

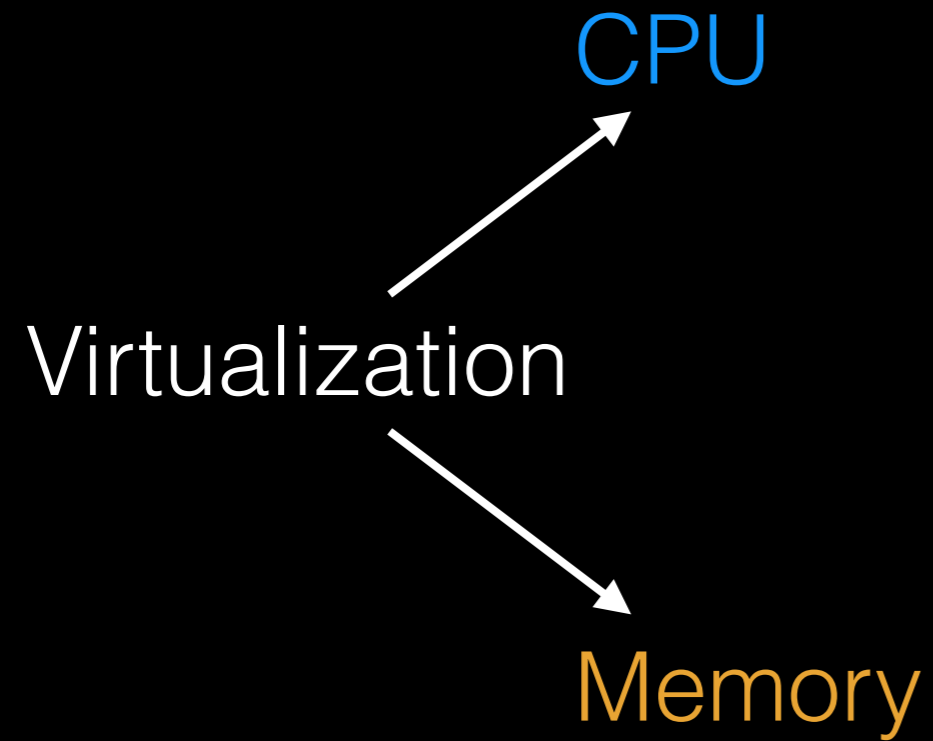
# [537] Threads

Chapters 26

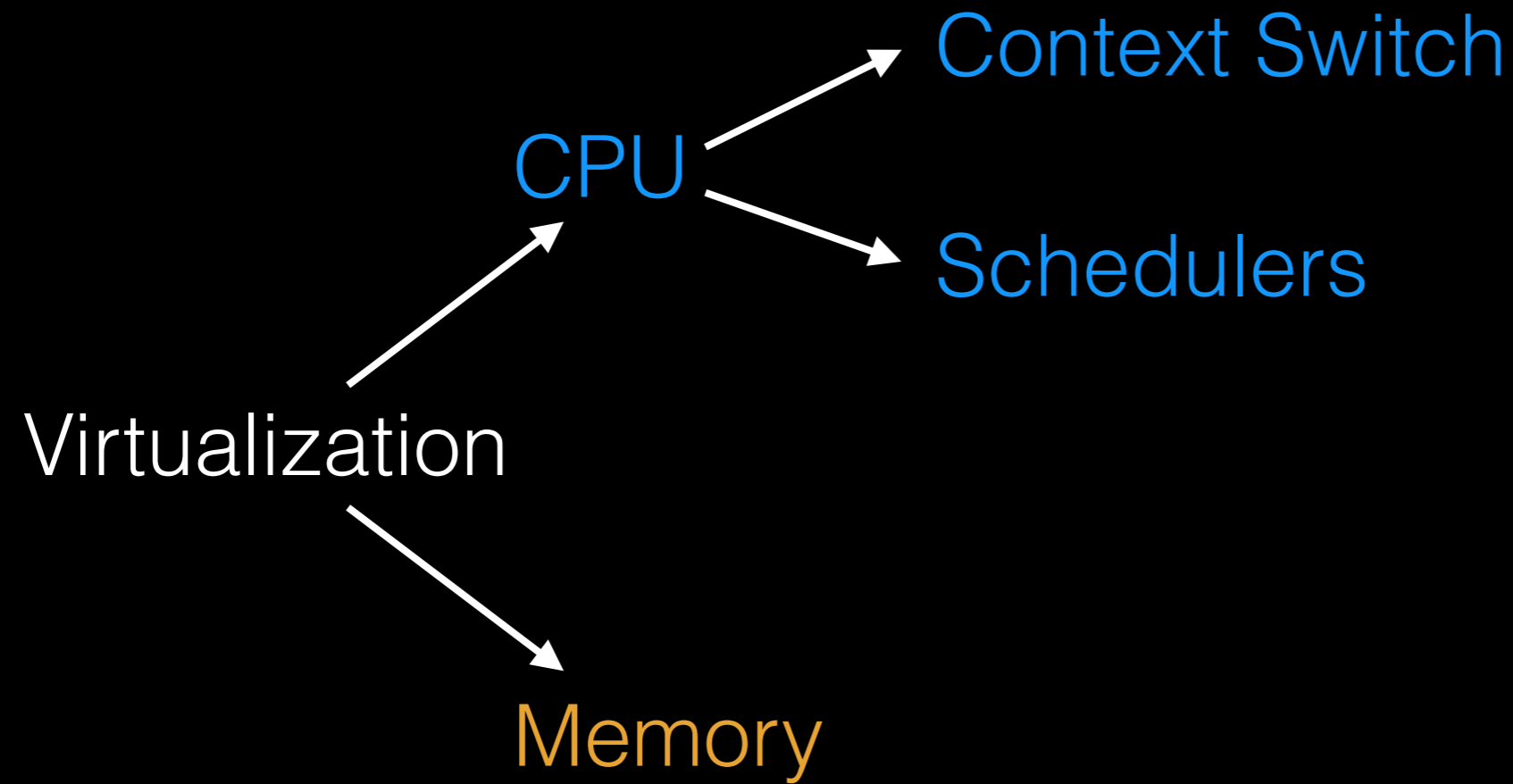
Tyler Harter

10/01/14

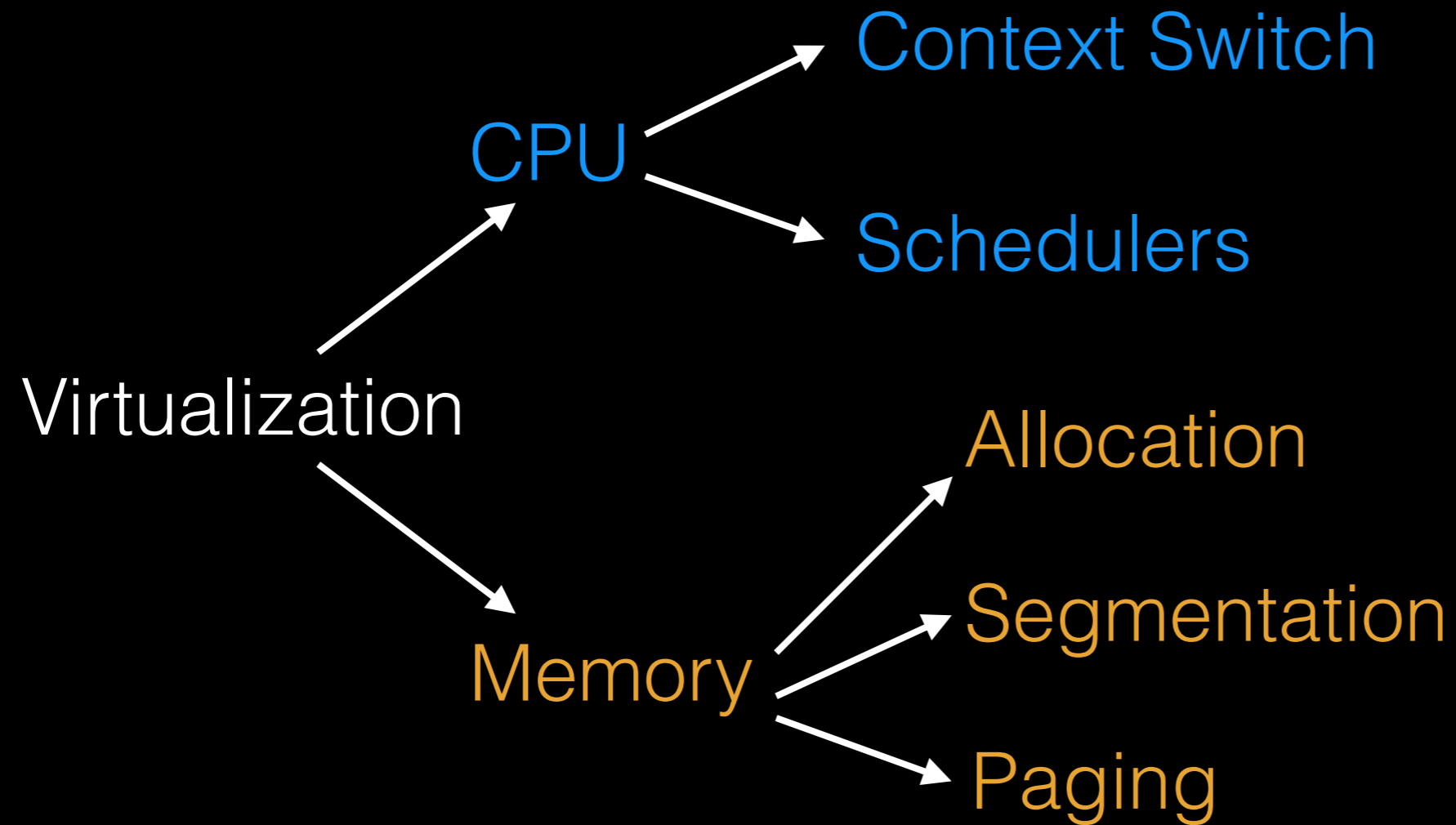
# Review: Easy Piece 1



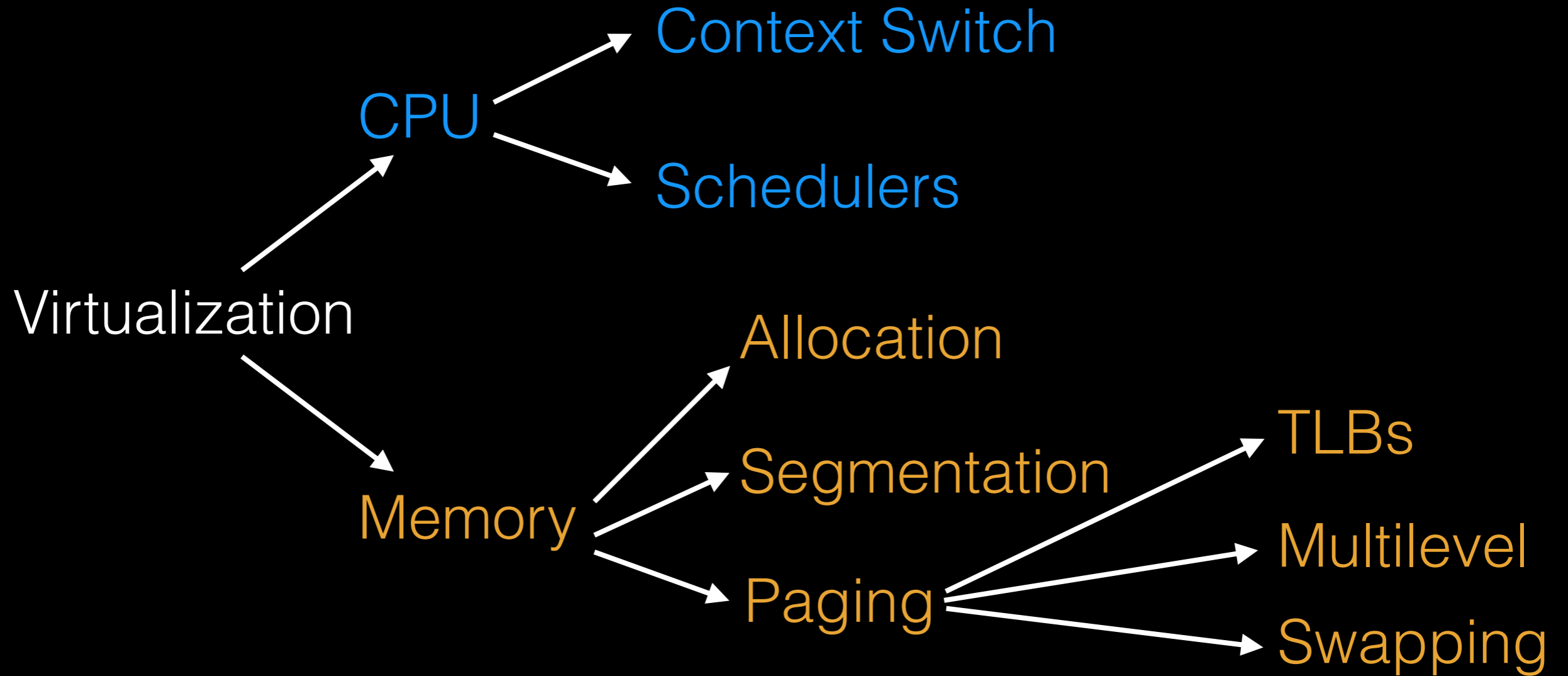
# Review: Easy Piece 1



# Review: Easy Piece 1



# Review: Easy Piece 1

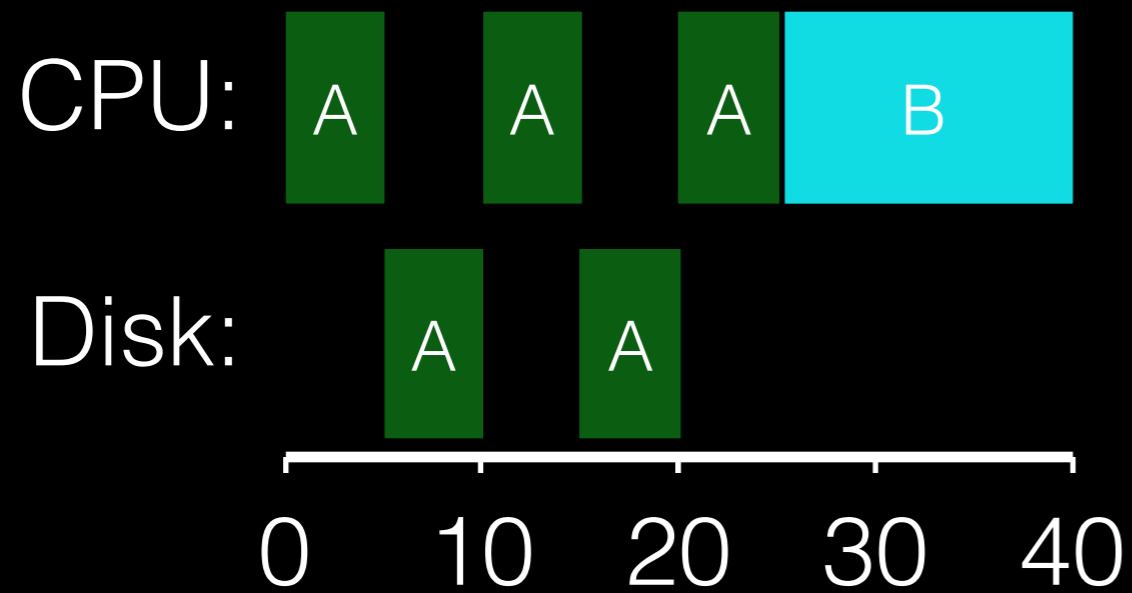


# Review

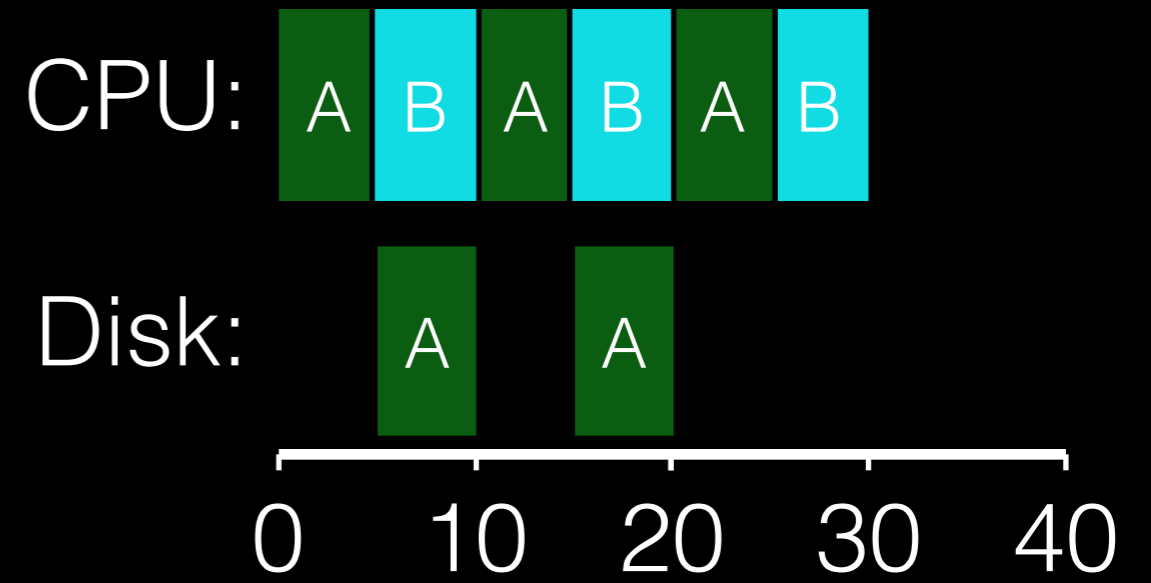
- (1) name 3 places xv6 **stores registers** for context switches?
  - (2) how can you **game** the multi-level feedback queue?
  - (3) what **field** does malloc need for a free node that a kernel doesn't need to track a free page?
  - (4) which segment needs different **translation rules**? Why?
  - (5) how can a TLB **avoid flushing** for context switches
  - (6) what's the advantage of **multi-level PT** over segmented PTs?
  - (7) what **H/W support** is needed to do LRU swapping?
-

Threads

**(a) not interleaved**

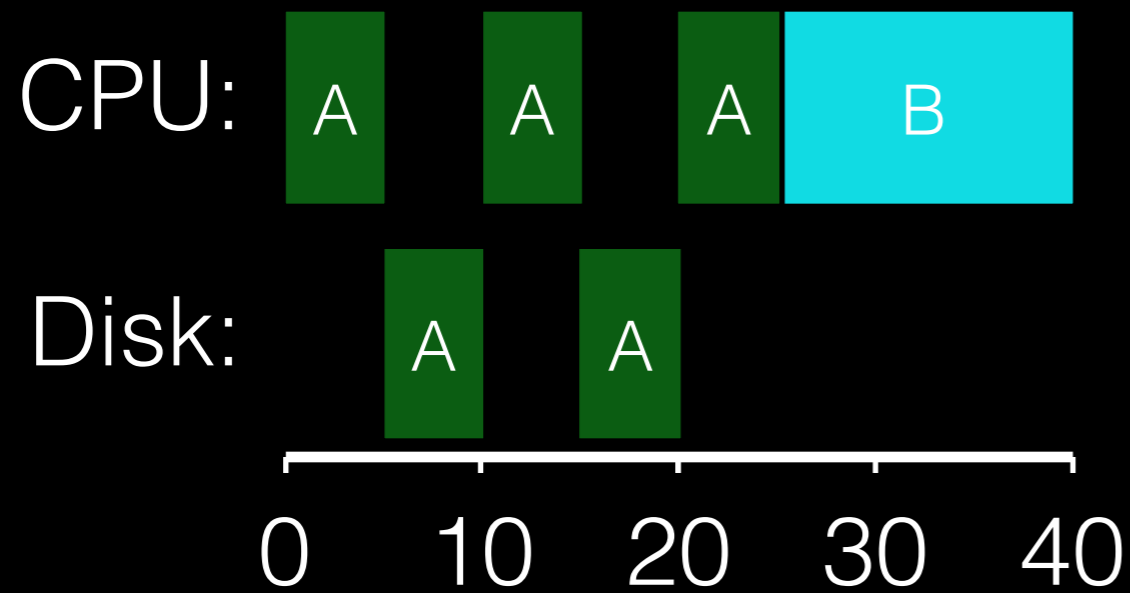


**(b) interleaved**

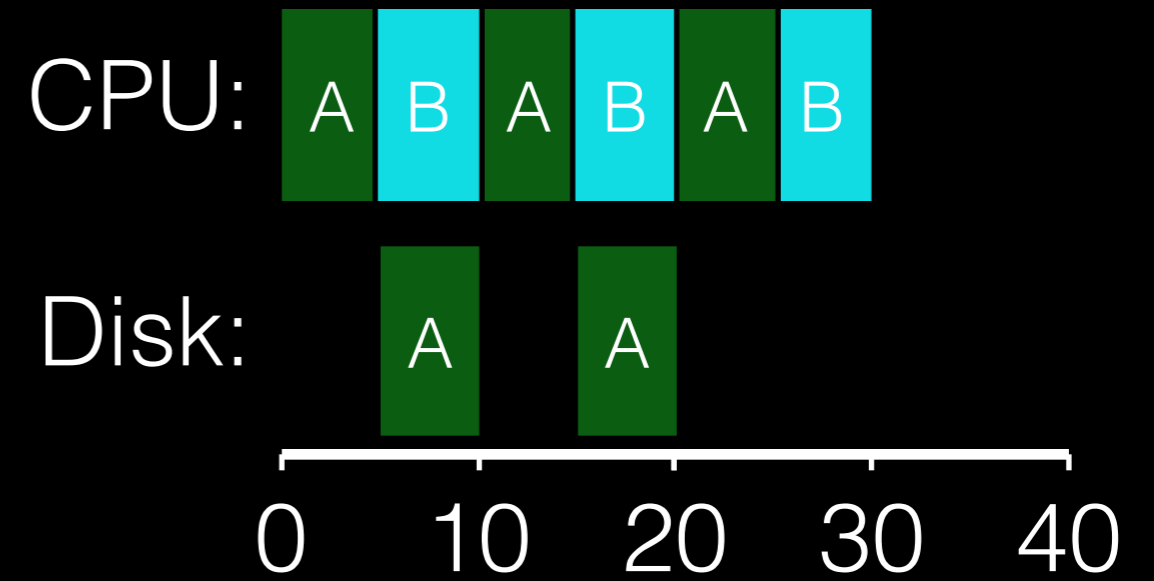




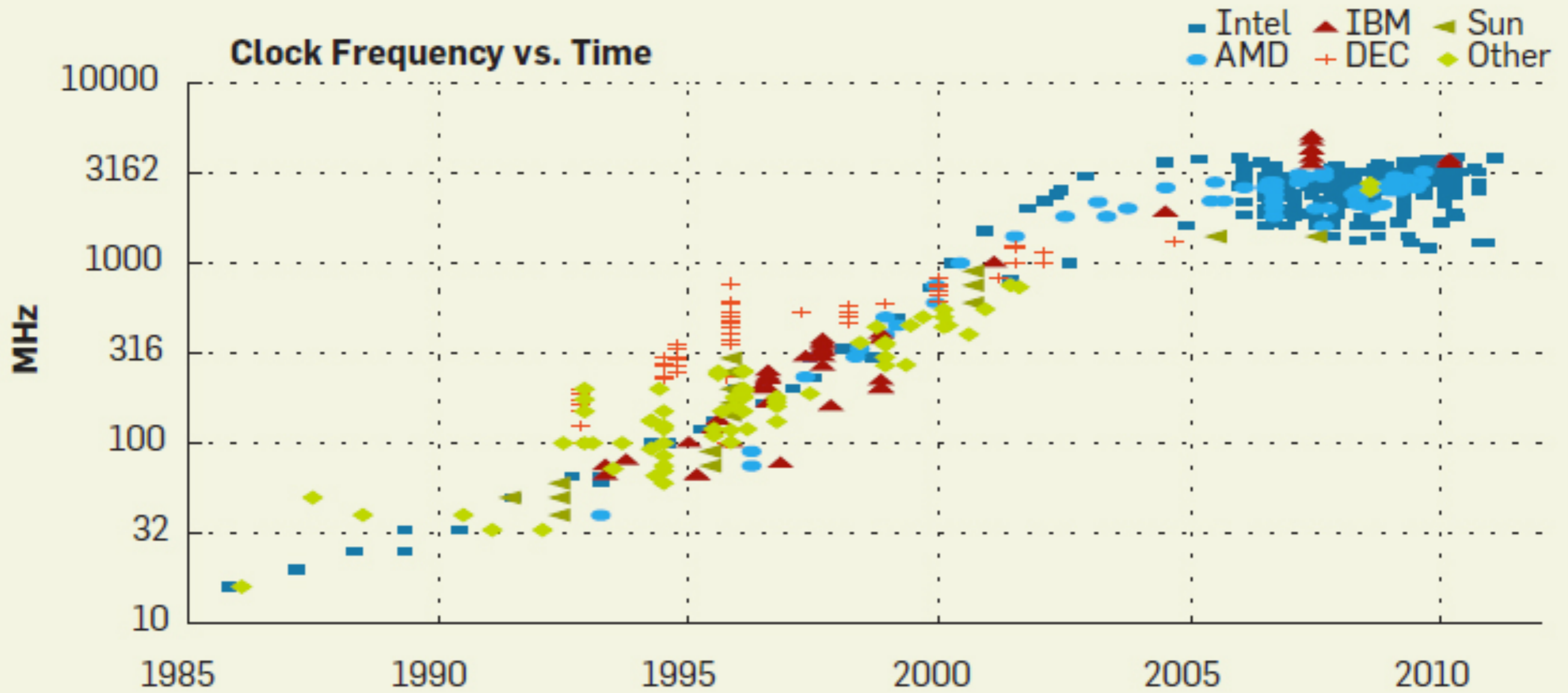
**(a) not interleaved**



**(b) interleaved**



What if there is only one process?



<http://cacm.acm.org/magazines/2012/4/147359-cpu-db-recording-microprocessor-history/fulltext>

# CPU Trends

The future:

- same speed
- more cores

Faster programs => concurrent execution

# Goal

Write applications that fully utilize many CPUs...

# Strategy 1

Build applications from many communicating processes

- like Chrome (process per tab)
- communicate via `pipe()` or similar

Pros/cons?

# Strategy 1

Build applications from many communicating processes

- like Chrome (process per tab)
- communicate via `pipe()` or similar

Pros/cons?

- don't need new abstractions
  - cumbersome programming
  - copying overheads
  - expensive context switching (why expensive?)
-

# Strategy 2

New abstraction: the **thread**.

Threads are just **like processes**, but they **share the address space** (e.g., using same PT).

CPU 1

running  
thread 1

CPU 2

running  
thread 2

RAM





CPU 1

running  
thread 1

CPU 2

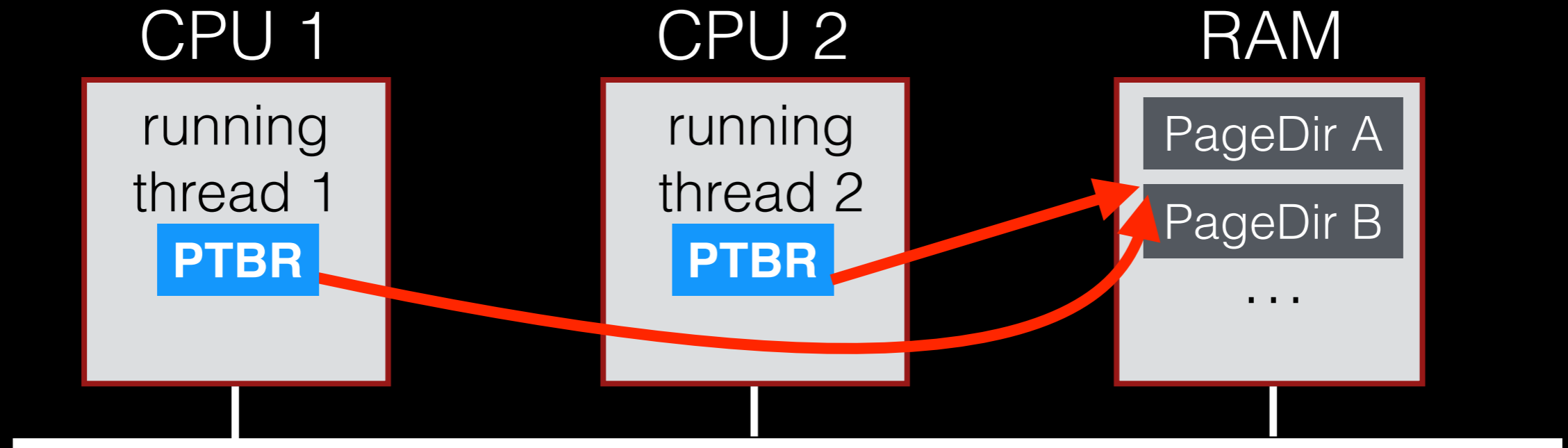
running  
thread 2

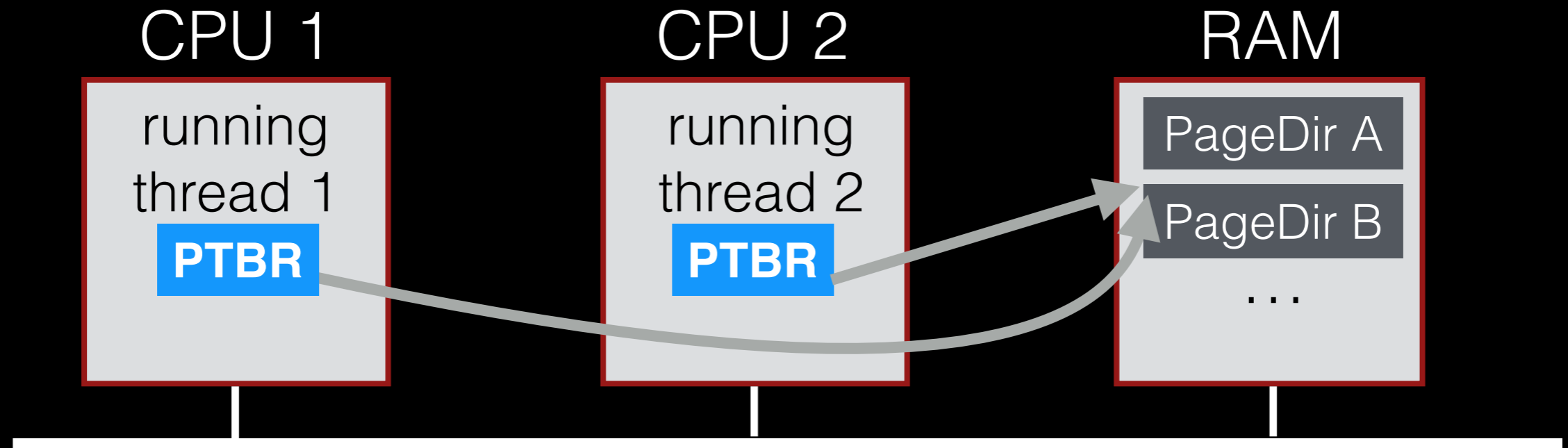
RAM

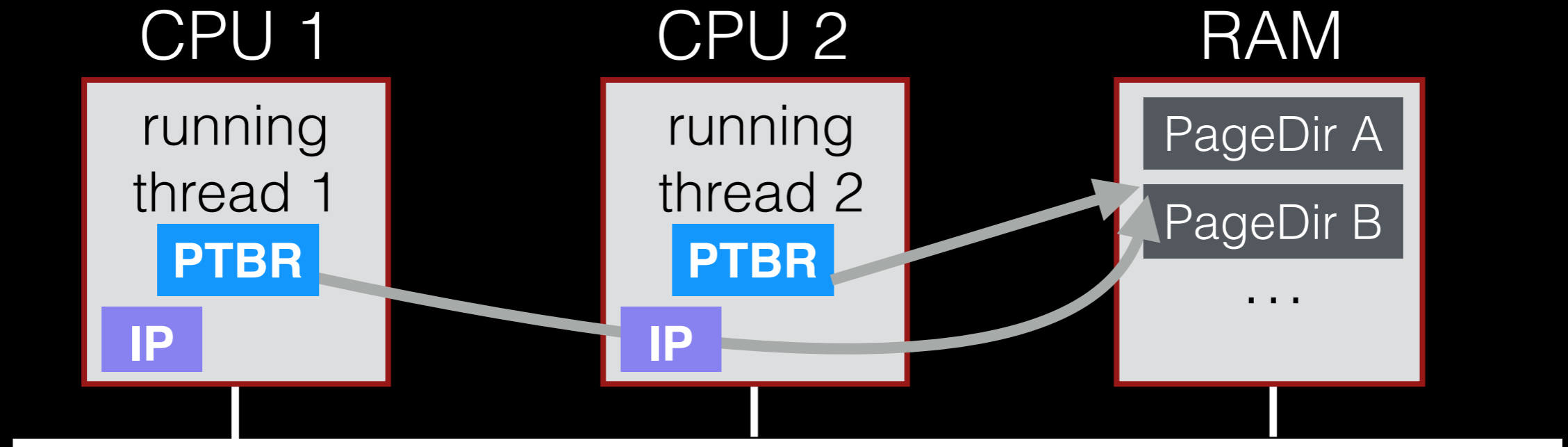
PageDir A

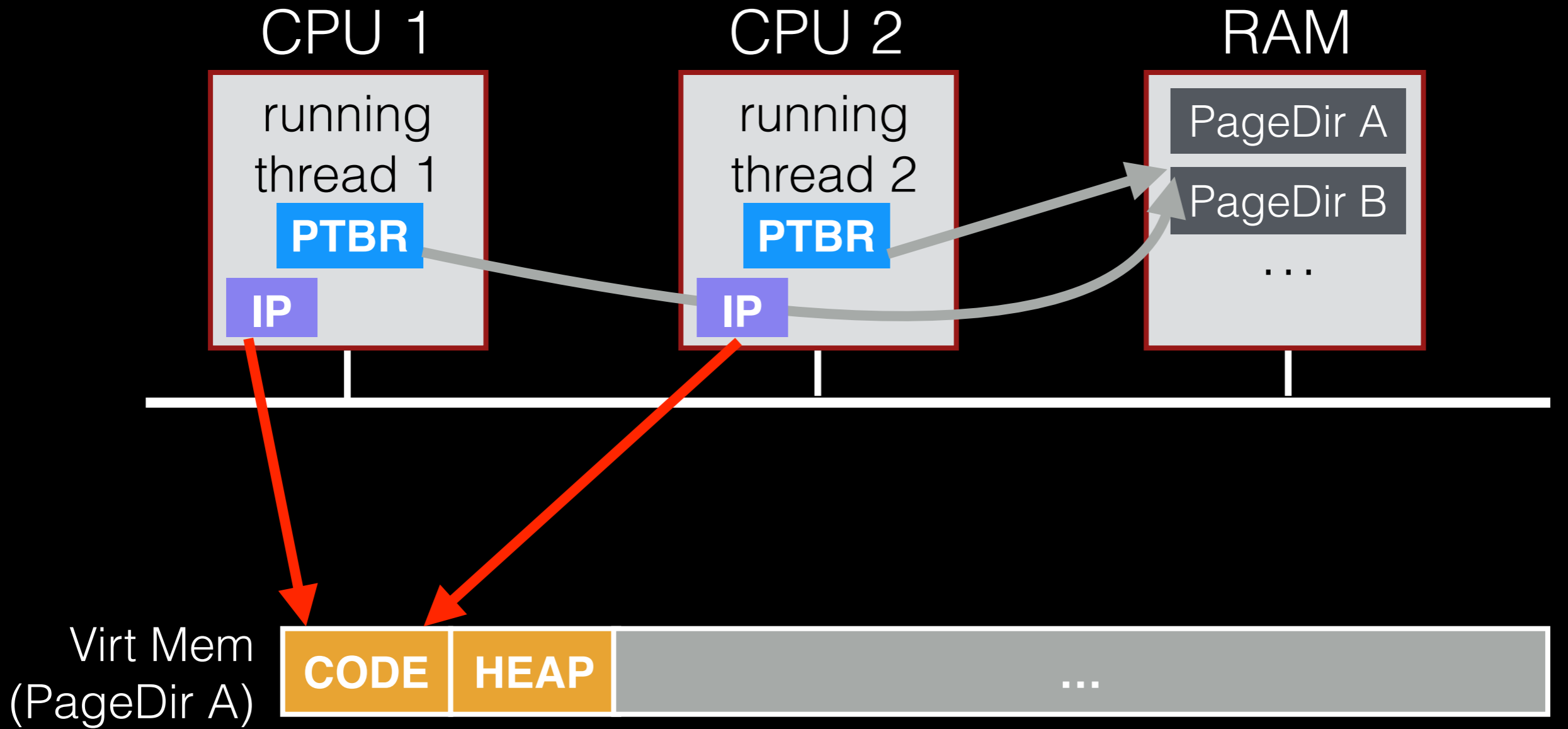
PageDir B

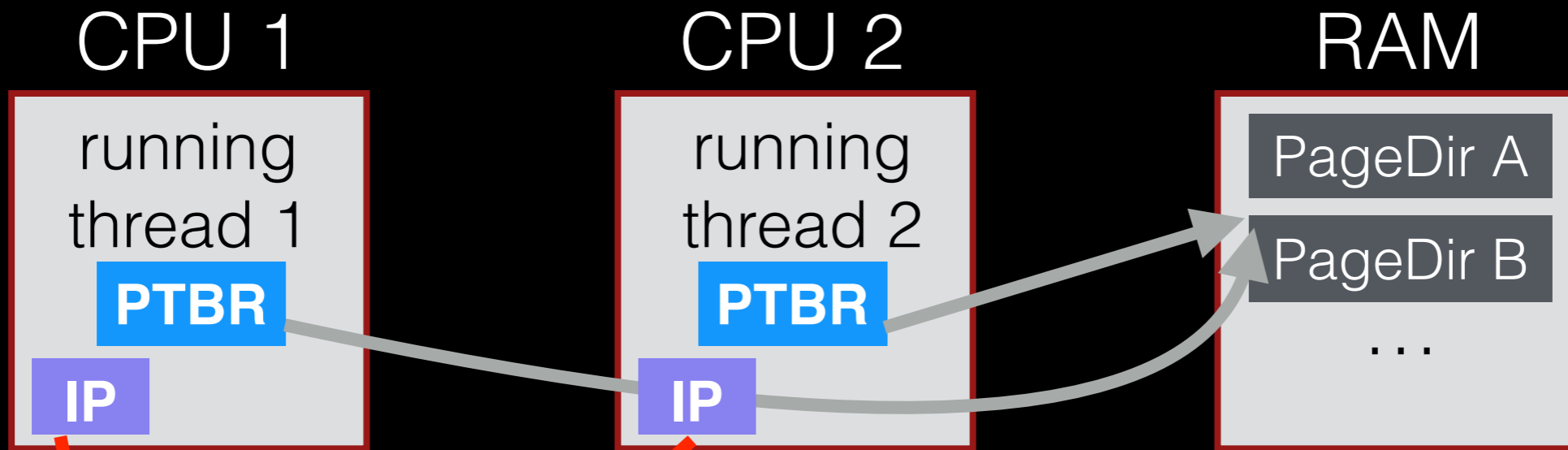
...





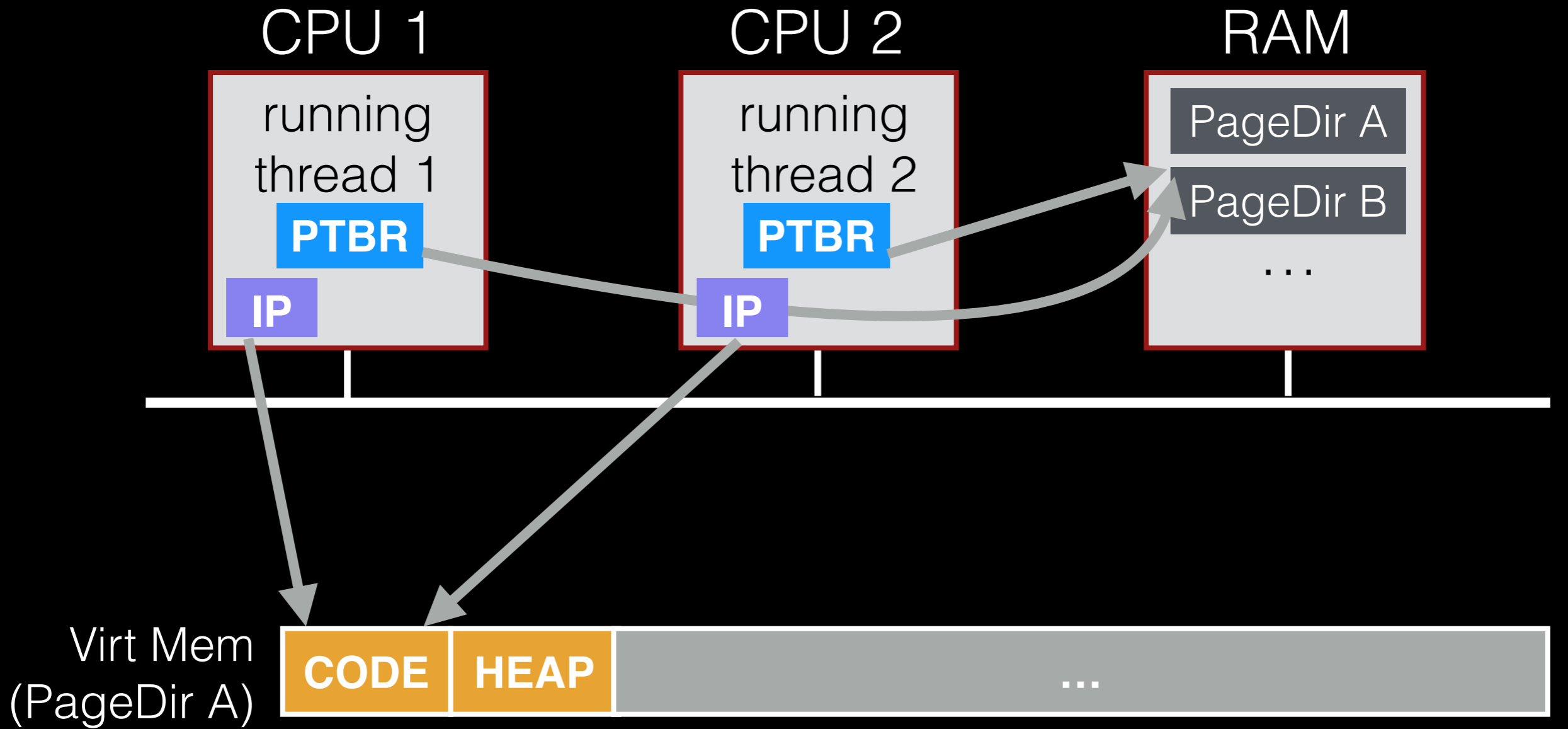


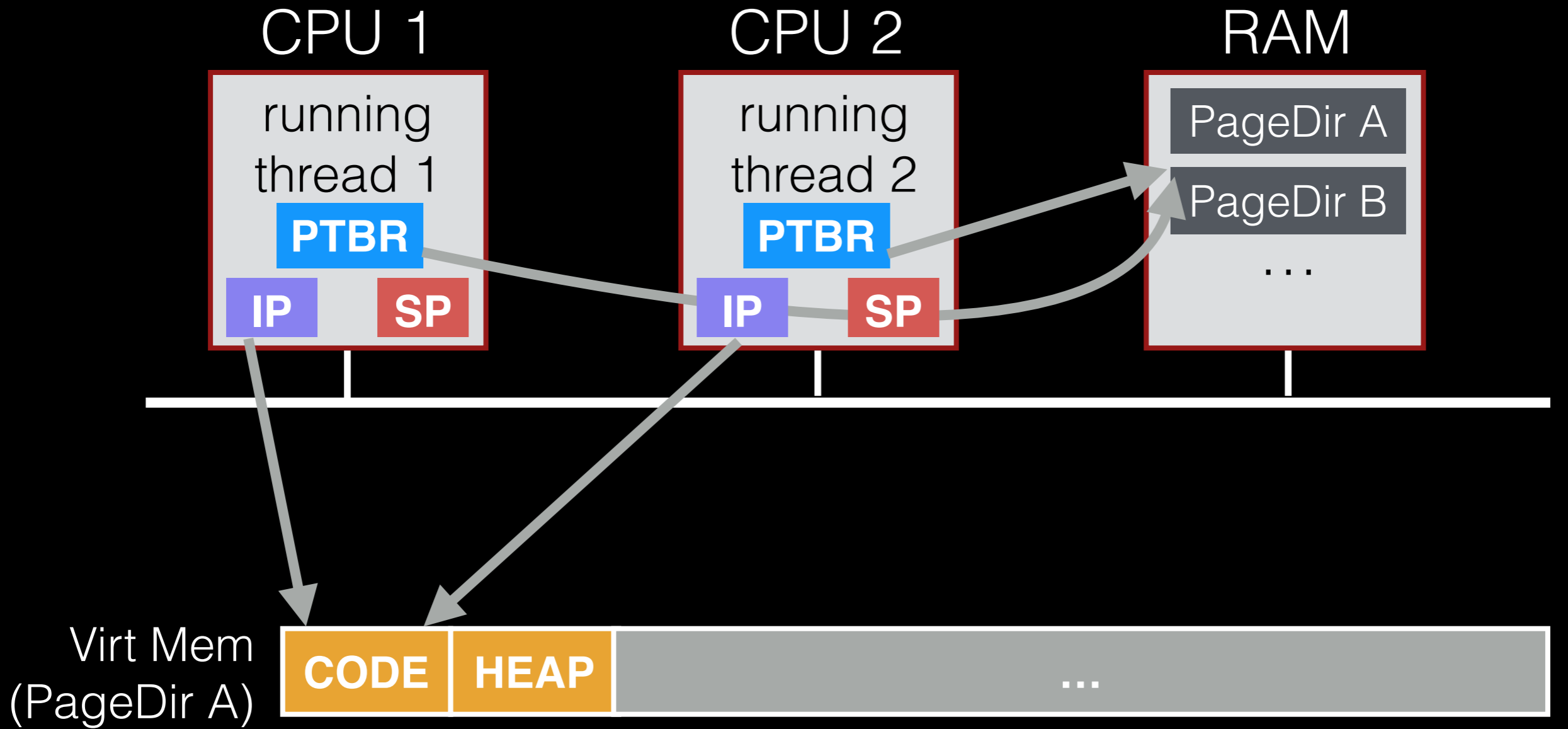




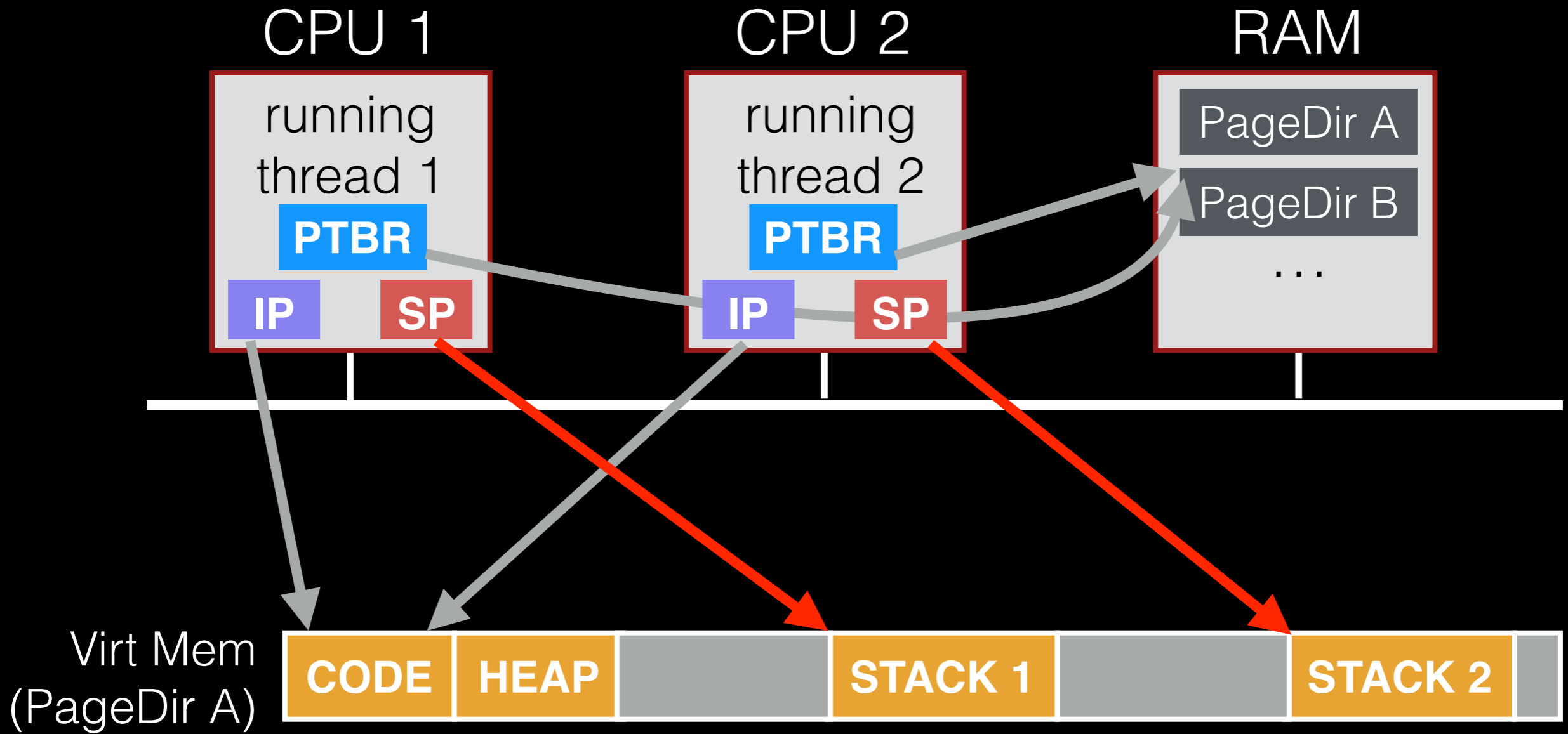
Each thread may be executing different code at the same time

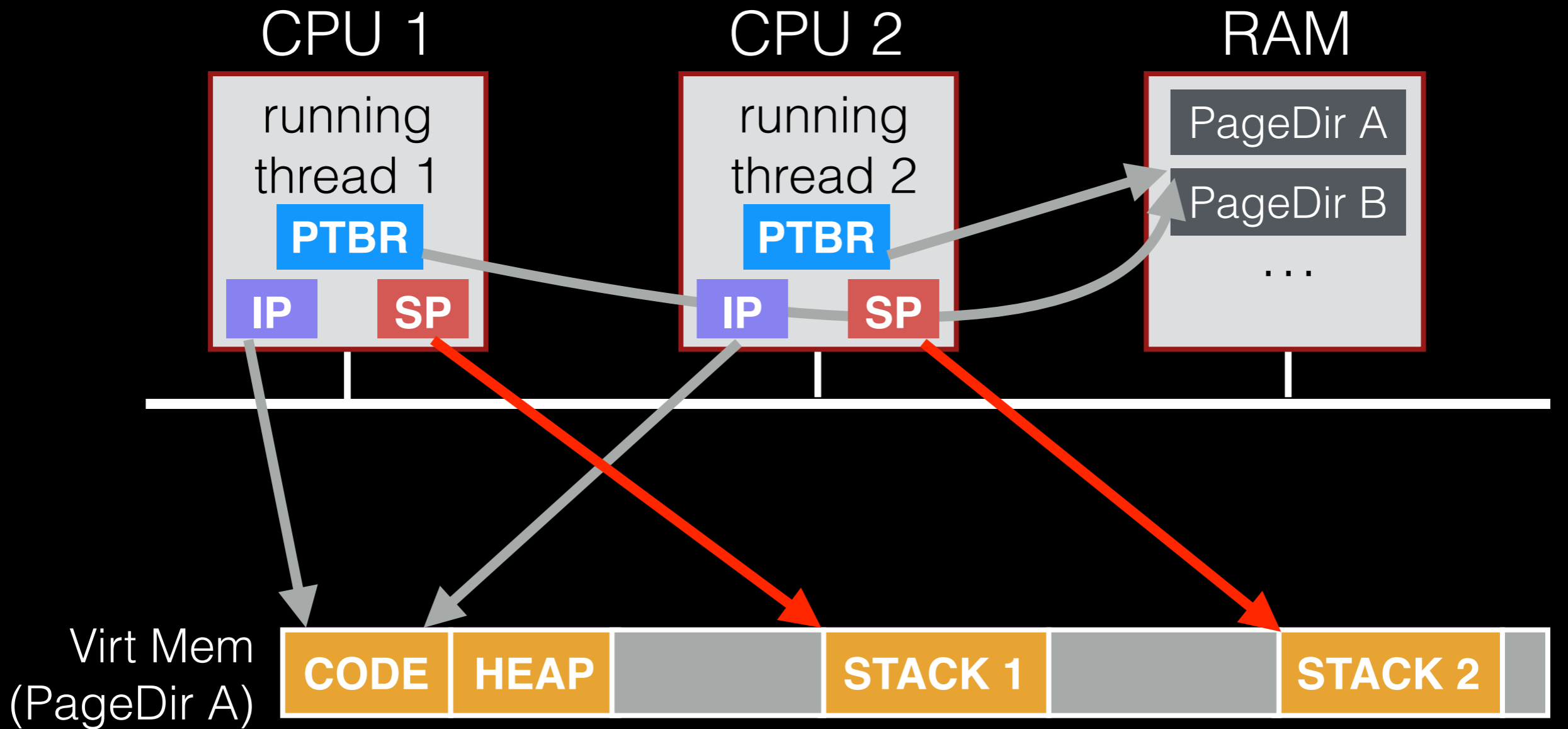












threads executing different functions need different stacks

Demo: basic threads

# Scheduling Problems (One CPU)

**State:**

0x9cd4: 100

%eax: ?

%rip = 0x195

process  
control  
blocks:

Thread 1

%eax: ?  
%rip: 0x195

Thread 2

%eax: ?  
%rip: 0x195

T1 → 0x195    mov 0x9cd4, %eax  
          0x19a    add \$0x1, %eax  
          0x19d    mov %eax, 0x9cd4

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: 100`

`%rip = 0x19a`

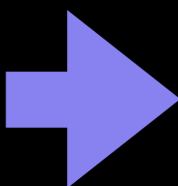
process  
control  
blocks:

Thread 1

`%eax: ?`  
`%rip: 0x195`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T1  `0x195 mov 0x9cd4, %eax`  
`0x19a add $0x1, %eax`  
`0x19d mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: 101`

`%rip = 0x19d`

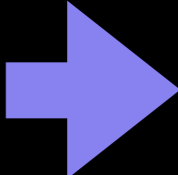
process  
control  
blocks:

Thread 1

`%eax: ?`  
`%rip: 0x195`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T1  `0x195` `mov 0x9cd4, %eax`  
`0x19a` `add $0x1, %eax`  
`0x19d` `mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 101`

`%rip = 0x1a2`

process  
control  
blocks:

Thread 1

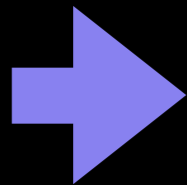
`%eax: ?`  
`%rip: 0x195`

Thread 2

`%eax: ?`  
`%rip: 0x195`

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```

T1



# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 101`

`%rip = 0x1a2`

process

control

blocks:

Thread 1

`%eax: ?`

`%rip: 0x195`

Thread 2

`%eax: ?`

`%rip: 0x195`

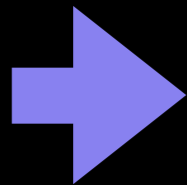
## Context Switch

`0x195 mov 0x9cd4, %eax`

`0x19a add $0x1, %eax`

`0x19d mov %eax, 0x9cd4`

T1





# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: ?`

`%rip = 0x195`

process

control

blocks:

Thread 1

`%eax: 101`  
`%rip: 0x1a2`

Thread 2

`%eax: ?`  
`%rip: 0x195`

## Context Switch

T2 →

<code>0x195</code>	<code>mov</code>	<code>0x9cd4</code>	<code>, %eax</code>
<code>0x19a</code>	<code>add</code>	<code>\$0x1</code>	<code>, %eax</code>
<code>0x19d</code>	<code>mov</code>	<code>%eax</code>	<code>, 0x9cd4</code>

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: ?`

`%rip = 0x195`

process  
control  
blocks:

Thread 1

`%eax: 101`  
`%rip: 0x1a2`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T2  `0x195` `mov 0x9cd4, %eax`  
`0x19a` `add $0x1, %eax`  
`0x19d` `mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 101`

`%rip = 0x19a`

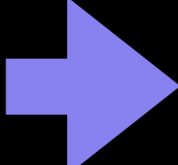
process  
control  
blocks:

Thread 1

`%eax: 101`  
`%rip: 0x1a2`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T2  `0x195 mov 0x9cd4, %eax`  
`0x19a add $0x1, %eax`  
`0x19d mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 102`

`%rip = 0x19d`

process

control

blocks:

Thread 1

`%eax: 101`  
`%rip: 0x1a2`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T2 → `0x195 mov 0x9cd4, %eax`  
`0x19a add $0x1, %eax`  
`0x19d mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 102`

`%eax: 102`

`%rip = 0x1a2`

process  
control  
blocks:

Thread 1

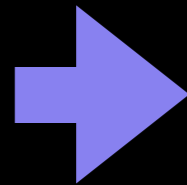
`%eax: 101`  
`%rip: 0x1a2`

Thread 2

`%eax: ?`  
`%rip: 0x195`

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```

T2



# Scheduling Problems (One CPU)

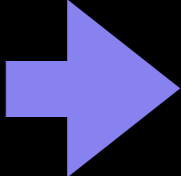
**State:** **GOOD!**  
**0x9cd4:** 102 process  
**%eax:** 102 control  
**%rip = 0x1a2** blocks:

Thread 1

%eax: 101  
%rip: 0x1a2

Thread 2

%eax: ?  
%rip: 0x195

T2 

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```

Another schedule

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: ?`

`%rip = 0x195`

process  
control  
blocks:

Thread 1

`%eax: ?`  
`%rip: 0x195`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T1  `0x195` `mov 0x9cd4, %eax`  
`0x19a` `add $0x1, %eax`  
`0x19d` `mov %eax, 0x9cd4`



# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: 100`

`%rip = 0x19a`

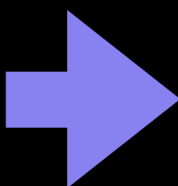
process  
control  
blocks:

Thread 1

`%eax: ?`  
`%rip: 0x195`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T1  `0x195 mov 0x9cd4, %eax`  
`0x19a add $0x1, %eax`  
`0x19d mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: 101`

`%rip = 0x19d`

process  
control  
blocks:

Thread 1

`%eax: ?`  
`%rip: 0x195`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T1  `0x195 mov 0x9cd4, %eax`  
`0x19a add $0x1, %eax`  
`0x19d mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: 101`

`%rip = 0x19d`

process

control

blocks:

Thread 1

`%eax: ?`

`%rip: 0x195`

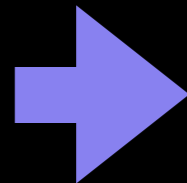
Thread 2

`%eax: ?`

`%rip: 0x195`

## Context Switch

T1



```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: ?`

`%rip = 0x195`

process

control

blocks:

Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: ?`  
`%rip: 0x195`

## Context Switch

T2 →

<code>0x195</code>	<code>mov</code>	<code>0x9cd4</code>	<code>, %eax</code>
<code>0x19a</code>	<code>add</code>	<code>\$0x1</code>	<code>, %eax</code>
<code>0x19d</code>	<code>mov</code>	<code>%eax</code>	<code>, 0x9cd4</code>

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: ?`

`%rip = 0x195`

process  
control  
blocks:

Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T2  `0x195` `mov 0x9cd4, %eax`  
`0x19a` `add $0x1, %eax`  
`0x19d` `mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: 100`

`%rip = 0x19a`

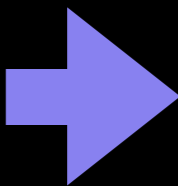
process  
control  
blocks:

Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T2  `0x195` `mov 0x9cd4, %eax`  
`0x19a` `add $0x1, %eax`  
`0x19d` `mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 100`

`%eax: 101`

`%rip = 0x19d`

process  
control  
blocks:

Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T2  `0x195` `mov 0x9cd4, %eax`  
`0x19a` `add $0x1, %eax`  
`0x19d` `mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 101`

`%rip = 0x1a2`

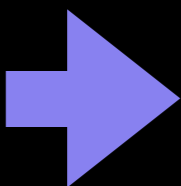
process  
control  
blocks:

Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: ?`  
`%rip: 0x195`

T2 

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```



# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 101`

`%rip = 0x1a2`

process

control

blocks:

Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: ?`  
`%rip: 0x195`

## Context Switch

`0x195 mov 0x9cd4, %eax`

`0x19a add $0x1, %eax`

`0x19d mov %eax, 0x9cd4`

T2 

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 101`

`%rip = 0x19d`

process

control

blocks:

Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: 101`  
`%rip: 0x1a2`

## Context Switch

T1 → `0x195 mov 0x9cd4, %eax`  
`0x19a add $0x1, %eax`  
`0x19d mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 101`

`%rip = 0x19d`

process  
control  
blocks:

Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: 101`  
`%rip: 0x1a2`

T1  `0x195 mov 0x9cd4, %eax`  
`0x19a add $0x1, %eax`  
`0x19d mov %eax, 0x9cd4`

# Scheduling Problems (One CPU)

**State:**

`0x9cd4: 101`

`%eax: 101`

`%rip = 0x1a2`

process  
control  
blocks:

Thread 1

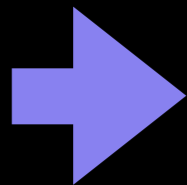
`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: 101`  
`%rip: 0x1a2`

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```

T1



# Scheduling Problems (One CPU)

**State:**  
`0x9cd4: 101`  
`%eax: 101`  
`%rip = 0x1a2`

**process control blocks:**

**WRONG!**

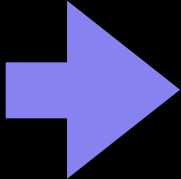
Thread 1

`%eax: 101`  
`%rip: 0x19d`

Thread 2

`%eax: 101`  
`%rip: 0x1a2`

```
0x195  mov 0x9cd4, %eax
0x19a  add $0x1, %eax
0x19d  mov %eax, 0x9cd4
```

T1 

# Timeline View

## Thread 1

```
mov 0x123, %eax  
add %0x1, %eax  
mov %eax, 0x123
```

## Thread 2

```
mov 0x123, %eax  
add %0x1, %eax  
mov %eax, 0x123
```

How much is added?

# Timeline View

## Thread 1

mov 0x123, %eax

add %0x1, %eax

mov %eax, 0x123

## Thread 2

mov 0x123, %eax

add %0x1, %eax

mov %eax, 0x123

How much is added?

# Timeline View

## Thread 1

mov 0x123, %eax

add %0x1, %eax

mov %eax, 0x123

## Thread 2

mov 0x123, %eax

add %0x1, %eax

mov %eax, 0x123

How much is added?



# Timeline View

## Thread 1

```
mov 0x123, %eax  
add %0x1, %eax  
mov %eax, 0x123
```

## Thread 2

```
mov 0x123, %eax  
add %0x1, %eax  
mov %eax, 0x123
```

How much is added?

# Timeline View

## Thread 1

```
mov 0x123, %eax  
add %0x1, %eax  
mov %eax, 0x123
```

## Thread 2

```
mov 0x123, %eax  
add %0x1, %eax
```

```
mov %eax, 0x123
```

How much is added?

# Debugging

Concurrency leads to **non-deterministic** bugs, called **race conditions**.

Whether bug manifests depends on **CPU schedule!**

**Passing tests** means little.

How to program: imagine **scheduler is malicious**.

---

# What do we want?

Want **all or none** of these instructions to execute.  
That is, we want them to be **atomic**.

```
mov 0x123, %eax  
add %0x1, %eax  
mov %eax, 0x123
```

# What do we want?

Want **all or none** of these instructions to execute.  
That is, we want them to be **atomic**.

```
mov 0x123, %eax  
add %0x1, %eax  
mov %eax, 0x123
```

— critical section

# What do we want?

Want **all or none** of these instructions to execute.  
That is, we want them to be **atomic**.

```
mov 0x123, %eax  
add %0x1, %eax  
mov %eax, 0x123
```

— critical section

We want **mutual exclusion** for critical sections.  
That is, if I run, you can't (and vice versa).

More Demos

# Announcements

Be sure you've read up to Chapter 24 in OSTEP!

p2a due this Friday.

Office hours today. In office. 1pm.