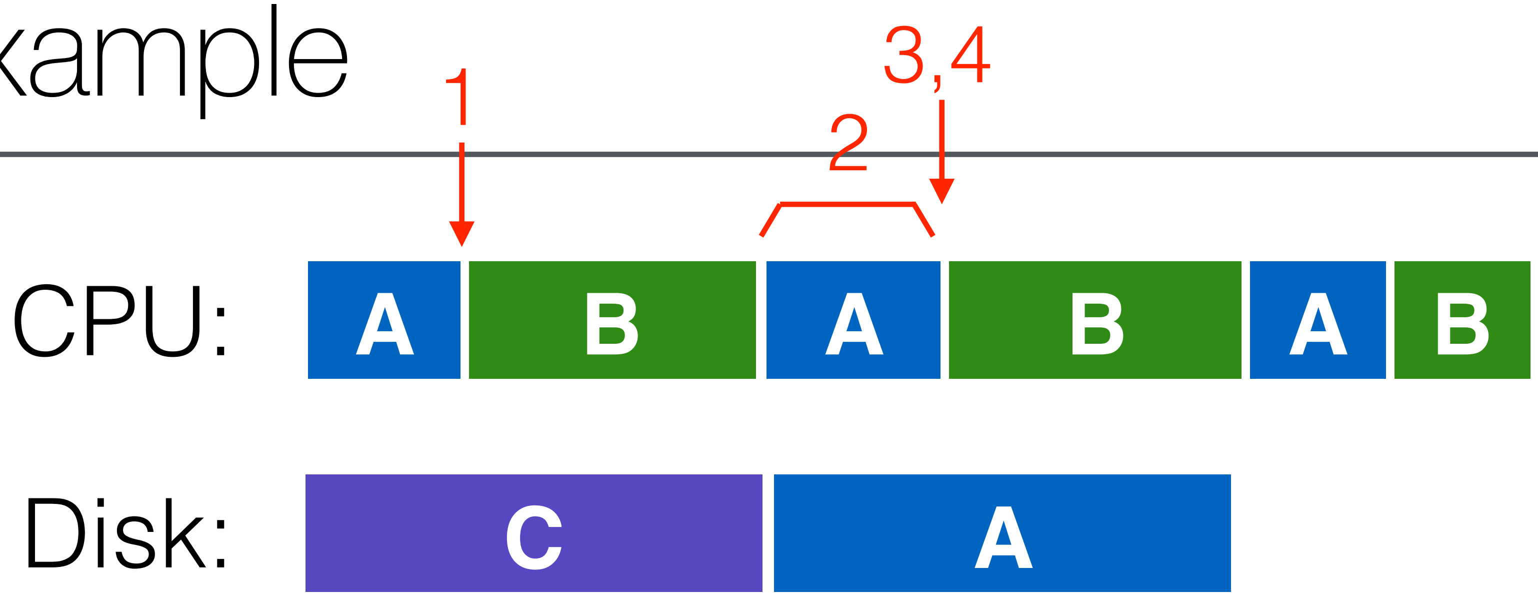
How to avoid spinning?
Interrupts!

Example




```
while (STATUS == BUSY)                 // 1
   wait for interrupt;
Write data to DATA register            // 2
Write command to COMMAND register      // 3
while (STATUS == BUSY)                 // 4
   wait for interrupt;
```

Example




```
while (STATUS == BUSY)                  // 1
   wait for interrupt;
Write data to DATA register             // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
   wait for interrupt;
```

Interrupts vs. Polling

Discuss: are interrupts ever worse?

Interrupts vs. Polling

Discuss: are interrupts ever worse?

Interrupts can sometimes lead to livelock
 - e.g., flood of network packets

Interrupts vs. Polling

Interrupts vs. Polling

- Discuss: are interrupts ever worse?

Interrupts vs. Polling

- Discuss: are interrupts ever worse?
- Interrupts can sometimes lead to livelock

Interrupts vs. Polling

- Discuss: are interrupts ever worse?
- Interrupts can sometimes lead to livelock
    - e.g., flood of network packets

Interrupts vs. Polling

- Discuss: are interrupts ever worse?
- Interrupts can sometimes lead to livelock
    - e.g., flood of network packets
- Techniques:

Interrupts vs. Polling

- Discuss: are interrupts ever worse?
- Interrupts can sometimes lead to livelock
  - e.g., flood of network packets
- Techniques:
  - hybrid approach

Interrupts vs. Polling

- Discuss: are interrupts ever worse?
- Interrupts can sometimes lead to livelock
    - e.g., flood of network packets
- Techniques:
    - hybrid approach
        - Poll for a while, then wait for interrupts

Interrupts vs. Polling

- Discuss: are interrupts ever worse?
- Interrupts can sometimes lead to livelock
    - e.g., flood of network packets
- Techniques:
    - hybrid approach
        - Poll for a while, then wait for interrupts
    - interrupt coalescing

Interrupts vs. Polling

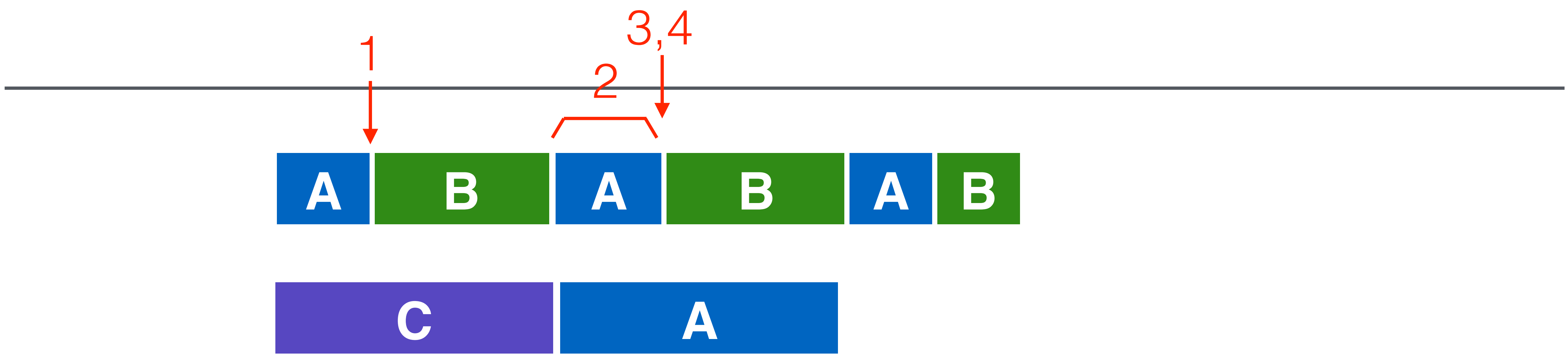
- Discuss: are interrupts ever worse?
- Interrupts can sometimes lead to livelock
    - e.g., flood of network packets
- Techniques:
    - hybrid approach
        - Poll for a while, then wait for interrupts
    - interrupt coalescing
        - Coalesce or combine the delivery of multiple interrupts

Protocol Variants

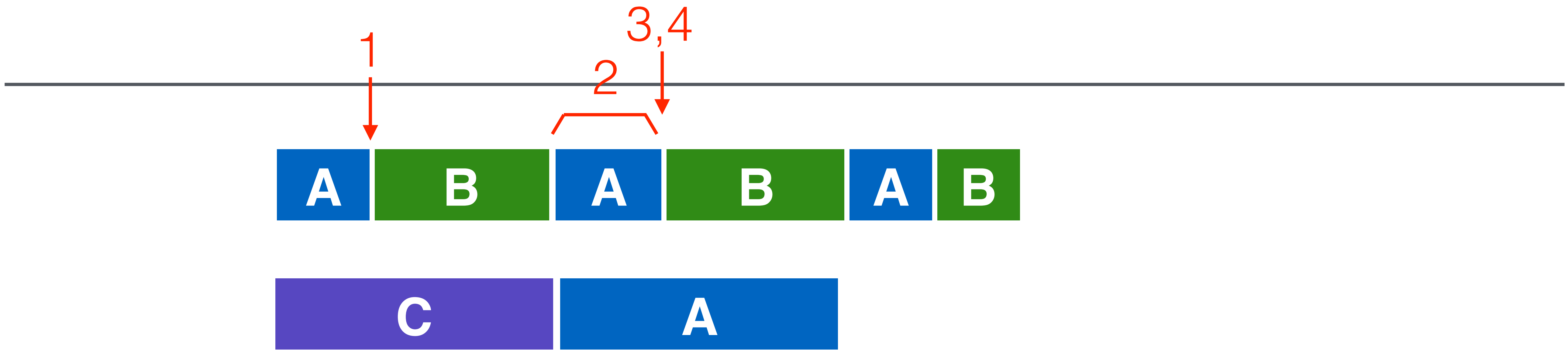
Status checks: polling *vs.* interrupts

Data: PIO *vs.* DMA

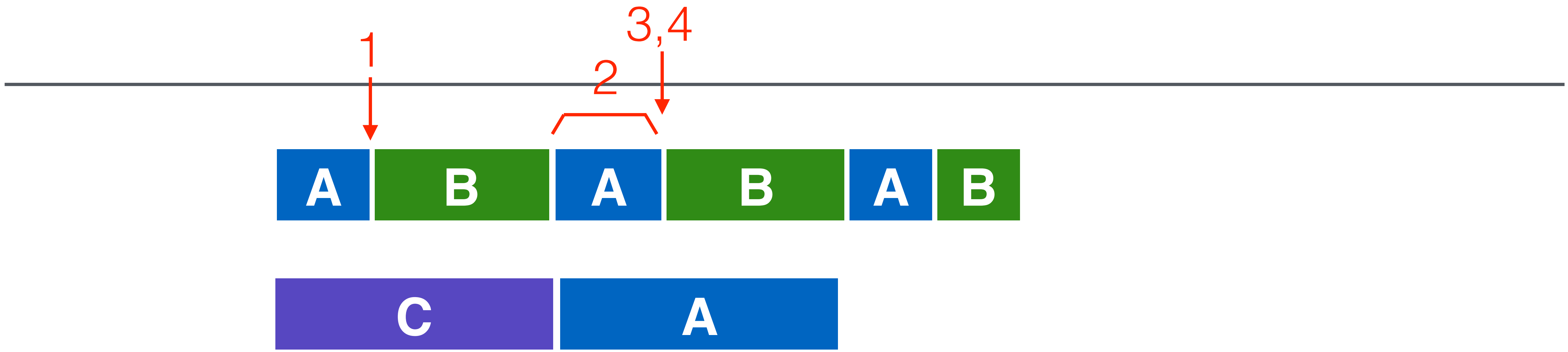
Control: special instructions *vs.* memory-mapped I/O




```
while (STATUS == BUSY)                  // 1
   wait for interrupt;
Write data to DATA register             // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
   wait for interrupt;
```

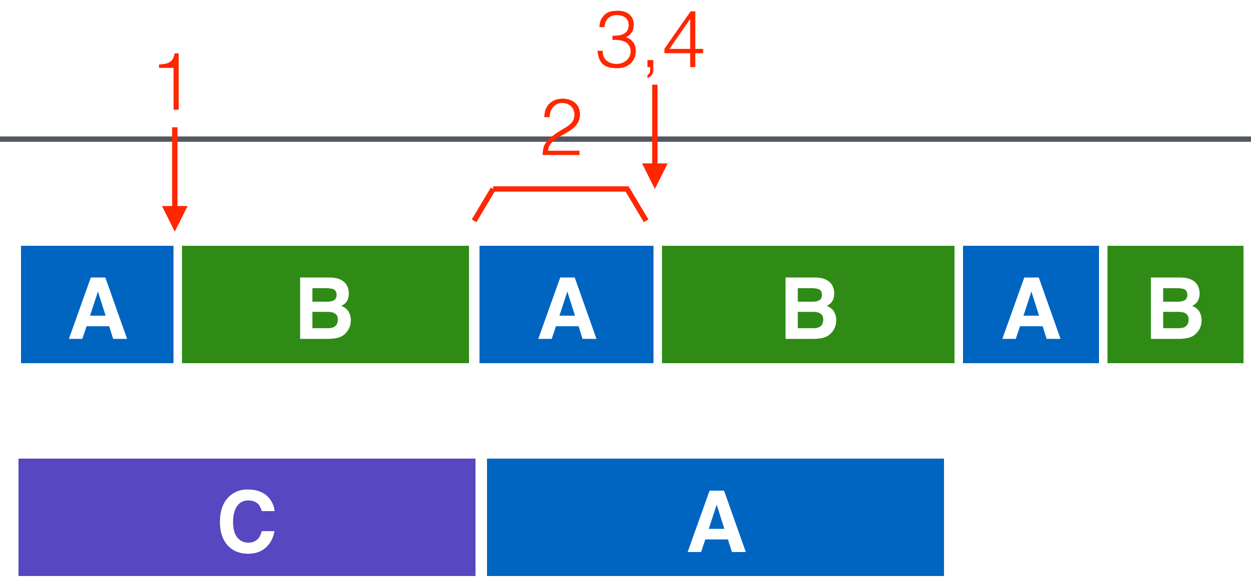



```
while (STATUS == BUSY)                  // 1
   wait for interrupt;
Write data to DATA register             // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
   wait for interrupt;
```




```
while (STATUS == BUSY)              // 1
   wait for interrupt;
Write data to DATA register         // 2
Write command to COMMAND register   // 3
while (STATUS == BUSY)              // 4
   wait for interrupt;
```

What else can we optimize?




```
while (STATUS == BUSY)                  // 1
   wait for interrupt;
Write data to DATA register             // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
   wait for interrupt;
```

What else can we optimize?

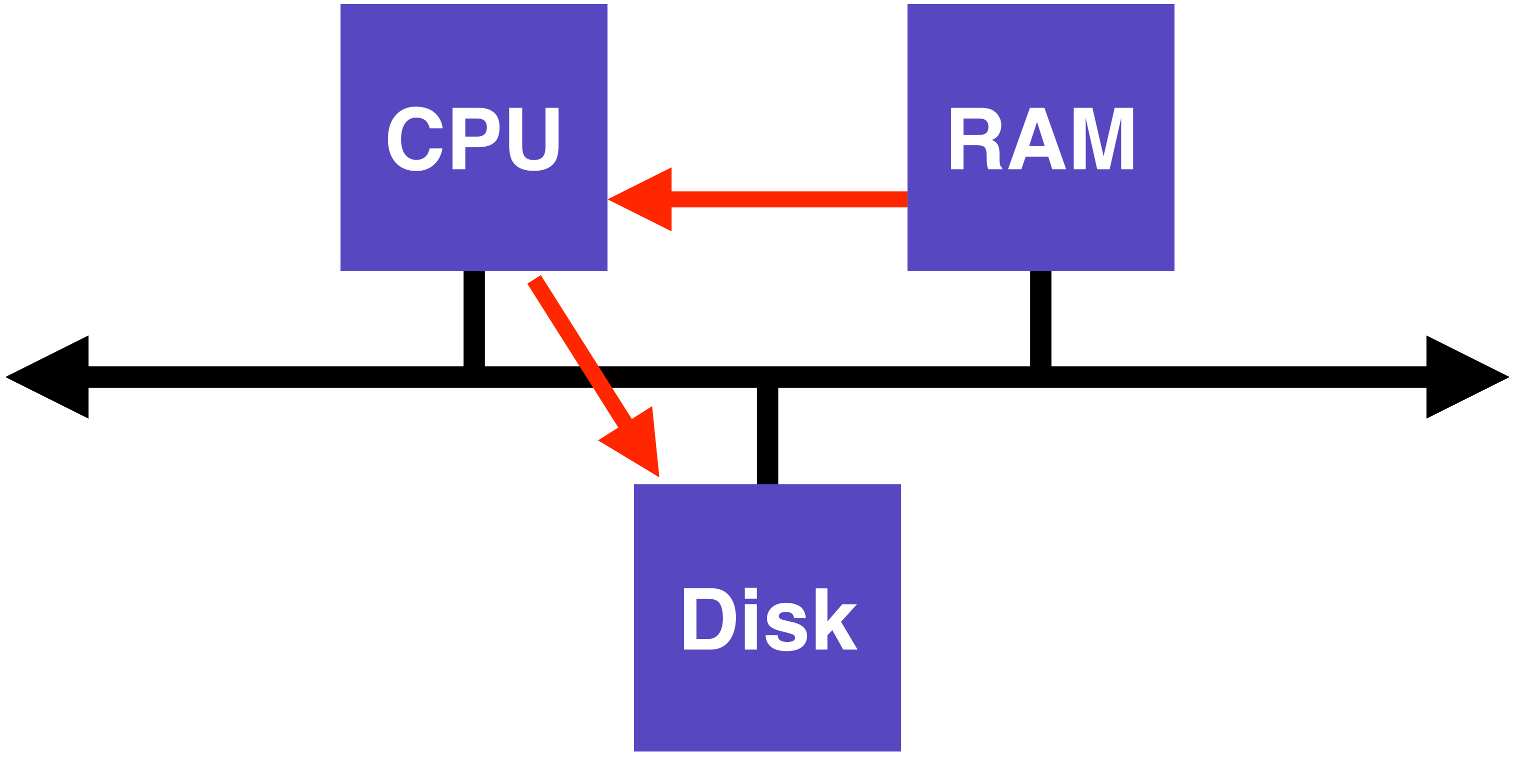
Data transfer!

Programmed I/O vs. Direct Memory Access

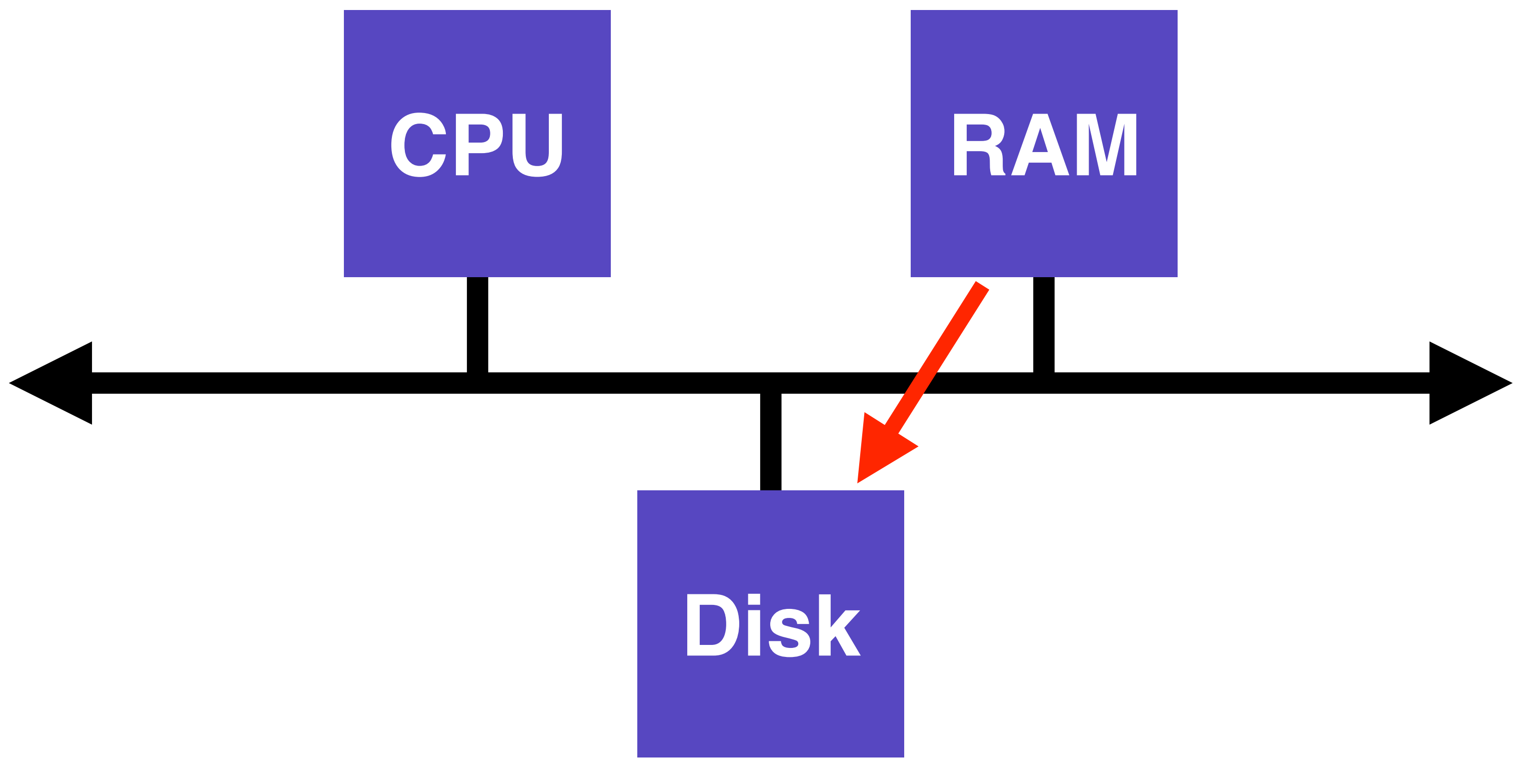
PIO (Programmed I/O):
- CPU directly tells device what data is

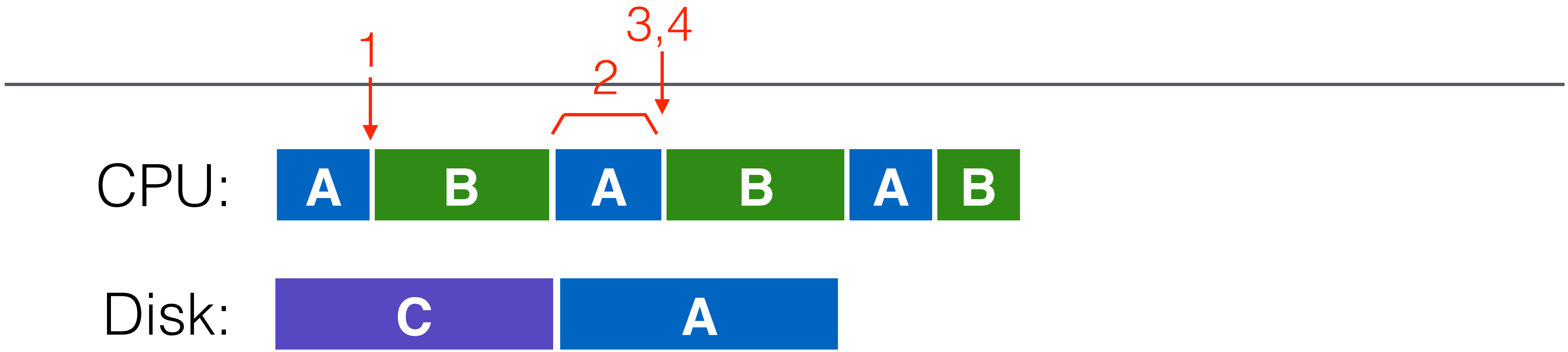
DMA (Direct Memory Access):
- CPU leaves data in memory
- DMA device does copy

PIO Flow

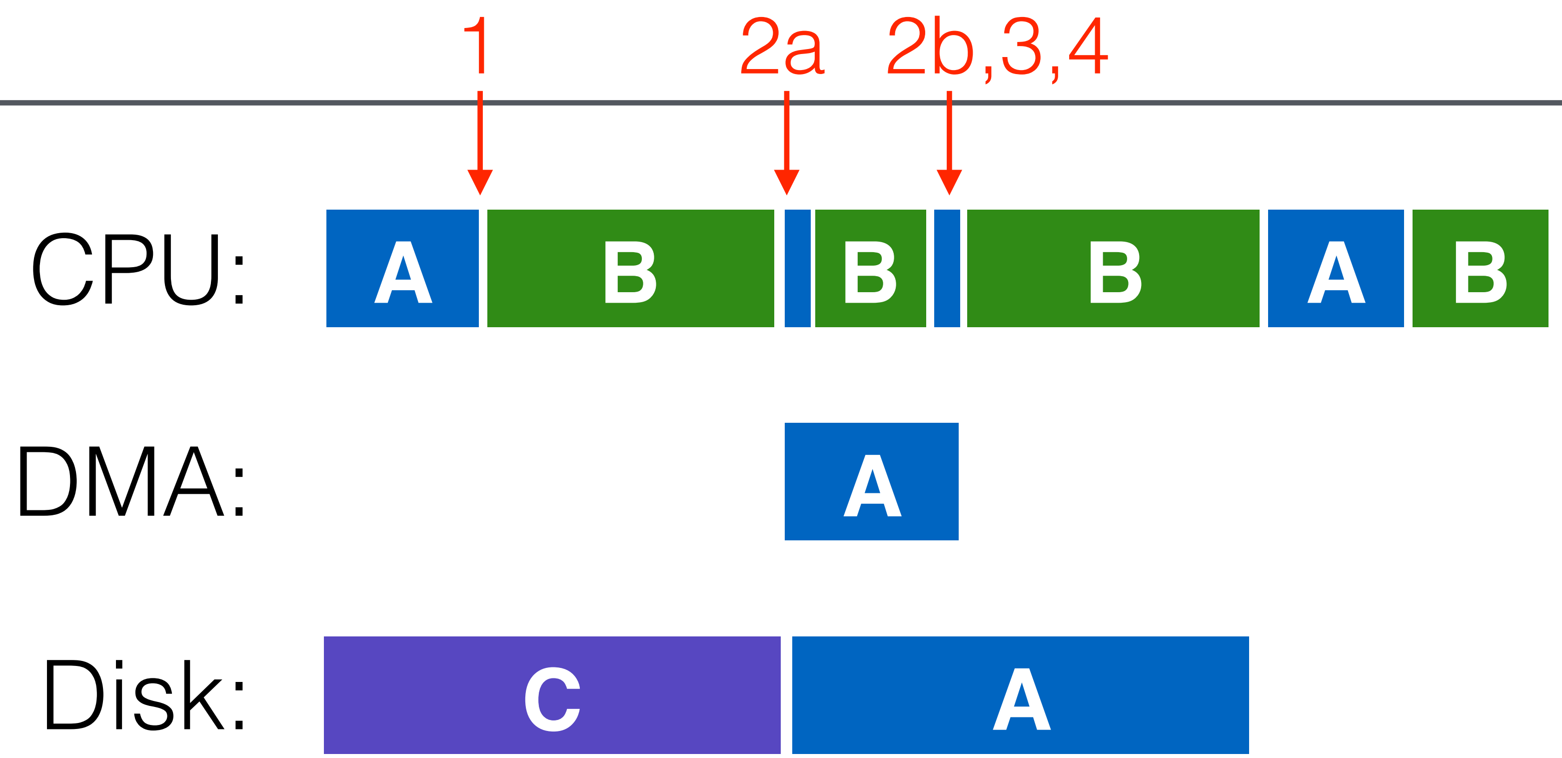


DMA Flow






```
while (STATUS == BUSY)                // 1
   wait for interrupt;
Write data to DATA register           // 2
Write command to COMMAND register     // 3
while (STATUS == BUSY)                // 4
   wait for interrupt;
```



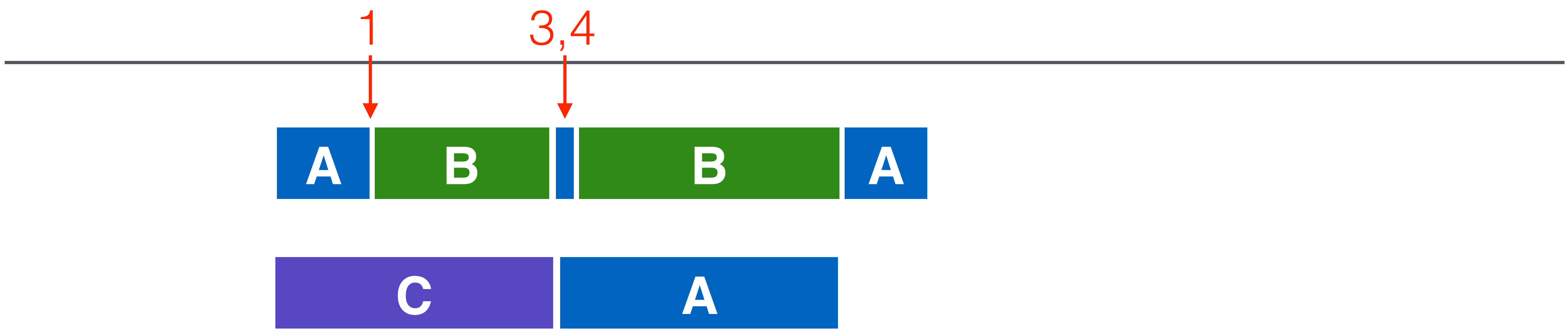

```
while (STATUS == BUSY)                // 1
   wait for interrupt;
initiate DMA transfer                 // 2a
wait for interrupt                    // 2b
Write command to COMMAND register     // 3
while (STATUS == BUSY)                // 4
   wait for interrupt;
```

Protocol Variants

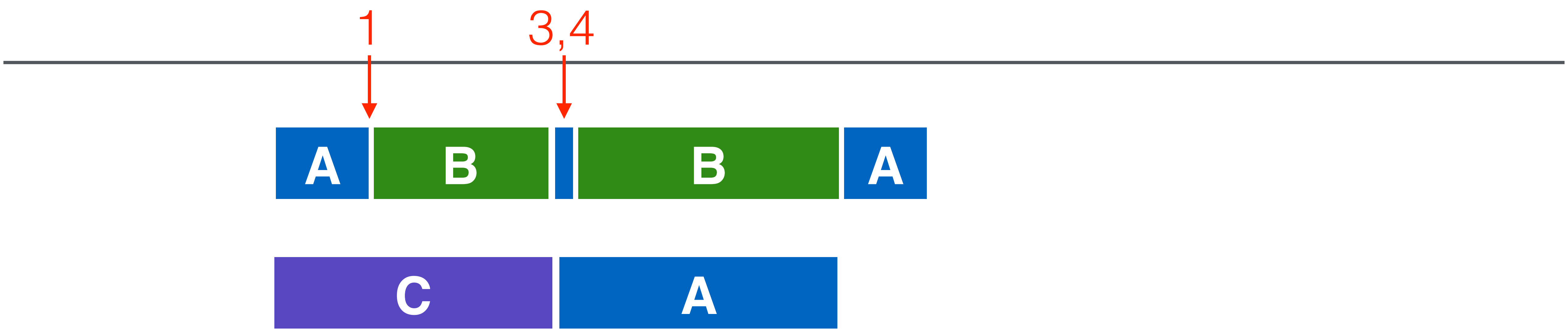
Status checks: polling *vs.* interrupts

Data: PIO *vs.* DMA

Control: special instructions *vs.* memory-mapped I/O




```
while (STATUS == BUSY)                  // 1
   wait for interrupt;
Write data to DATA register             // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
   wait for interrupt;
```




```
while (STATUS == BUSY)                  // 1
  wait for interrupt;
~~Write data to DATA register~~         // 2
Write command to COMMAND register       // 3
while (STATUS == BUSY)                  // 4
  wait for interrupt;
```

How does OS read and write registers?

Special Instructions vs. Mem-Mapped I/O

Special instructions
- each device has a port
- in/out instructions (x86) communicate with device

Memory-Mapped I/O
- H/W maps registers into address space
- loads/stores sent to device

Doesn't matter much (both are used).

Variety is a Challenge

Problem:
- many, many devices
- each has its own protocol

How can we avoid writing a slightly different OS for each H/W combination?

Solution

Encapsulation!

Write driver for each device.

Drivers are **70%** of Linux source code.

Solution

Encapsulation!

Write driver for each device.

Drivers are **70%** of Linux source code.

Encapsulation also enables us to mix-and-match devices, schedulers, and file systems.

Storage Stack

application

file system

scheduler

driver

hard drive

Storage Stack

application

file system

scheduler

driver

hard drive

build common interface
on top of all HDDs

Storage Stack

what about special capabilities?

application

file system

scheduler

driver

hard drive

build common interface on top of all HDDs

Hard-disk Basic Interface

Hard-disk Basic Interface

Disk has a sector-addressable address space

Hard-disk Basic Interface

Disk has a sector-addressable address space
(so a disk is like an array of sectors).

Hard-disk Basic Interface

Disk has a sector-addressable address space (so a disk is like an array of sectors).

Hard-disk Basic Interface

Disk has a sector-addressable address space
(so a disk is like an array of sectors).

Sectors are typically 512 bytes or 4096 bytes.

Hard-disk Basic Interface

Disk has a sector-addressable address space (so a disk is like an array of sectors).

Sectors are typically 512 bytes or 4096 bytes.

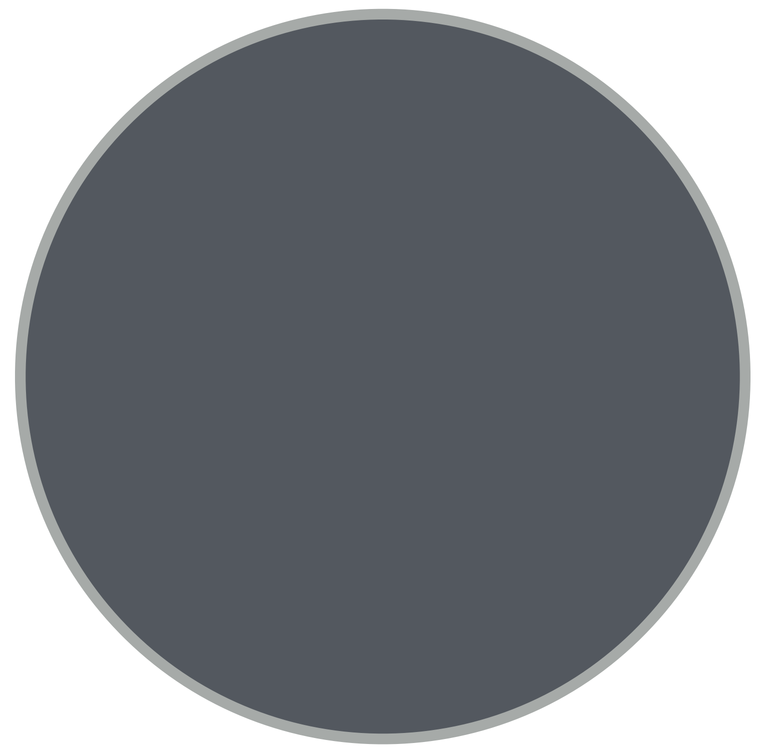
Hard-disk Basic Interface

Disk has a sector-addressable address space
(so a disk is like an array of sectors).

Sectors are typically 512 bytes or 4096 bytes.

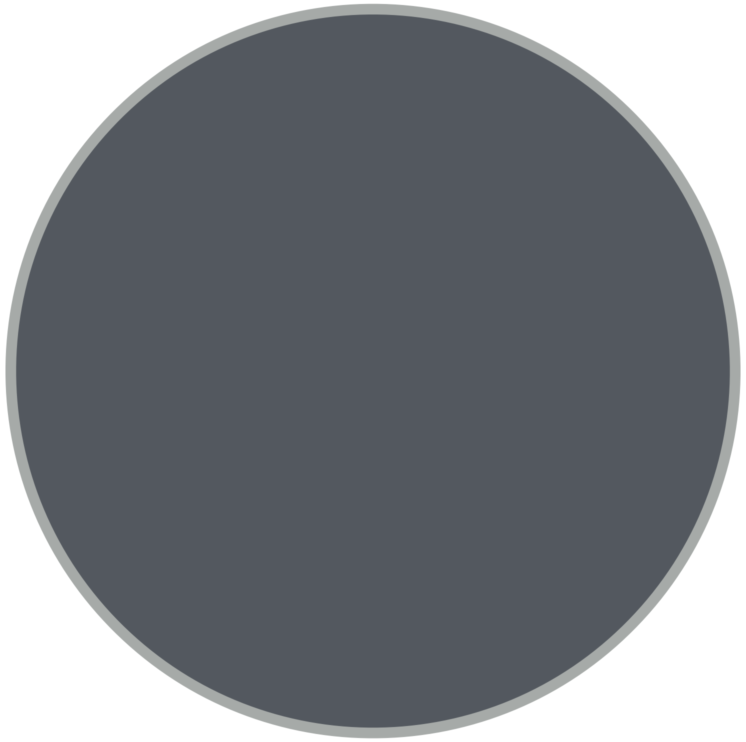
Main operations: reads + writes to sectors (blocks).

Hard-disk Basic Interface



Hard-disk Basic Interface

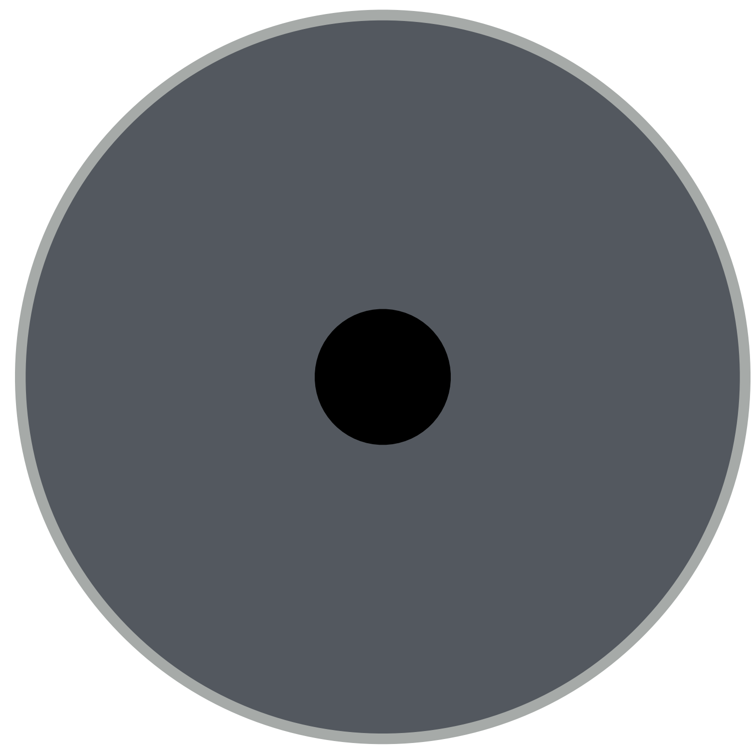
Platter



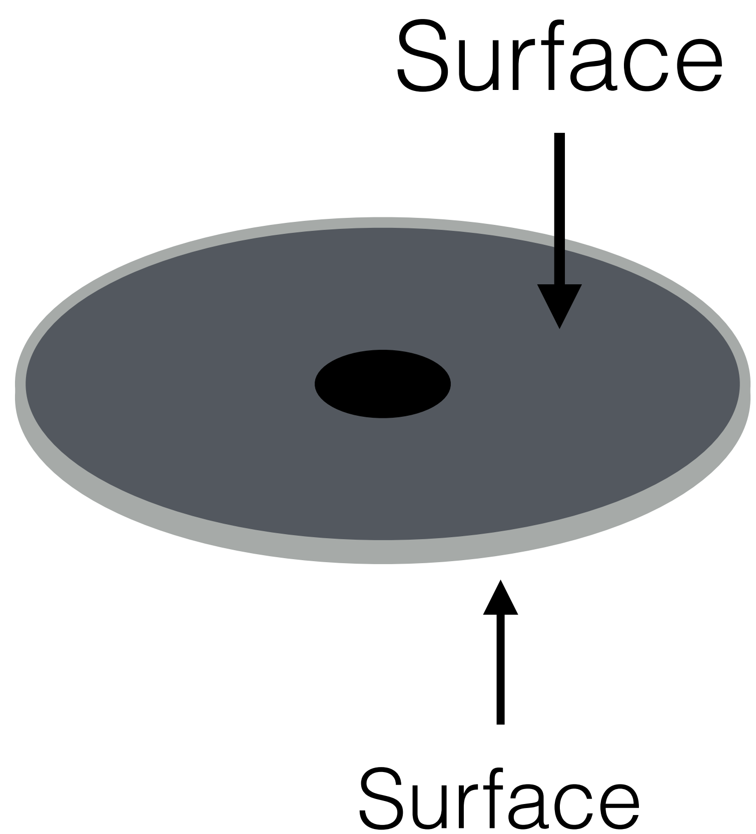
Platter is covered with a magnetic film.

Hard-disk Basic Interface

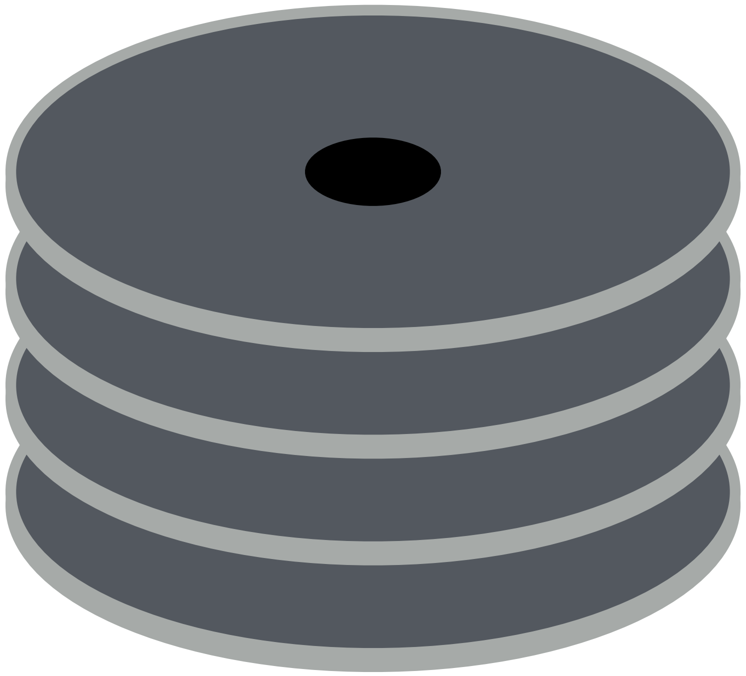
Spindle



Hard-disk Basic Interface

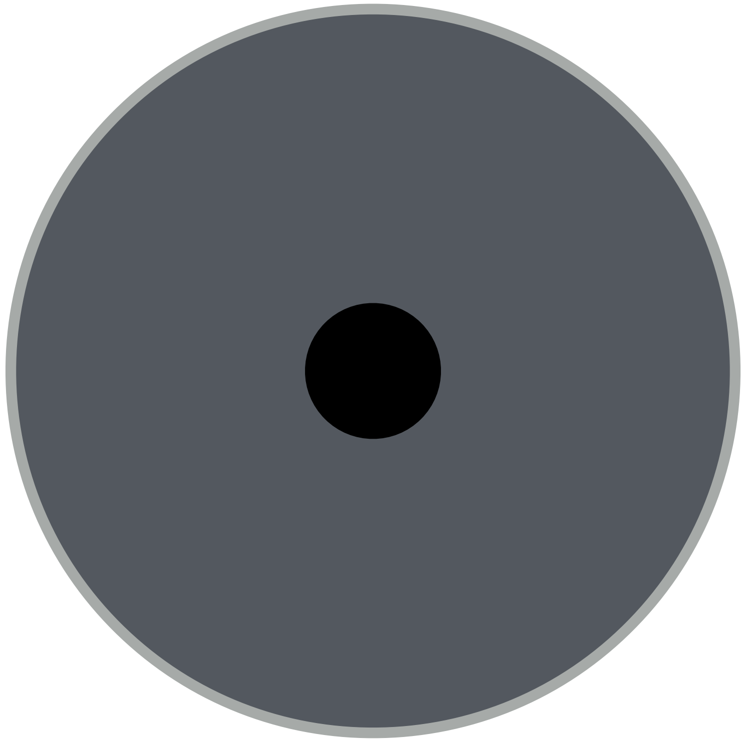


Hard-disk Basic Interface

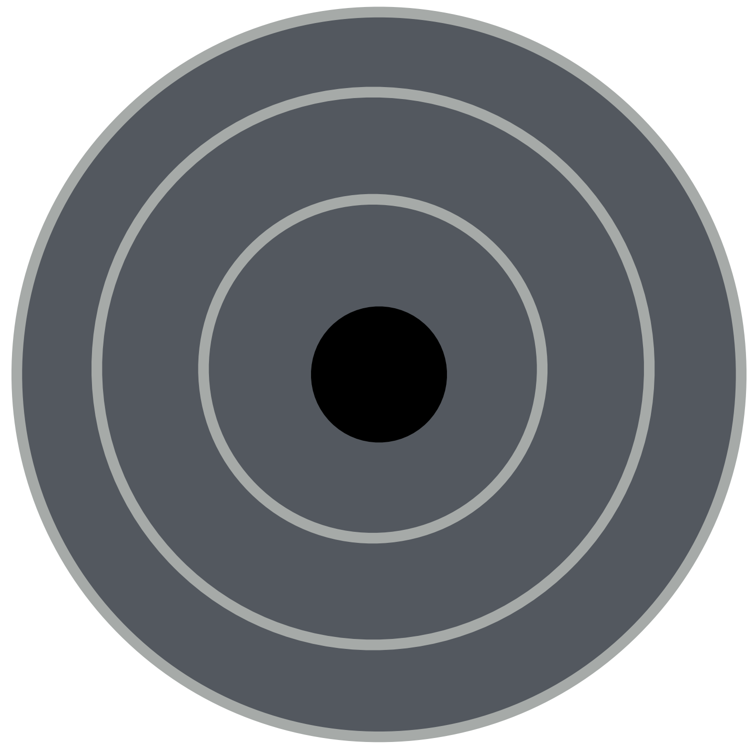


Many platters may be bound to the spindle.

Hard-disk Basic Interface

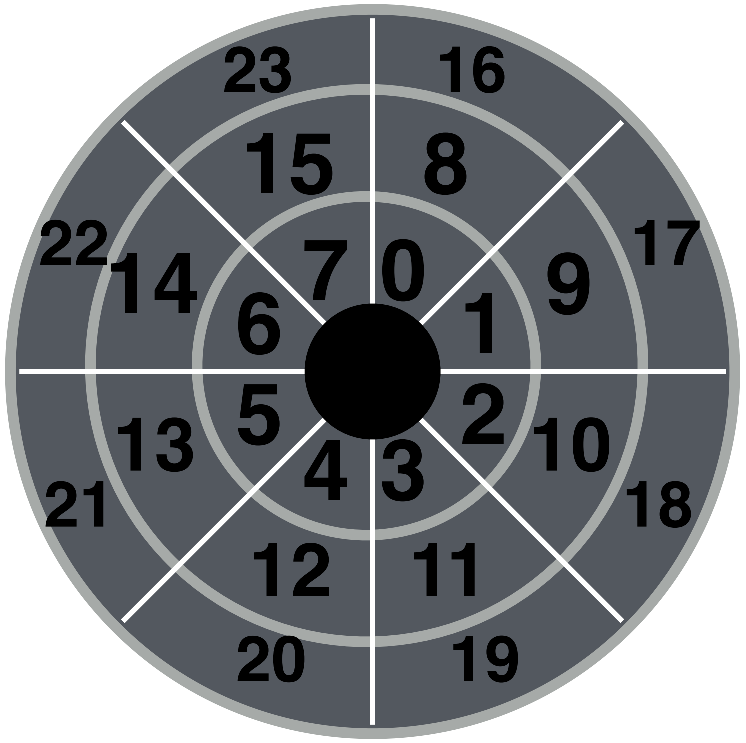


Hard-disk Basic Interface



Each surface is divided into rings called <u>tracks</u>.
A stack of tracks (across platters) is called a <u>cylinder</u>.

Hard-disk Basic Interface

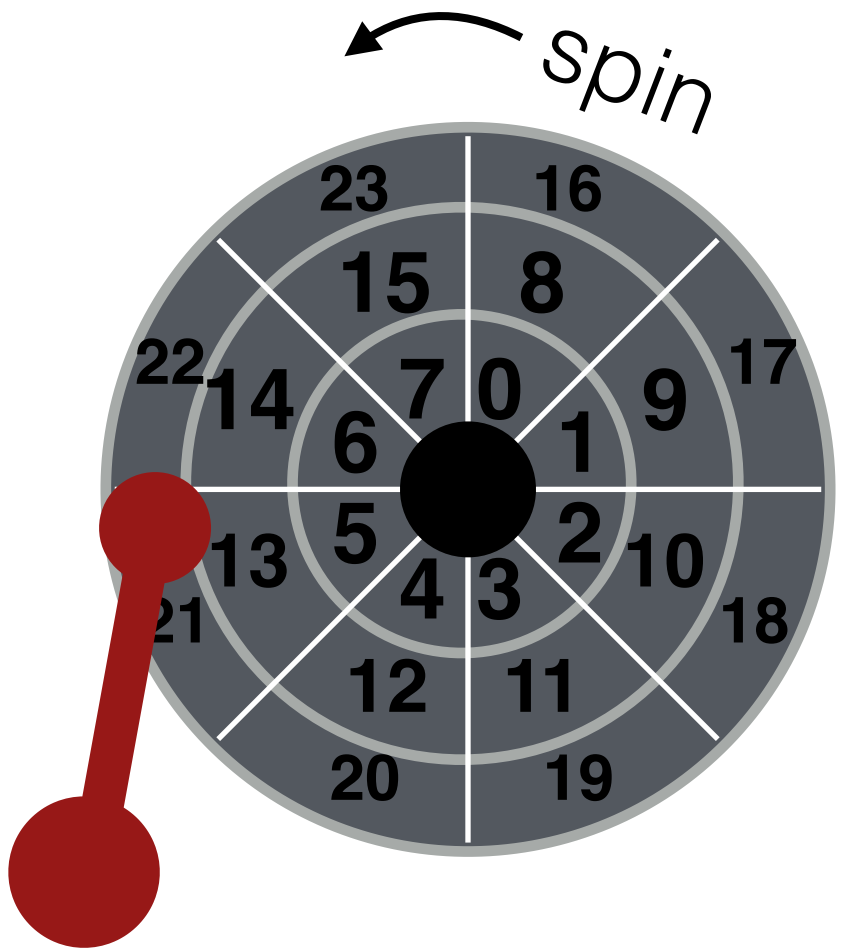


The tracks are divided into numbered sectors.

Hard-disk Basic Interface

<u>Heads</u> on a moving <u>arm</u> can read from each surface.

Hard-disk Basic Interface

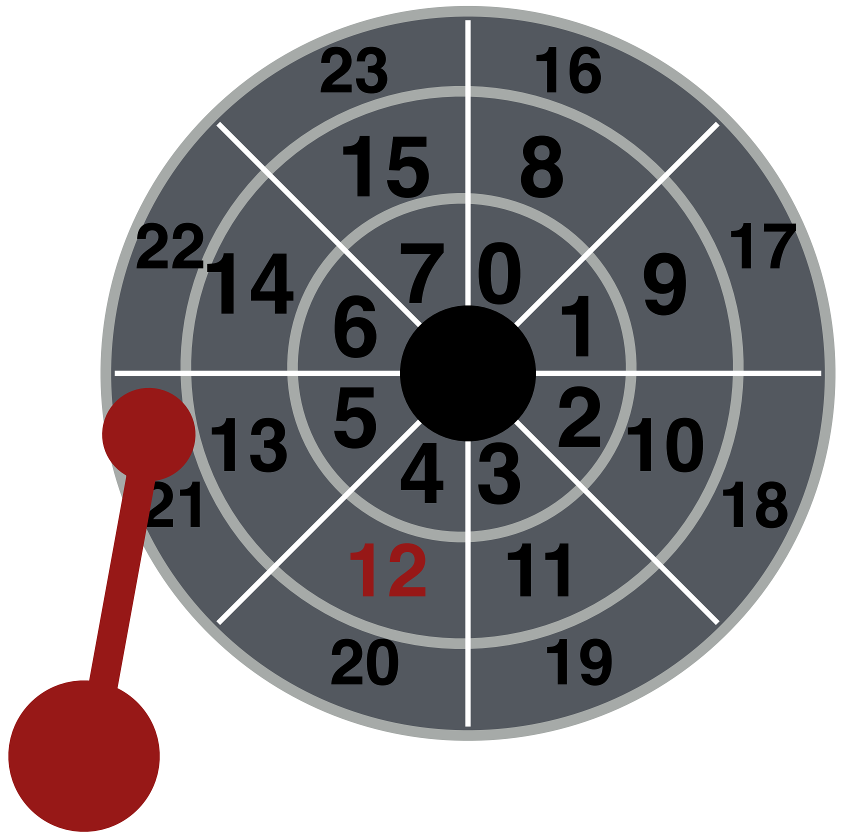


Spindle/platters rapidly spin.

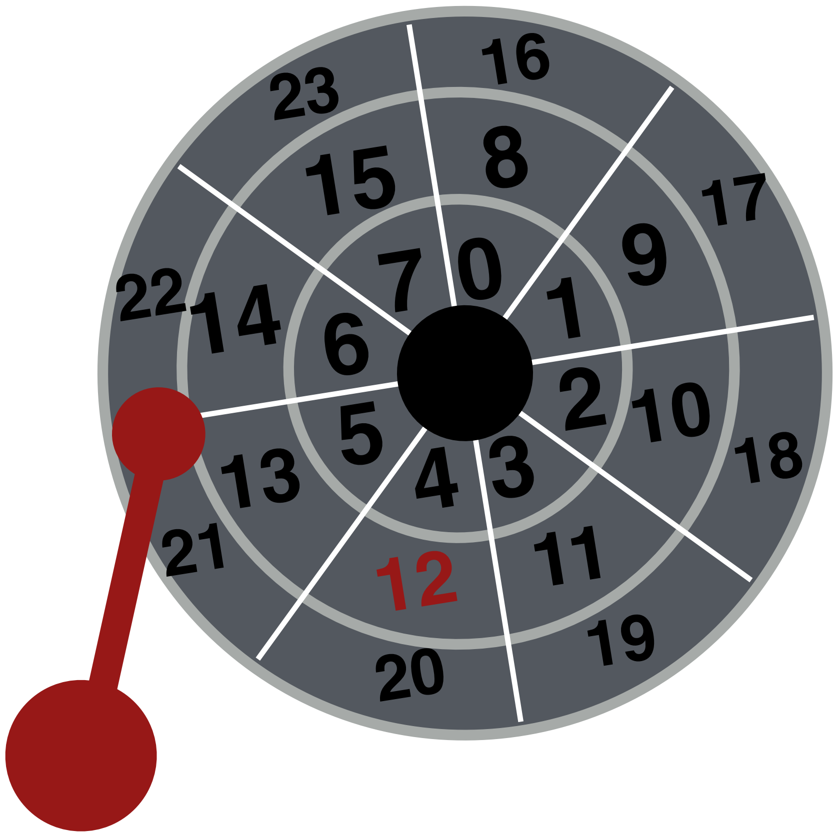
Don't try this at home!

http://youtu.be/9eMWG3fwiEU?t=30s

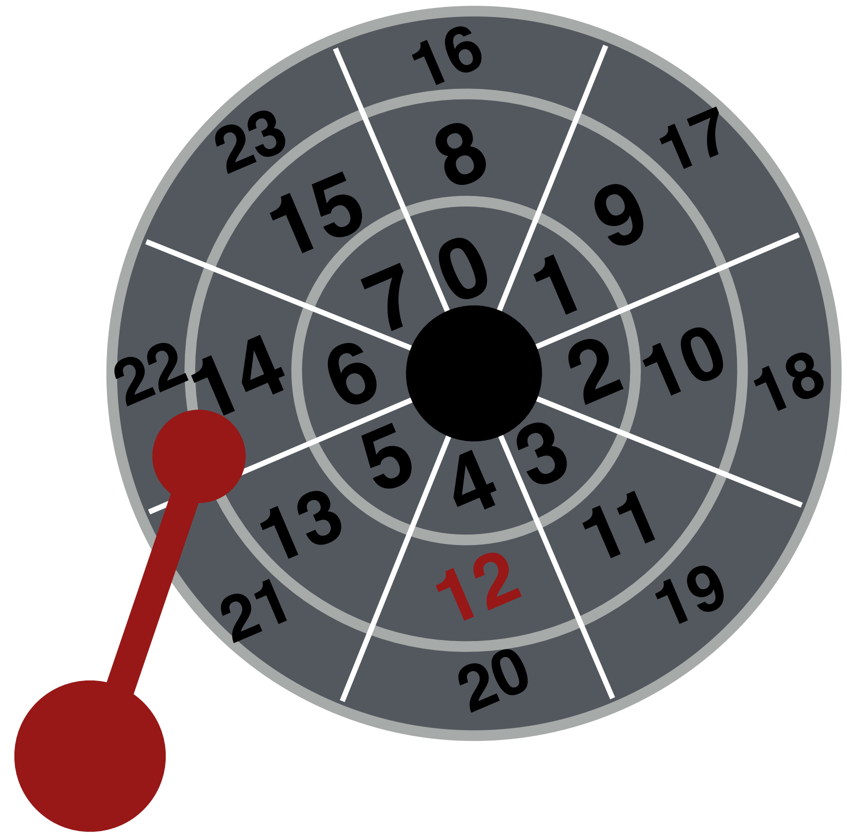
Let's Read 12!



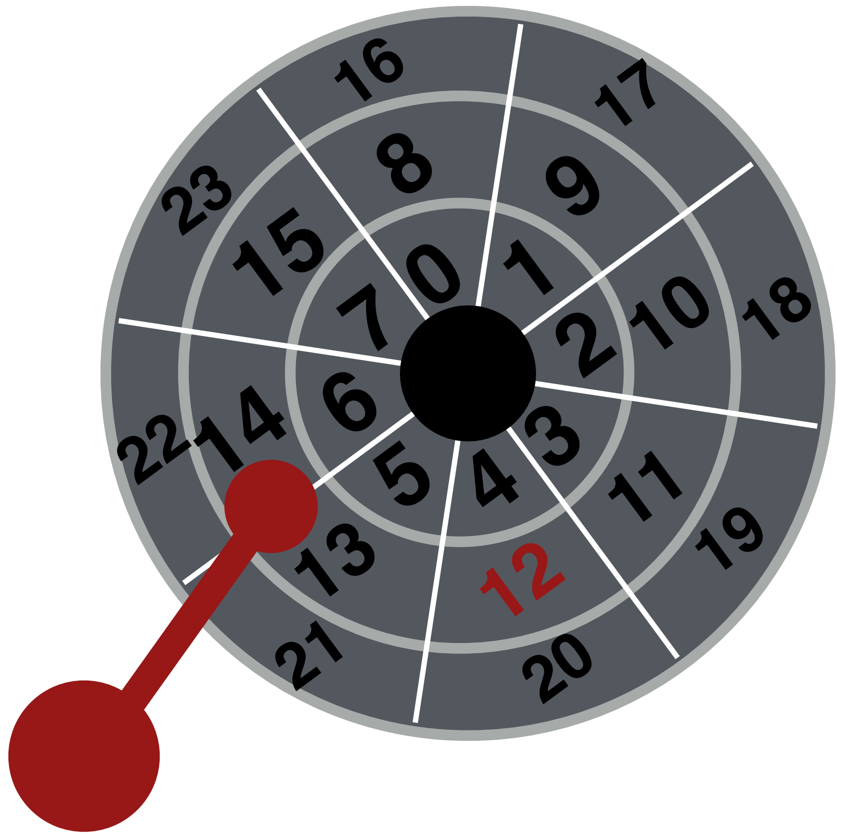
Seek to right track.



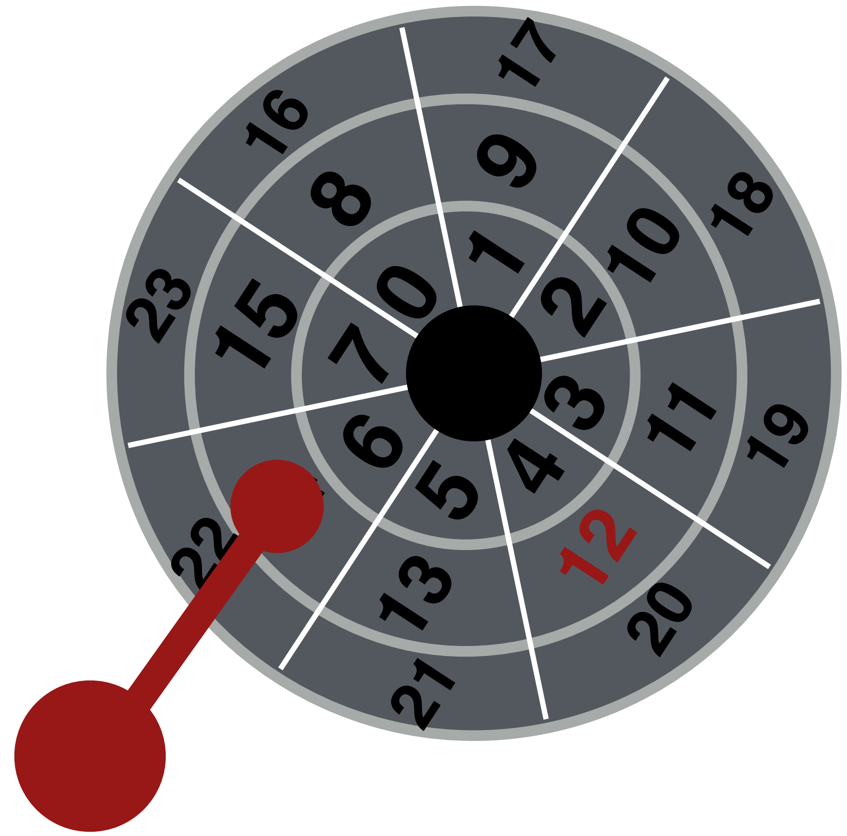
Seek to right track.



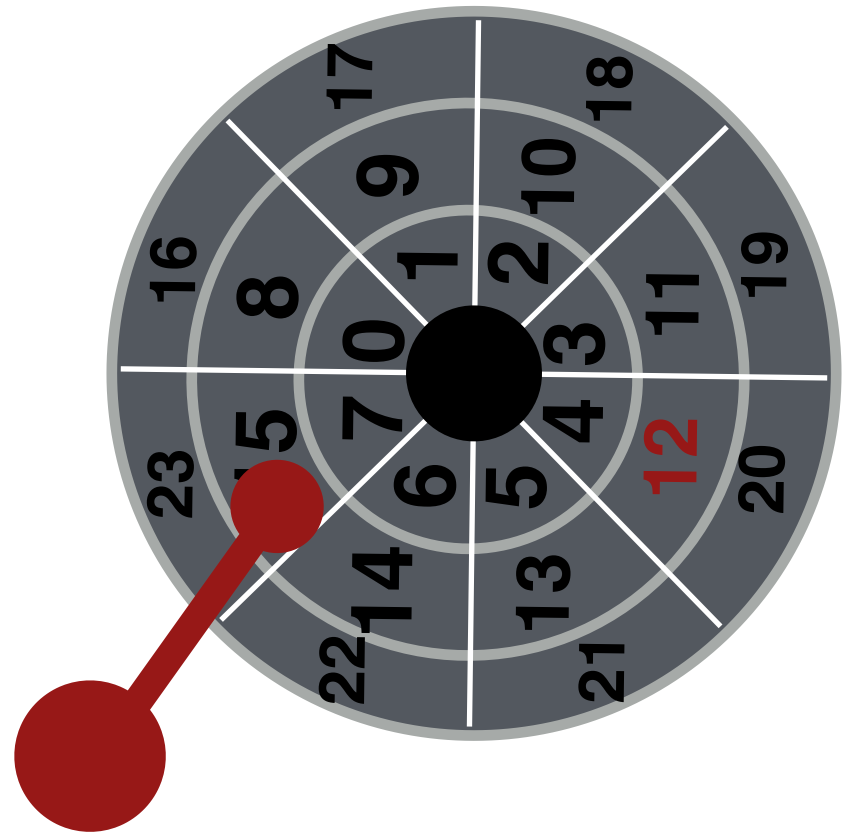
Seek to right track.



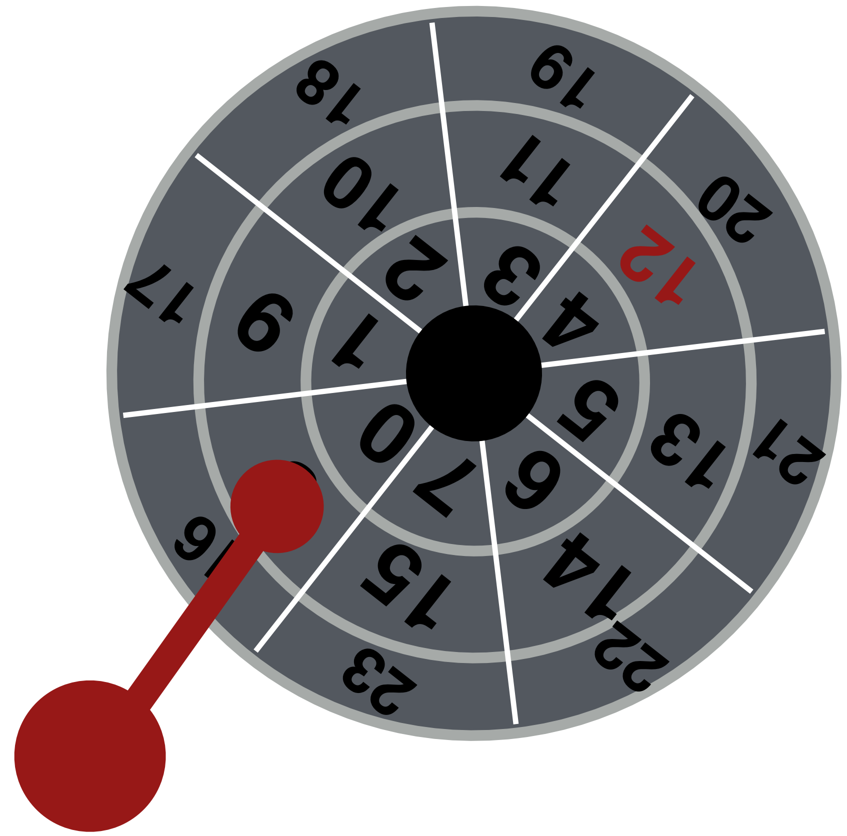
Wait for rotation.



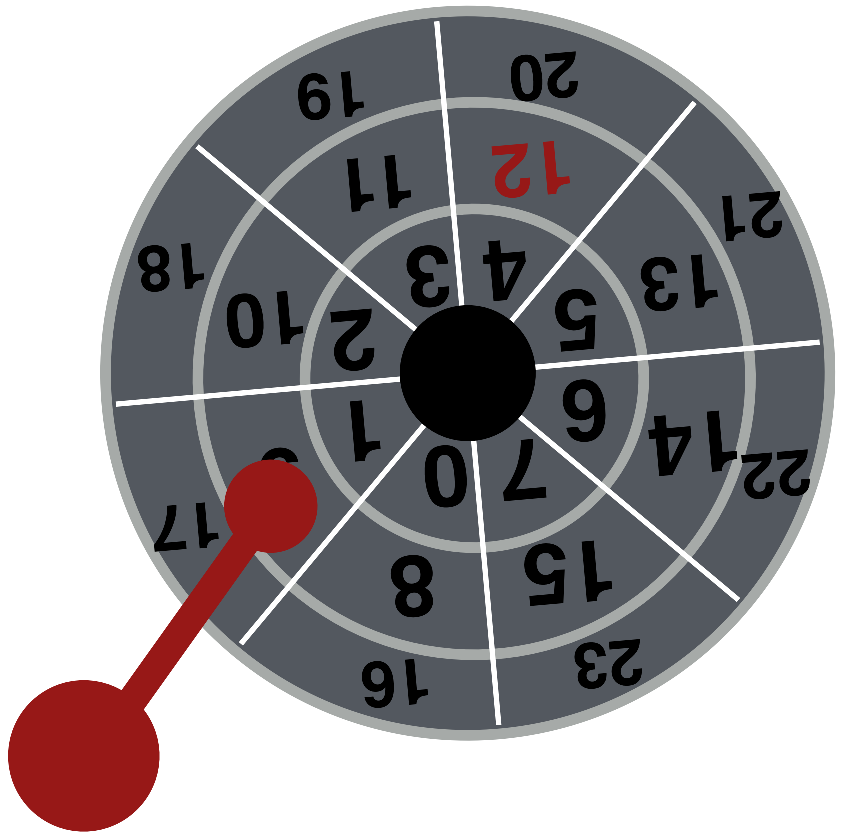
Wait for rotation.



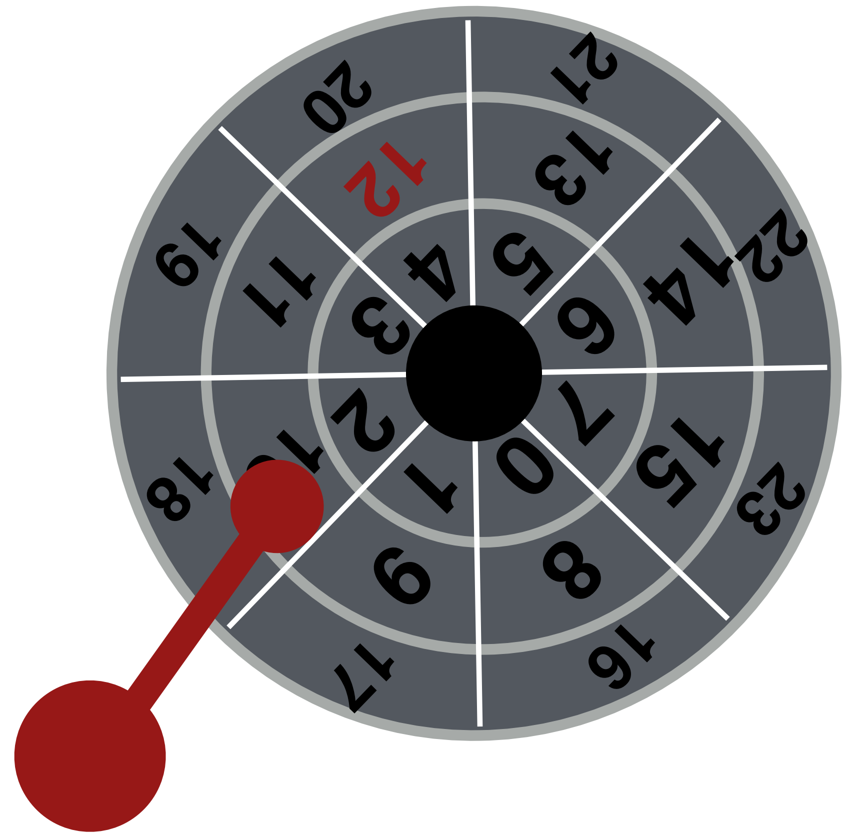
Wait for rotation.



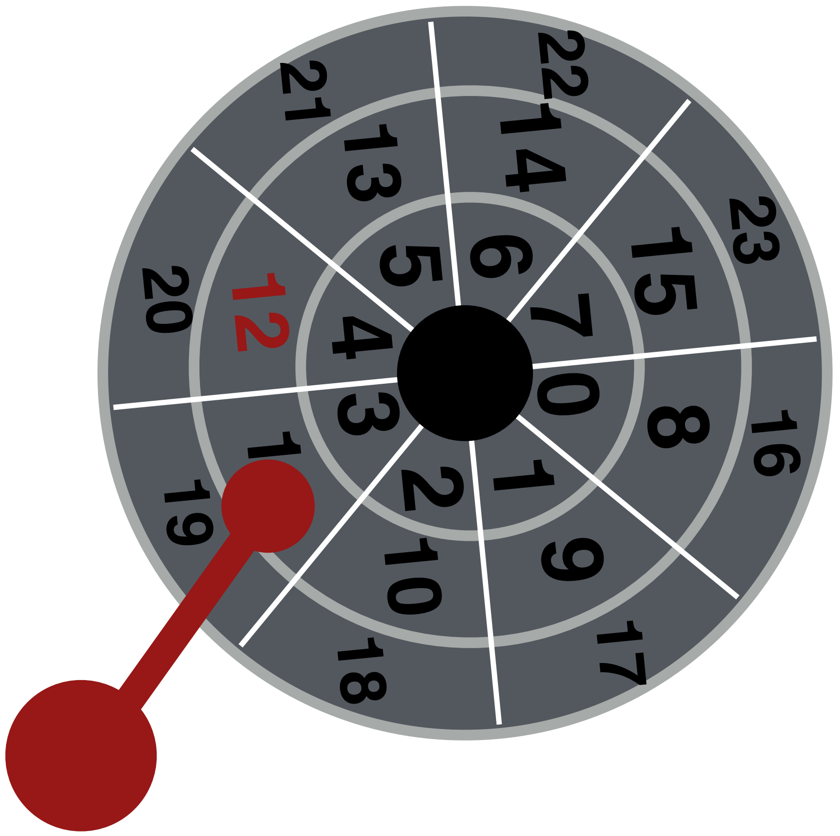
Wait for rotation.



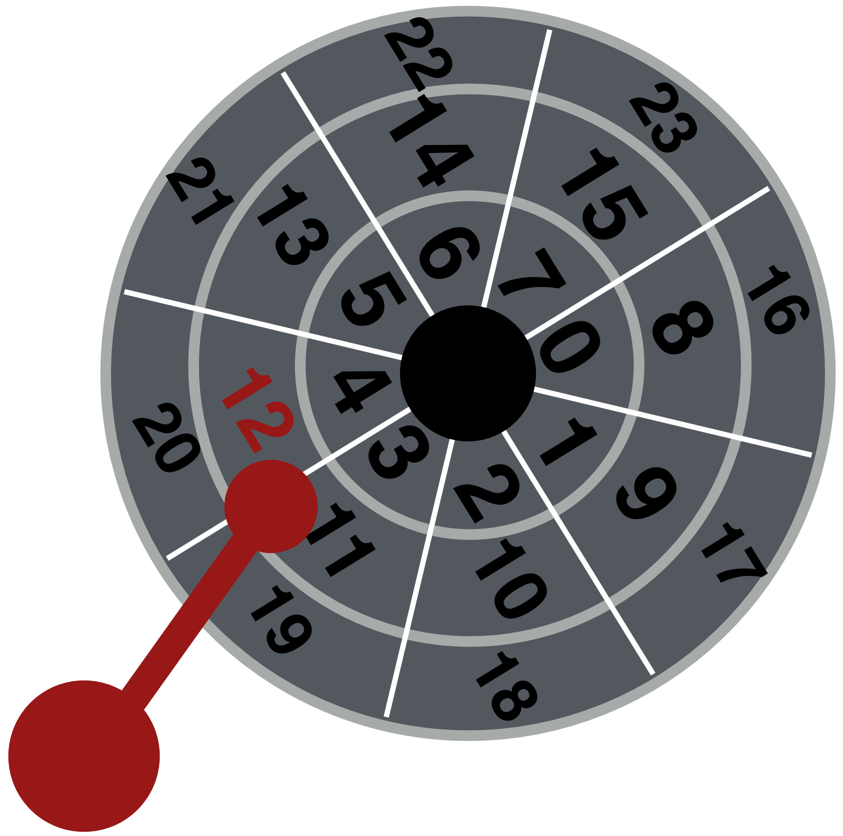
Wait for rotation.



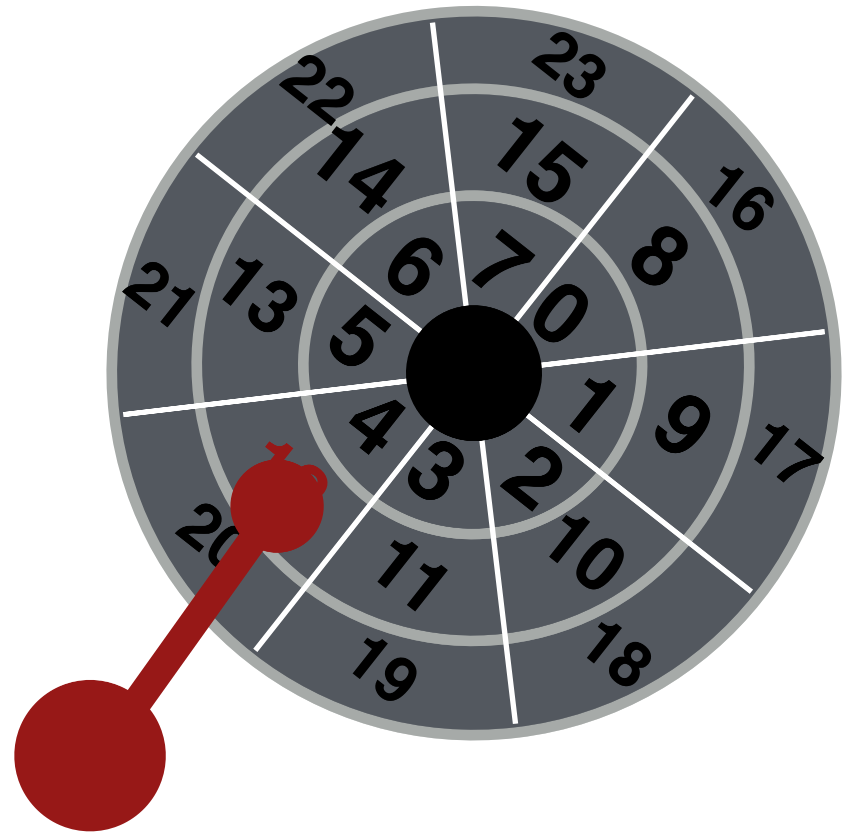
Wait for rotation.



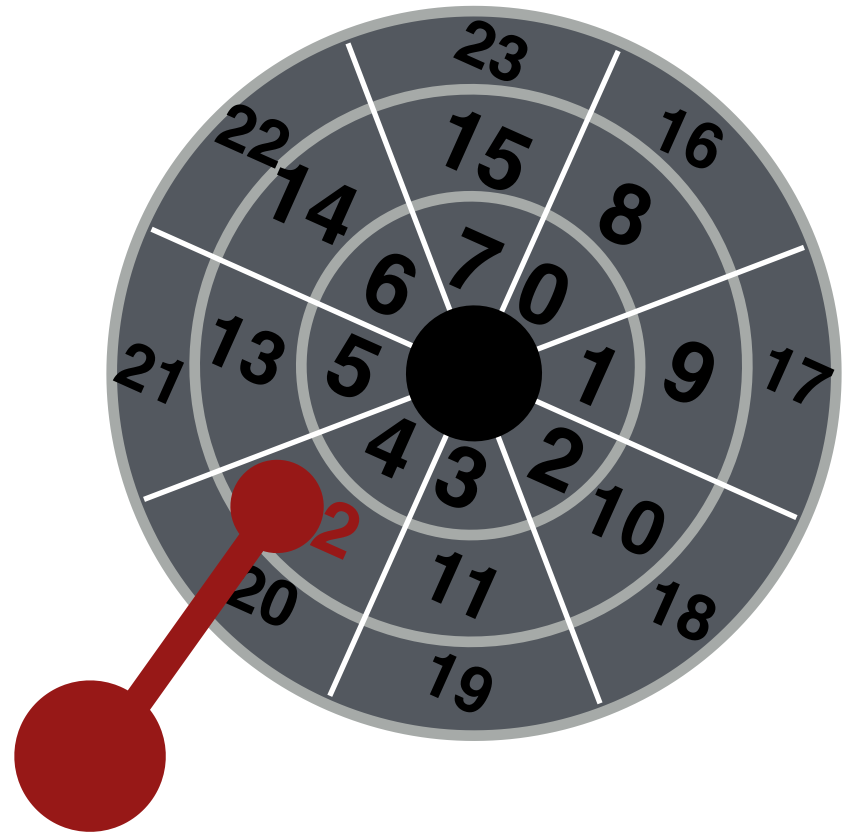
Transfer data.



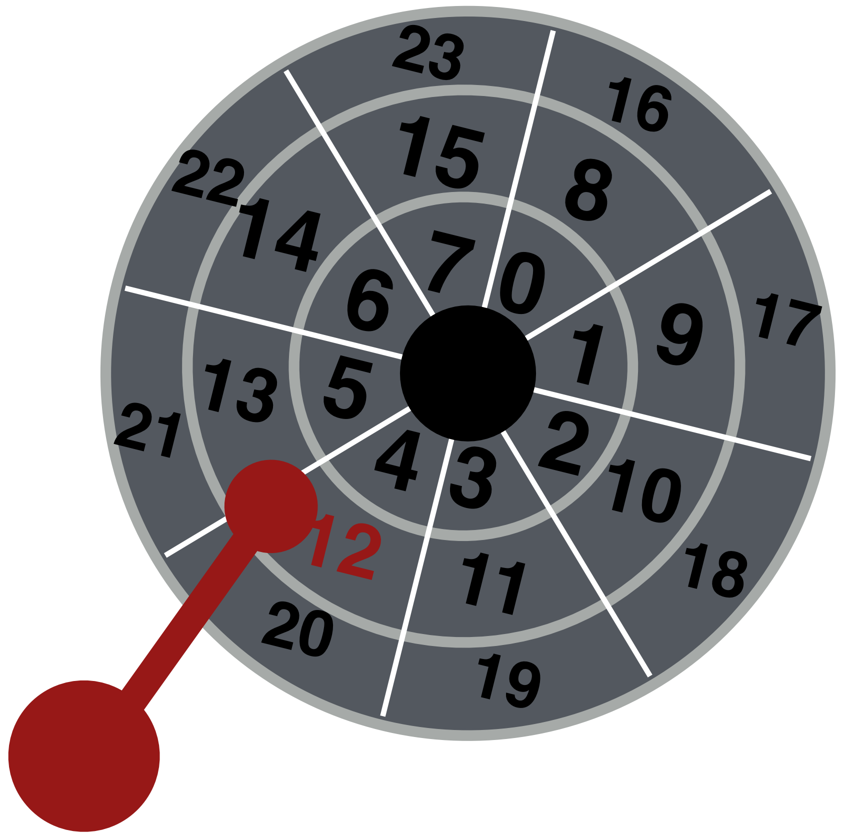
Transfer data.



Transfer data.



Yay!



Seek, Rotate, Transfer

Seek, Rotate, Transfer

Must accelerate, coast, decelerate, settle

Seek, Rotate, Transfer

Must accelerate, coast, decelerate, settle

Seek, Rotate, Transfer

Must accelerate, coast, decelerate, settle

Seeks often take several milliseconds!

Seek, Rotate, Transfer

Must accelerate, coast, decelerate, settle

Seeks often take several milliseconds!

Seek, Rotate, Transfer

Must accelerate, coast, decelerate, settle

Seeks often take several milliseconds!

Settling alone can take 0.5 - 2 ms.

Seek, Rotate, Transfer

Must accelerate, coast, decelerate, settle

Seeks often take several milliseconds!

Settling alone can take 0.5 - 2 ms.

Seek, Rotate, Transfer

Must accelerate, coast, decelerate, settle

Seeks often take several milliseconds!

Settling alone can take 0.5 - 2 ms.

Entire seek often takes 4 - 10 ms.

Seek, Rotate, Transfer

Seek, Rotate, Transfer

Depends on rotations per minute (RPM).

Seek, Rotate, Transfer

Depends on rotations per minute (RPM).

- 7200 RPM is common, 15000 RPM is high end.

Seek, Rotate, Transfer

Depends on rotations per minute (RPM).

- 7200 RPM is common, 15000 RPM is high end.

Seek, Rotate, Transfer

Depends on rotations per minute (RPM).
- 7200 RPM is common, 15000 RPM is high end.

1 / 7200 RPM =

Seek, Rotate, Transfer

Depends on rotations per minute (RPM).
- 7200 RPM is common, 15000 RPM is high end.

1 / 7200 RPM =
1 minute / 7200 rotations =

Seek, Rotate, Transfer

Depends on rotations per minute (RPM).
- 7200 RPM is common, 15000 RPM is high end.

1 / 7200 RPM =
1 minute / 7200 rotations =
1 second / 120 rotations =

Seek, Rotate, Transfer

Depends on rotations per minute (RPM).
- 7200 RPM is common, 15000 RPM is high end.

1 / 7200 RPM =
1 minute / 7200 rotations =
1 second / 120 rotations =
12 ms / rotation

Seek, Rotate, Transfer

Depends on rotations per minute (RPM).
- 7200 RPM is common, 15000 RPM is high end.

1 / 7200 RPM =
1 minute / 7200 rotations =
1 second / 120 rotations =
12 ms / rotation

so it may take **6 ms** on avg to rotate to target (0.5 * 12 ms)

Seek, Rotate, Transfer

Seek, Rotate, Transfer

Pretty fast — depends on RPM and sector density.

Seek, Rotate, Transfer

Pretty fast — depends on RPM and sector density.

Seek, Rotate, Transfer

Pretty fast — depends on RPM and sector density.

100+ MB/s is typical.

Seek, Rotate, Transfer

Pretty fast — depends on RPM and sector density.

100+ MB/s is typical.

Seek, Rotate, Transfer

Pretty fast — depends on RPM and sector density.

100+ MB/s is typical.

1s / 100 MB = 10 ms / MB = 4.9 us / sector

Seek, Rotate, Transfer

Pretty fast — depends on RPM and sector density.

100+ MB/s is typical.

1s / 100 MB = 10 ms / MB = 4.9 us / sector
(assuming 512-byte sector)

Workload

Workload

So…

Workload

So…

- seeks are slow

Workload

So…

- seeks are slow
- rotations are slow

Workload

So…

- seeks are slow
- rotations are slow
- transfers are fast

Workload

So…

- seeks are slow
- rotations are slow
- transfers are fast

Workload

So…
- seeks are slow
- rotations are slow
- transfers are fast

What kind of workload is fastest for disks?

Workload

So…
- seeks are slow
- rotations are slow
- transfers are fast

What kind of workload is fastest for disks?

Workload

So…
- seeks are slow
- rotations are slow
- transfers are fast

What kind of workload is fastest for disks?

Sequential: access sectors in order (transfer dominated)

Random: access sectors arbitrarily (seek+rotation dominated)

Demos: example-rand.csh and example-seq.csh

Disk Spec

	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	32 MB

Disk Spec

	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	32 MB

Sequential workload: what is throughput for each?

Disk Spec

	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	**125 MB/s**	**105 MB/s**
Platters	4	4
Cache	16 MB	32 MB

Cheeta: 125 MB/s.
Barracuda: 105 MB/s.

Disk Spec

	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	32 MB

Random workload: what is throughput for each?
(what else do you need to know?)

Disk Spec

	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	32 MB

Random workload: what is throughput for each?
Assume 16-KB reads.

Disk Spec

	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	32 MB

Random workload: what is throughput for each?
Assume 16-KB reads.

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Cheetah?

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Cheetah?

$$\text{avg rotation} = \frac{1}{2} \times \frac{1 \text{ min}}{15000}$$

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Cheetah?

$$\text{avg rotation} = \frac{1}{2} \times \frac{1 \text{ min}}{15000} \times \frac{60 \text{ sec}}{1 \text{ min}} \times \frac{1000 \text{ ms}}{1 \text{ sec}} = 2 \text{ ms}$$

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

$$\text{transfer} = \frac{1 \text{ sec}}{125 \text{ MB}} \times 16 \text{ KB} \times \frac{1{,}000{,}000 \text{ us}}{1 \text{ sec}} = 125 \text{ us}$$

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Cheetah?

Cheetah time = 4ms + 2ms + 125us = 6.1ms

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Cheetah?

Cheetah time = 4ms + 2ms + 125us = 6.1ms

$$\text{throughput} = \frac{16 \text{ KB}}{6.1\text{ms}}$$

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Cheetah?

Cheetah time = 4ms + 2ms + 125us = 6.1ms

$$\text{throughput} = \frac{16\text{ KB}}{6.1\text{ms}} \times \frac{1\text{ MB}}{1024\text{ KB}} \times \frac{100\text{ ms}}{1\text{ sec}} = 2.5\text{ MB/s}$$

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Barracuda?

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Barracuda?

$$\text{avg rotation} = \frac{1}{2} \times \frac{1\text{ min}}{7200} \times \frac{60\text{ sec}}{1\text{ min}} \times \frac{1000\text{ ms}}{1\text{ sec}} = 4.1\text{ ms}$$

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Barracuda?

$$\text{transfer} = \frac{1 \text{ sec}}{105 \text{ MB}} \times 16 \text{ KB} \times \frac{1{,}000{,}000 \text{ us}}{1 \text{ sec}} = 149 \text{ us}$$

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Barracuda?

Barracuda time = 9ms + 4.1ms + 149us = 13.2ms

$$\text{throughput} = \frac{16\ \text{KB}}{13.2\text{ms}} \times \frac{1\ \text{MB}}{1024\ \text{KB}} \times \frac{1000\ \text{ms}}{1\ \text{sec}}$$

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

How long does an average 16-KB read take w/ Barracuda?

Barracuda time = 9ms + 4.1ms + 149us = 13.2ms

$$\text{throughput} = \frac{16\text{ KB}}{13.2\text{ms}} \times \frac{1\text{ MB}}{1024\text{ KB}} \times \frac{1000\text{ ms}}{1\text{ sec}} = 1.2\text{ MB/s}$$