OS and Virtual Memory

CS110 Discussion 13

Suting Chen

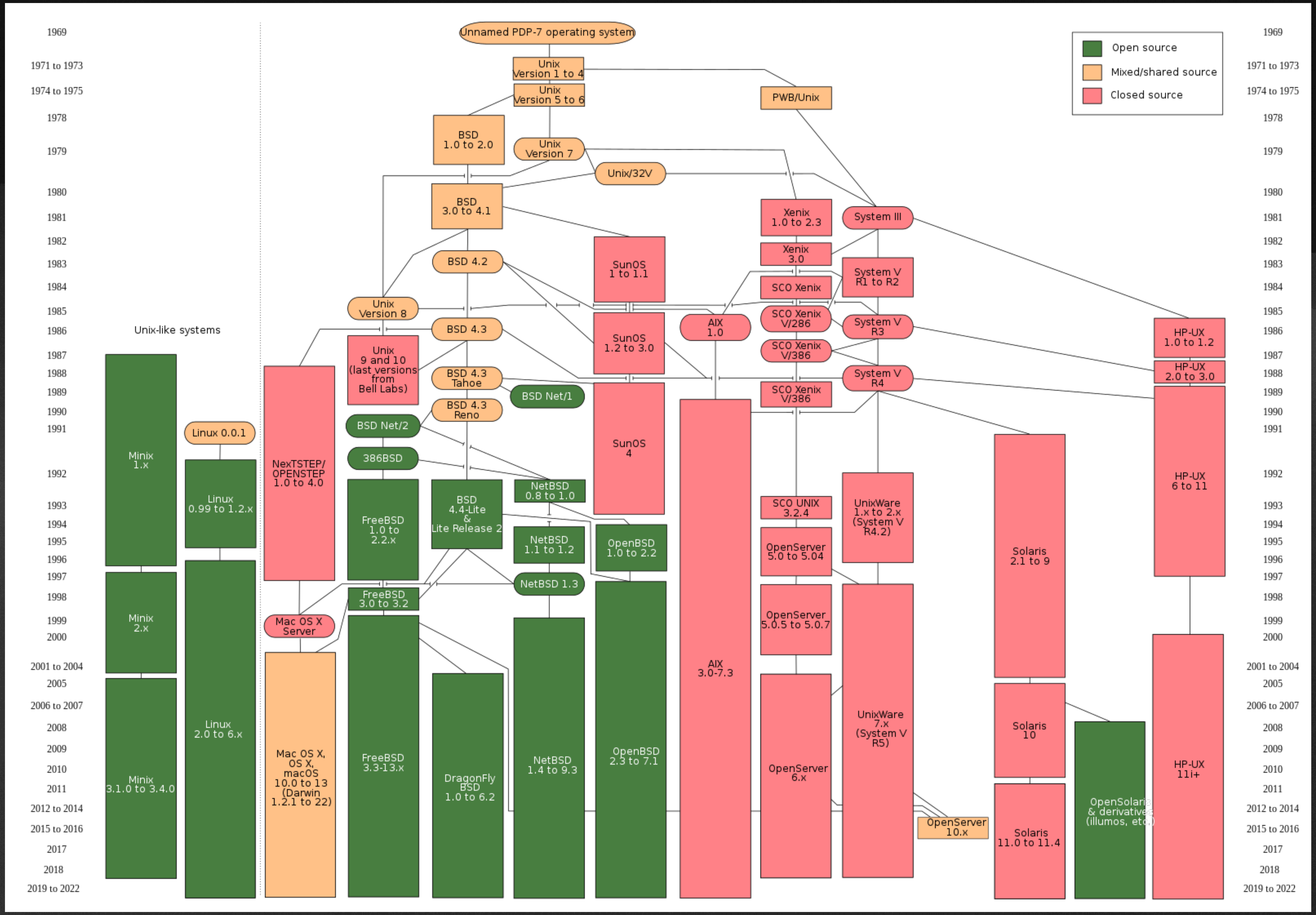
Topics for today

- What's Operating System?
- How OS helps your programs?
- Virtual Memory

Suting Chen

What's Operating System?

- Linux / Unix / Windows
- Debian / Fedora / Arch



Suting Chen

What Operating Systems do?

- Referee
- Glue
- Illusionist

What Operating Systems do?
Referee

- Protect your program from others
- Arrange system resources fairly

What Operating Systems do?
Glue

- Basic services: Loader (CALL)
- Provide hardware interfaces:
  - printf / scanf — Terminal
  - malloc — Memory
  - fopen — Disk
  - socket — Network card

Suting Chen

What Operating Systems do?

Glue — Importance

- Saves programmers' time to read docs and interact with hardware
- Offers unified interface for multiple devices:
  - `pthread_create()` works for all platforms — Intel, AMD, Apple...
  - `fopen()` opens file on all devices — SSD, HDD, even floppy disks!
- Enables permissions

```
chmod 750 /home/cs110
```

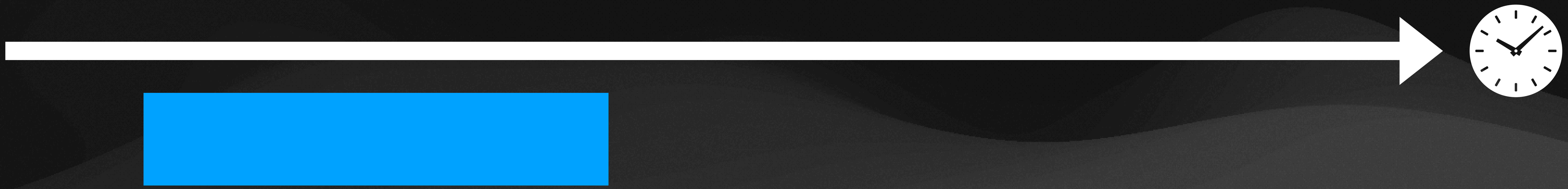
Suting Chen

What Operating Systems do?
Illusionist

- Each process believe that it have the full system resource, including:
  - Whole physical CPU core
  - Whole memory space
  - Direct access to any other resource as if there are no other processes

Illusionist — Whole physical CPU core

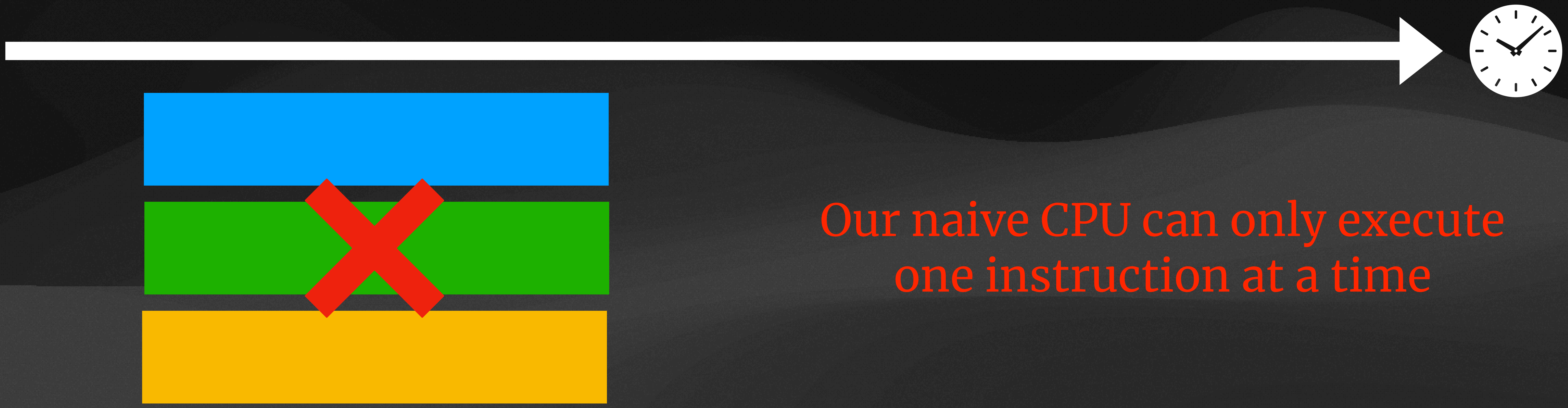
When running single program...



Suting Chen

Illusionist — Whole physical CPU core

When running multiple programs …

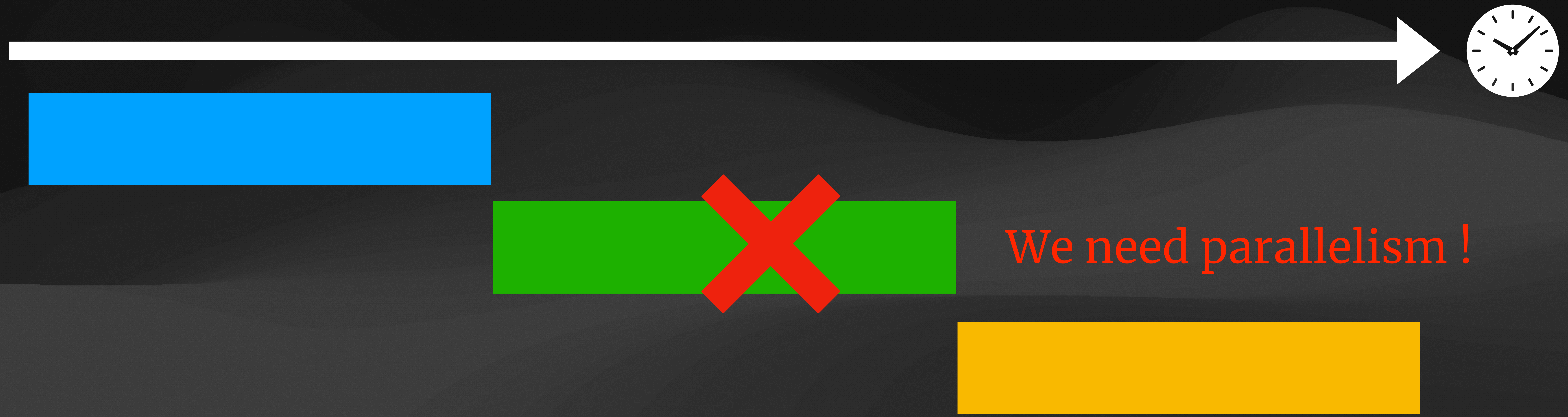


Our naive CPU can only execute one instruction at a time

Suting Chen

Illusionist — Whole physical CPU core

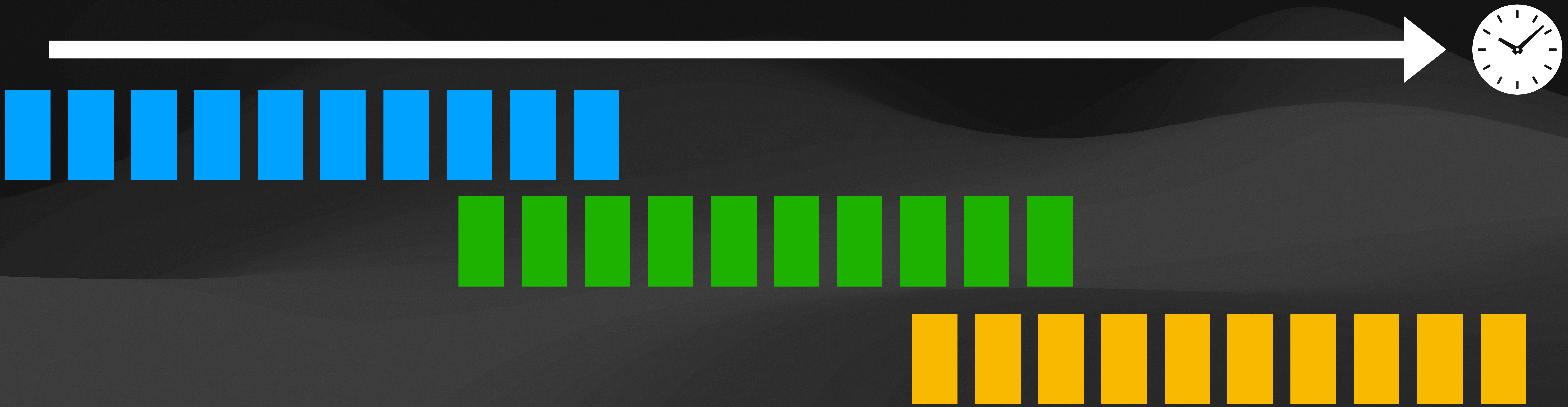
When running multiple programs ...



Suting Chen

Illusionist — Whole physical CPU core

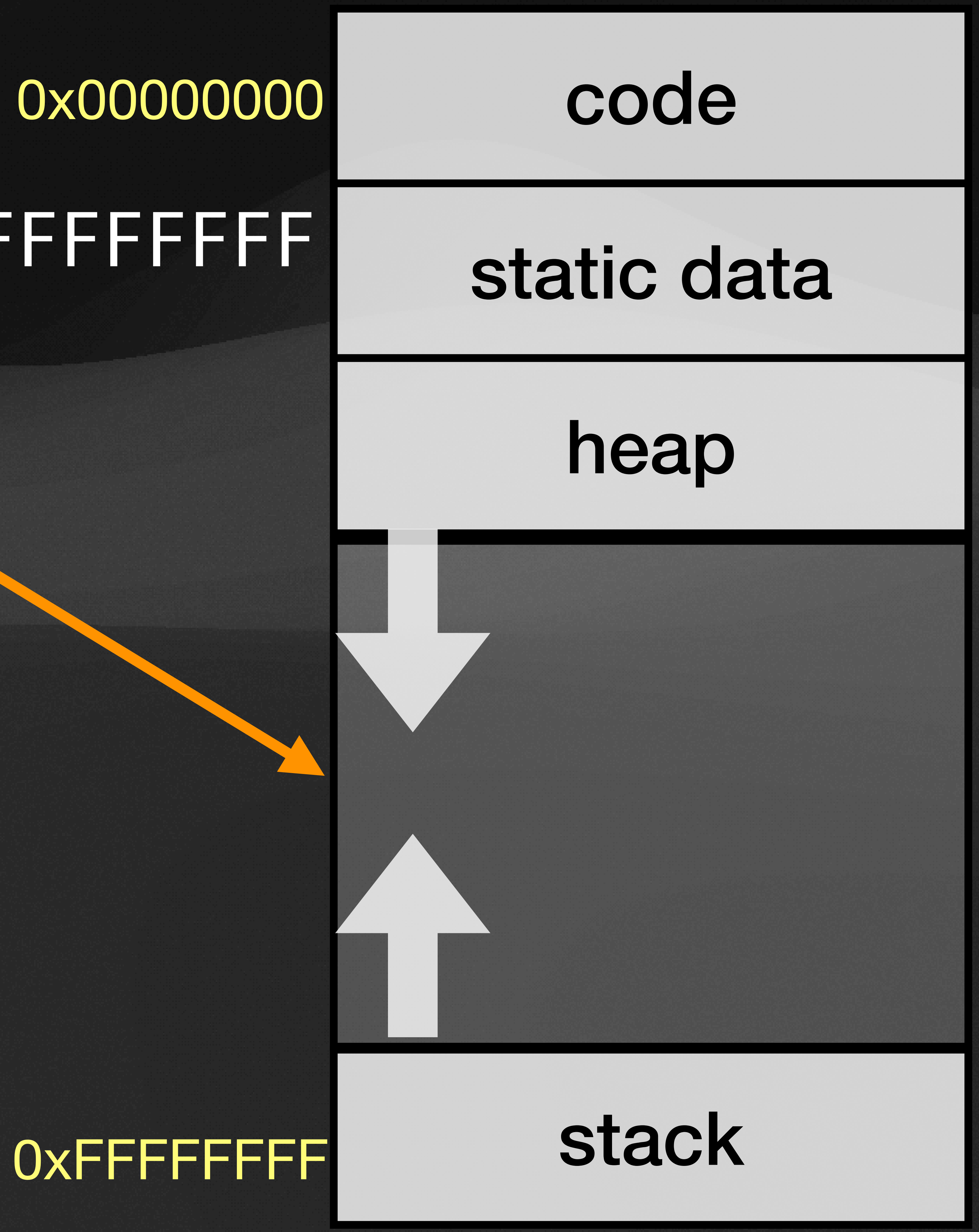
When running multiple programs ...



Suting Chen

Illusionist — Whole memory space

- Every program assert it has memory size of `0xFFFFFFFF`
- Problem:
  - We don't have so much memory to **waste**
  - We only have one 0x0 and one `0xFFFFFFFFF`



Suting Chen

Illusionist — Whole memory space

If you have a 64-bit machine (certainly you do!)

- 64-bit addresses!
- Total size 2^{64} Bytes $= 16$ EB $= 16 \times 1024$ PB
  - Impossible to find such memory (at least in 2023)

Suting Chen

实际上市面上的64位机器一般不会真的有64位的内存空间，这里为了演示方便忽略了很多问题
具体可以去看AMD64 Architecture Programmer's Manual，你也可以用lscpu查看自己的CPU情况
大部分情况下都是48位的Virtual Address Space（这里当故事听就好）

Virtual Memory

A modern solution

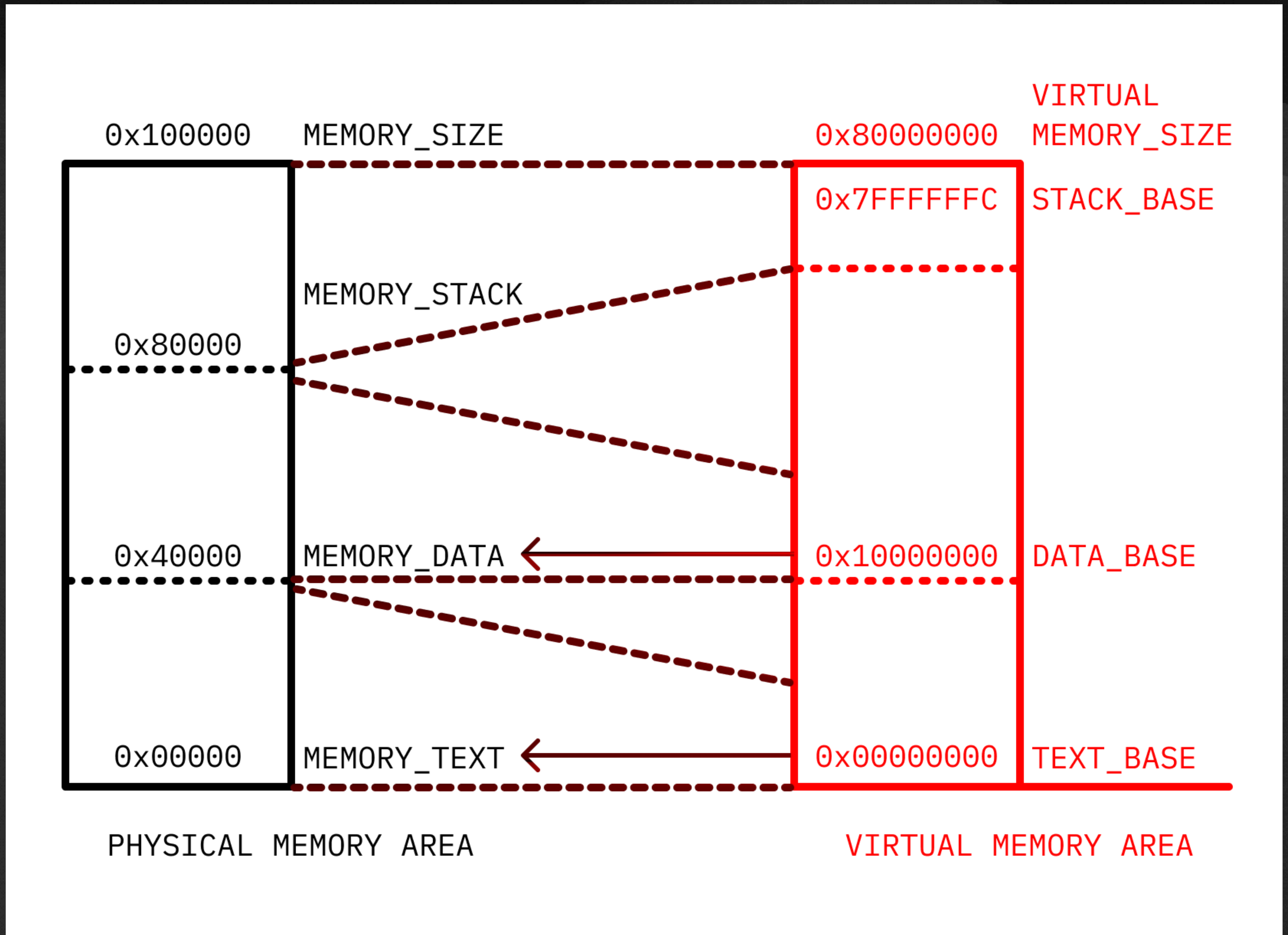
- Idea: hide physical address from programs
- Introduce Virtual Address Space

Suting Chen

Virtual Memory — Base and Bound

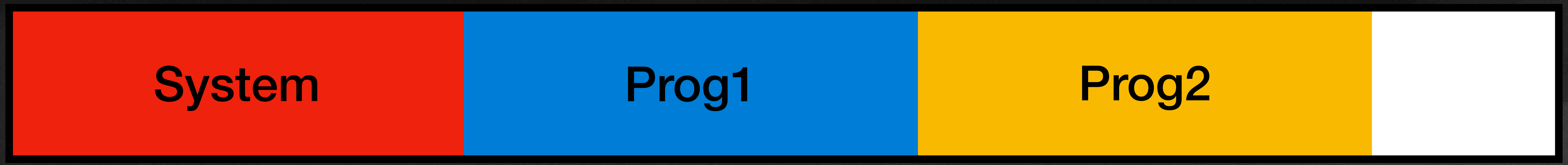
Segment	Virtual Base	Physical Base	Bound
Text	0x00000000	0x00000	0x40000
Data	0x10000000	0x40000	0x40000
stack	0x7FF80000	0x80000	0x20000
heap	-	-	-

Suting Chen



Virtual Memory — Base and Bound

Memory fragmentation



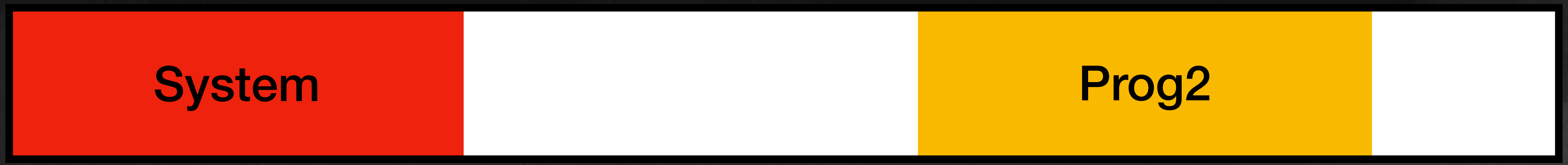
$$1GB - 300MB - 300MB - 300MB \;\; = 124MB$$

Program	System	Prog1	Prog2			
Size	300M	300M	300M			

Suting Chen

Virtual Memory — Base and Bound

Memory fragmentation



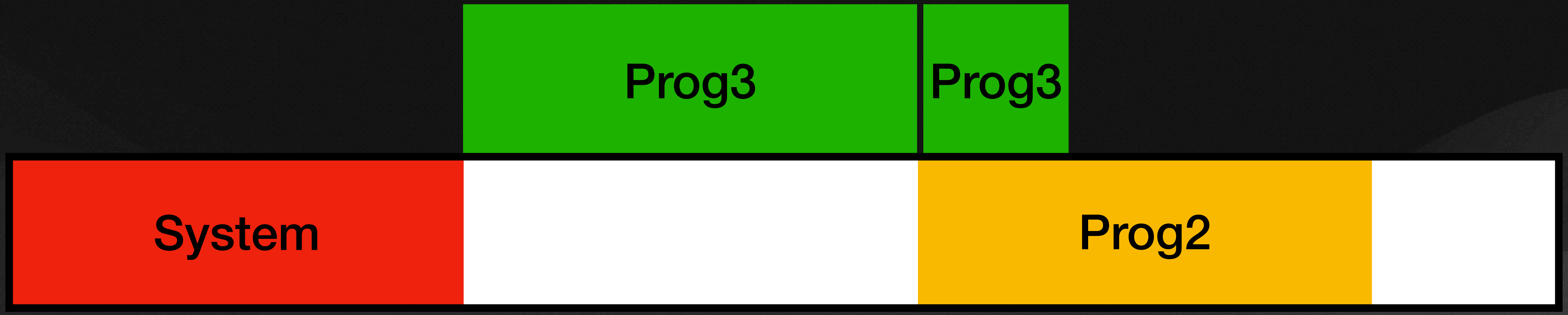
$$1GB - 300MB - 300MB \qquad = 424MB$$

Program	System	Prog1	Prog2			
Size	300M	-	300M			

Suting Chen

Virtual Memory — Base and Bound

Memory fragmentation



Temporary solution

(partition)

$$1GB - 300MB - 300MB - 400MB \; = 24MB$$

Program	System	Prog1	Prog2	Prog3		
Size	300M	-	300M	400M		

Suting Chen

Why not partition the whole memory?

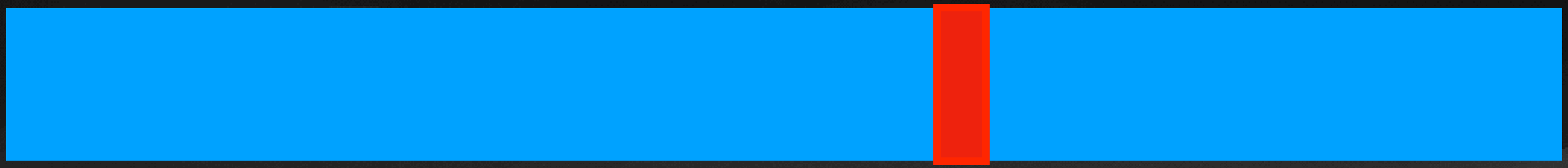
Introducing Pages

Pages

Pages
Basic idea

- See memory in pages (blocks) instead of bytes
- When a process need more memory, simply allocate one page
  - Avoid fragmentation

Pages
On my machine

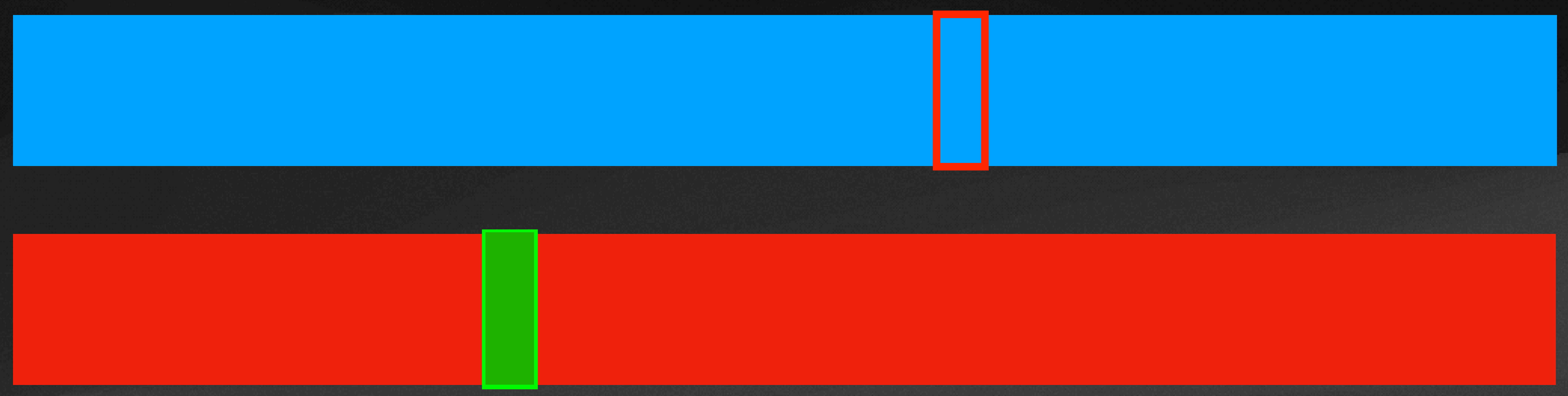


Whole memory (32GB)

Suting Chen

Pages

On my machine

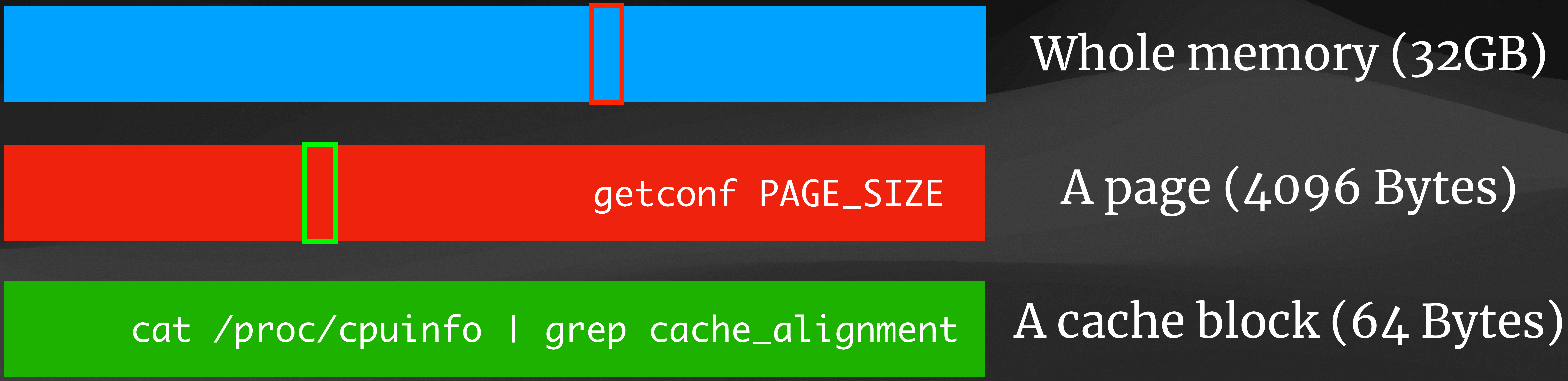


Whole memory (32GB)

A page (4096 Bytes)

Suting Chen

Pages
On my machine



Suting Chen

Virtual Memory — Pages

Harness pages in address translation

Virtual Page Number | Offset

Address translation ↓ ↓ copy

Physical Page Number | Offset

Virtual	Physical
0b00000	0b0110
0b00001	0b0100
0b00010	-
0b00011	-
0b00100	0b0001
...	...
0b11111	0b0000

Suting Chen

Virtual Memory — Pages

Harness pages in address translation

Virtual Page Number | Offset

Address translation ↓ ↓ copy

Physical Page Number | Offset

Simple Page Table

	Physical
0	0b0110
1	0b0100
2	-
3	-
4	0b0001
…	…
31	0b0000

Page Table Entry →

Suting Chen

Virtual Memory — Pages

Why NOT Simple Page Table?

32-bit Address, 4096 Bytes page, 4 Bytes PTE

- Calculate the size of simple page table:

1. $\#\ of\ offset\ bits = \log_2 4096 = 12$
2. $\#\ of\ bits\ of\ virtual\ page\ number = 32 - 12 = 20$
3. $\#\ of\ PTE = 2^{20}$
4. $Size\ of\ Page\ Table = 4 \times 2^{20} = 2^{22}\ Bytes = 4MB$

Suting Chen

For every single program!

Virtual Memory — Pages

Why NOT Simple Page Table?

48-bit Address, 4096 Bytes page, 4 Bytes PTE

- Calculate the size of simple page table:

1. $\# \ of \ offset \ bits = \log_2 4096 = 12$
2. $\# \ of \ bits \ of \ virtual \ page \ number = 48 - 12 = 36$
3. $\# \ of \ PTE = 2^{36}$
4. $Size \ of \ Page \ Table = 4 \times 2^{36} = 2^{38} \ Bytes = 256GB$

For every single program!

Suting Chen

Virtual Memory — Pages

We add hierarchy to Page Table

VPN1 | VPN2 | Offset

Address translation

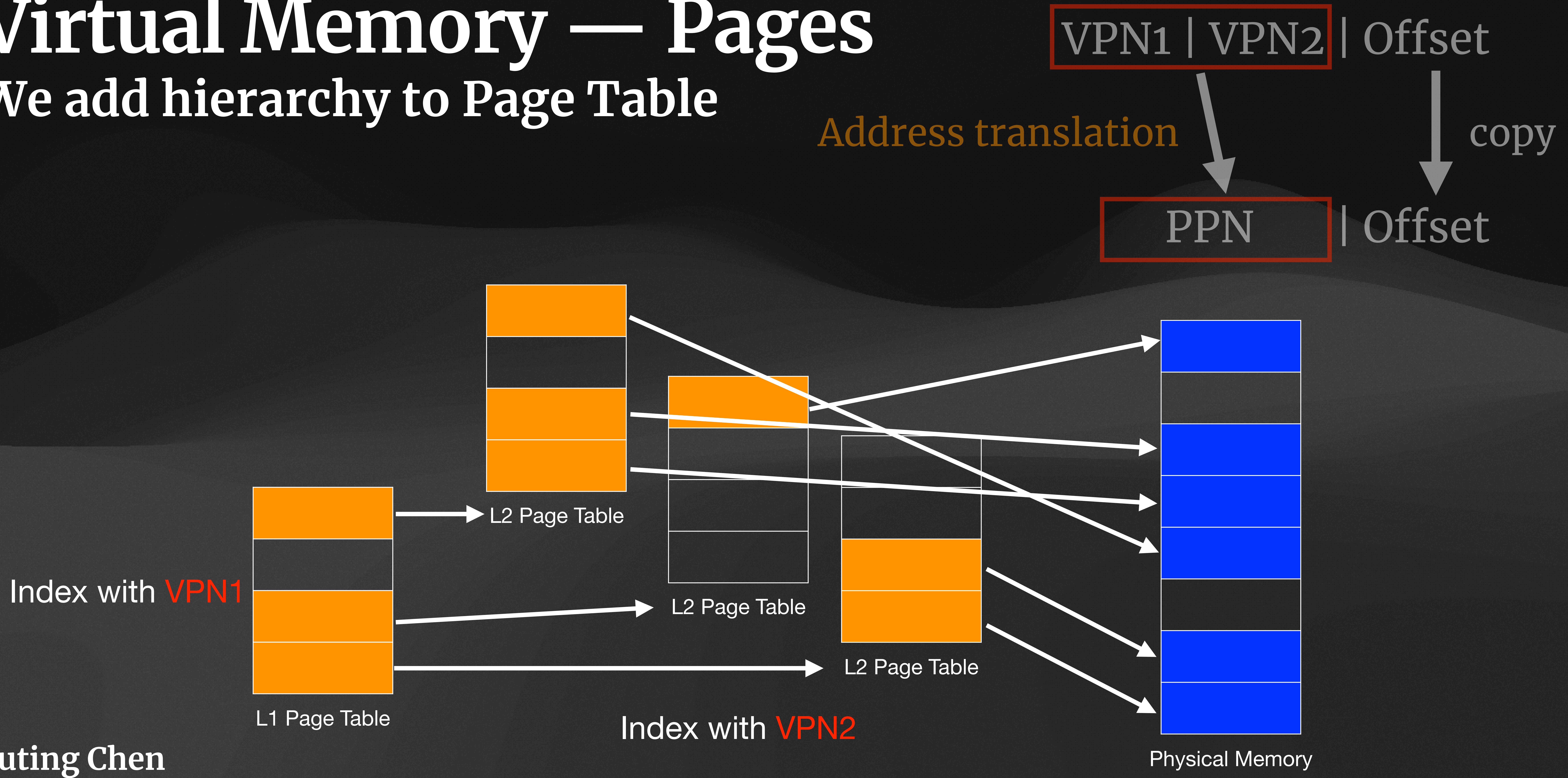
copy

PPN | Offset

Suting Chen

Virtual Memory — Pages

We add hierarchy to Page Table



Suting Chen

Virtual Memory — Pages

Let's do some easy calculation!

4 Bytes PTE

- Given length of each segment — VPN1: 10 | VPN2: 10 | Offset: 12

1. $Size\ of\ page = 2^{12}\ Bytes = 4096\ Bytes$
2. $\#\ of\ PTE\ in\ a\ L2\ page\ table = 2^{10}$
3. $Size\ of\ a\ L2\ page\ table = 2^{10} \times 4 = 2^{12} = 4096\ Bytes$

Suting Chen

Exercise

- Physical memory : 8GB
- Page size : 8KB
- Virtual address : 46-bit
- PTE : 4B

- Assume every page table exactly fits into a single page.
- How many levels of page tables would be required.

题目来源 https://www.geeksforgeeks.org/multilevel-paging-in-operating-system/，但我个人不太喜欢他的答案，建议来听dicussion

Virtual Memory — Translation Lookaside Buffer

Recall: Memory & (fully associative) Cache

	Data
0	0x34
1	0x7c
2	0x19
3	0x01
4	0x99
…	…
31	0x12

Tag	Data
01101101	0x91e989af
01000010	0xf4958104
11111011	0x82a0f265
01100101	0xfcf4ceef
00010101	0x88b5af65
10110101	0x190349ff
10011010	0x4358f247

Suting Chen

Virtual Memory — Translation Lookaside Buffer

(Simple) page table & TLB

	Physical
0	0x34
1	0x7c
2	0x19
3	0x01
4	0x99
…	…
31	0x12

Virtual	Physical
0b10010	0b0110
0b11101	0b0100
0b00010	0b0001
0b00011	0b1011
0b01100	0b0001
0b11111	0b1101
0b10011	0b0100

Suting Chen

Virtual Memory — Translation Lookaside Buffer

- Fully associative most of the time
- Can also apply replacement policy
- Each program has different address space, so invalidate TLB when switching

- TLB reach: Assume simple page table, 64 entries TLB, 4KB page
  - $TLB\ reach = 64 \times 4\ KB = 256\ KB$

Suting Chen