OS & Virtual Memory

Week 14

Suting Chen

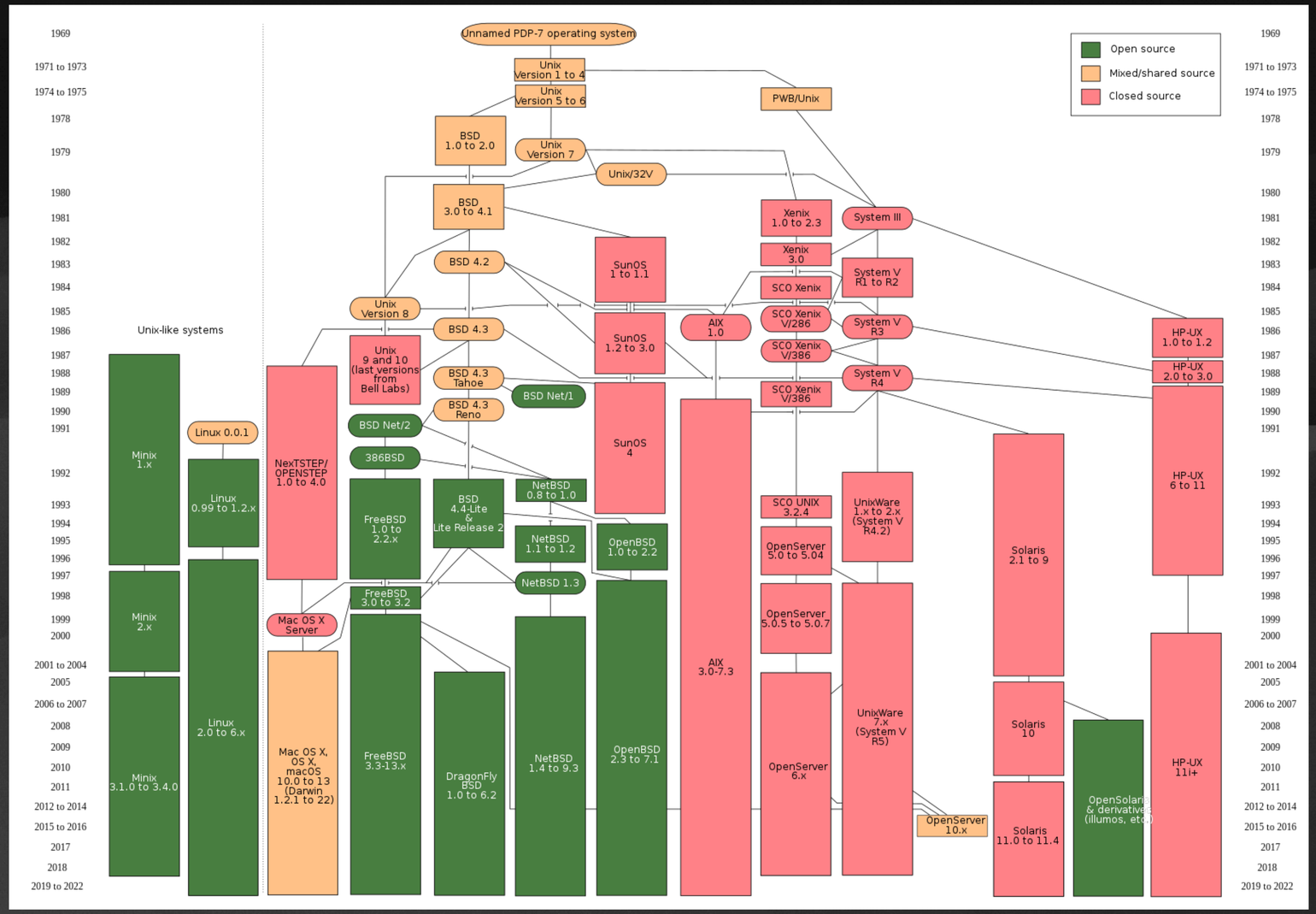
OS & Virtual Memory

- 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’s Operating System?

	目标纯粹 必须管理一台计算机	目标中立 管理计算机硬件就行	目标混乱 管理什么都行
对象纯粹 必须服务二进制代码	 Windows/Linux/vxWorks 当然是操作系统	 固件也是操作系统	 gdb也是操作系统
对象中立 服务对象是程序就行	 浏览器/微信/支付宝/JVM 都是操作系统	 Hadoop也是操作系统	 资源管理器也是操作系统
对象混乱 服务谁都可以	 机箱是操作系统	 机房也可以算操作系统	 校长为什么不是操作系统?

Suting Chen

How OS helps your programs?

- Helps loading your program (CALL)
- Offers a set of functions
- Gives an illusion of the system
- Schedule your threads fairly

How OS helps your programs?

Helps loading your program (CALL)

- Linux load a program in ELF format from disk to memory.
- Make space for stack and heap.

Suting Chen

How OS helps your programs?

Offers a set of functions

- How to print to console, read from input, allocate memory?

- printf( ), scanf( ), malloc( )

How OS helps your programs?

Offers a set of functions — interacting with hardware

- printf / scanf — Keyboard
- malloc — Memory
- fopen — Disk
- socket — Network card
- pthread_create — CPU

Suting Chen

How OS helps your programs?

Offers a set of functions — pros

- Saves programmers’ time to read docs and interact with hardware

How OS helps your programs?

Offers a set of functions — pros

- 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 and even floppy disks!

How OS helps your programs?

Offers a set of functions — pros

- 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 and even floppy disks!
- Enables permissions

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

How OS helps your programs?

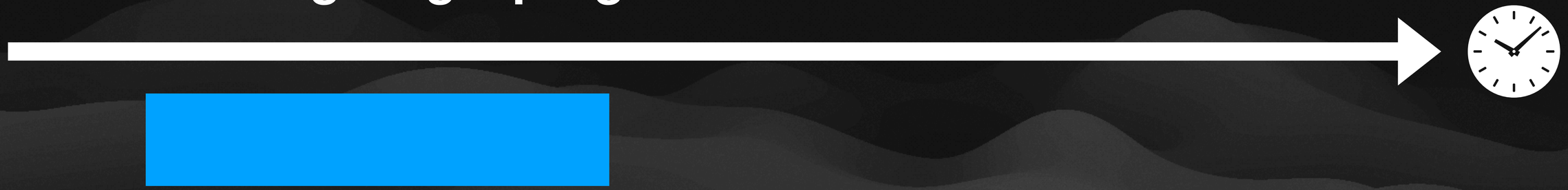
Gives an illusion of the system

- A real CPU
- Ideal memory addresses

Suting Chen

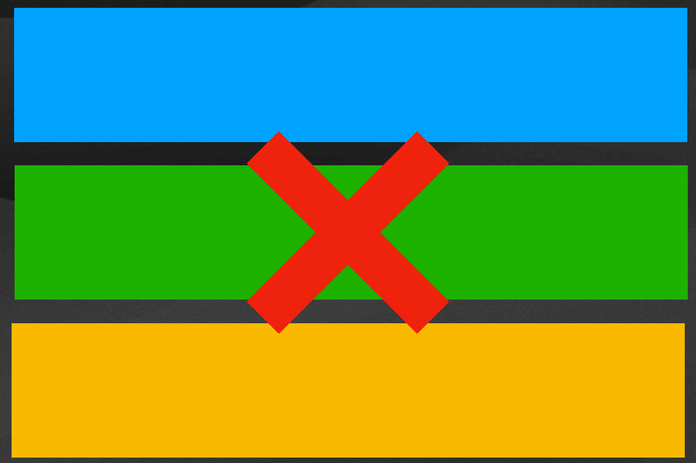
Gives an illusion of the system — A real CPU

When running single program…



Gives an illusion of the system — A real CPU

When running multiple programs…



Our naive CPU can only execute one instruction at a time

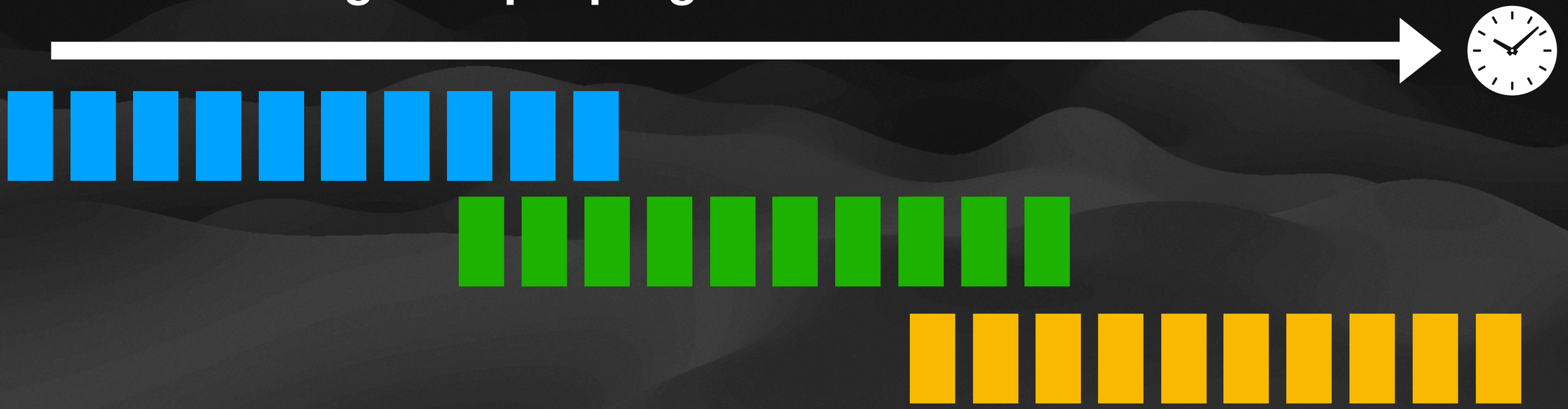
Gives an illusion of the system — A real CPU

When running multiple programs…

Suting Chen

Gives an illusion of the system — A real CPU

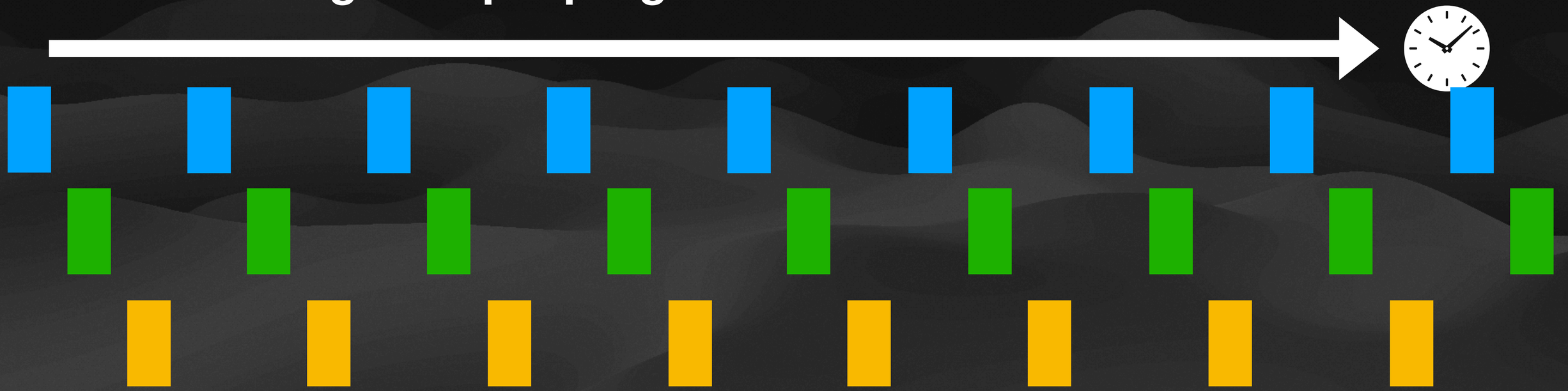
When running multiple programs…



Suting Chen

Gives an illusion of the system — A real CPU

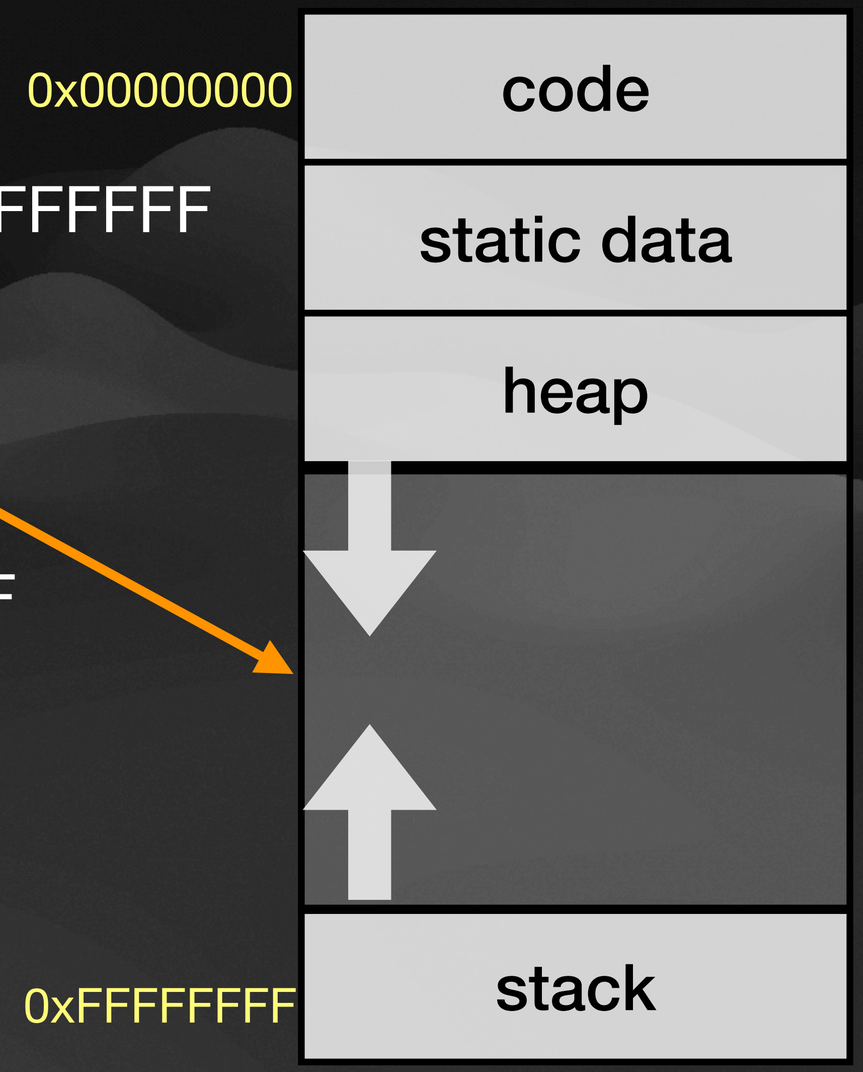
When running multiple programs…



Suting Chen

Gives an illusion of the system — Ideal memory addresses

- 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 0xFFFFFFFF



Gives an illusion of the system — Ideal memory addresses

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

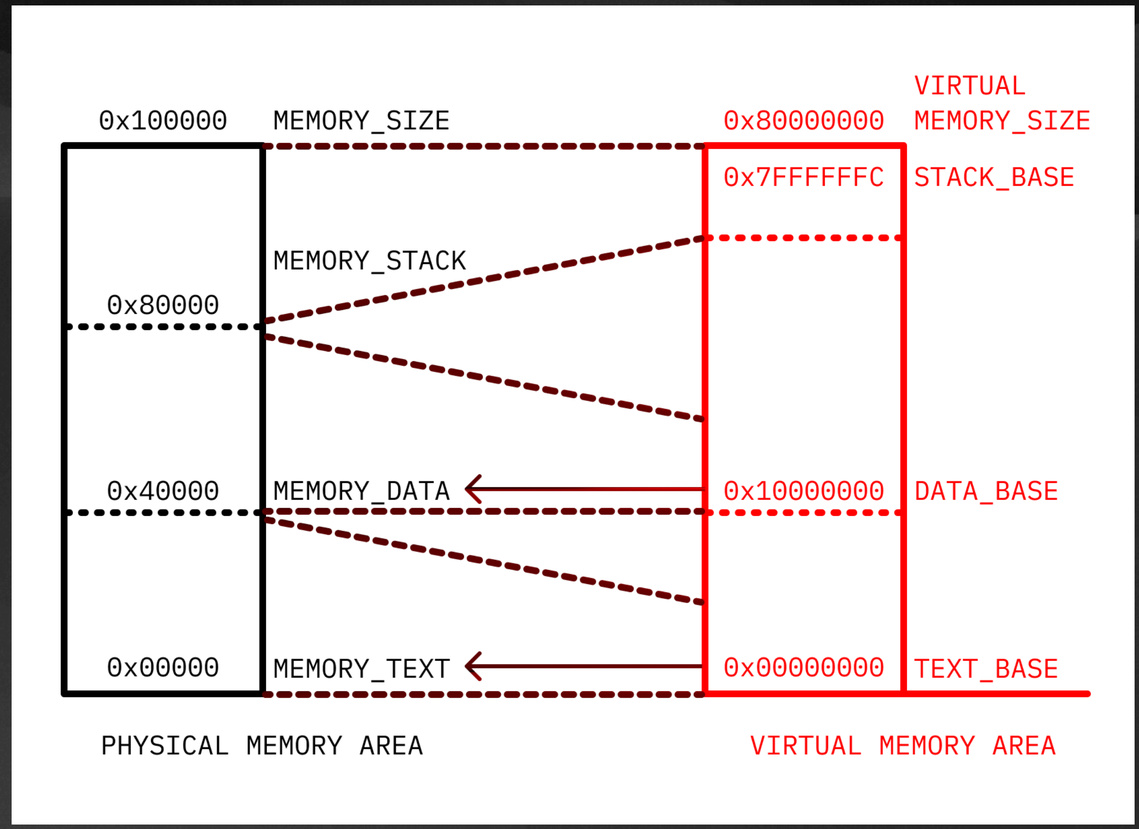
- 64-bit addresses!
- Total size $2^{64}\ Bytes = 16\ EB = 16 \times 1024\ PB$
    - Impossible to find such memory (at least in 2023)

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 very modern solution

- Idea: hide physical address from programs
- Introduce Virtual Address Space
- Project 1.2 ➡️

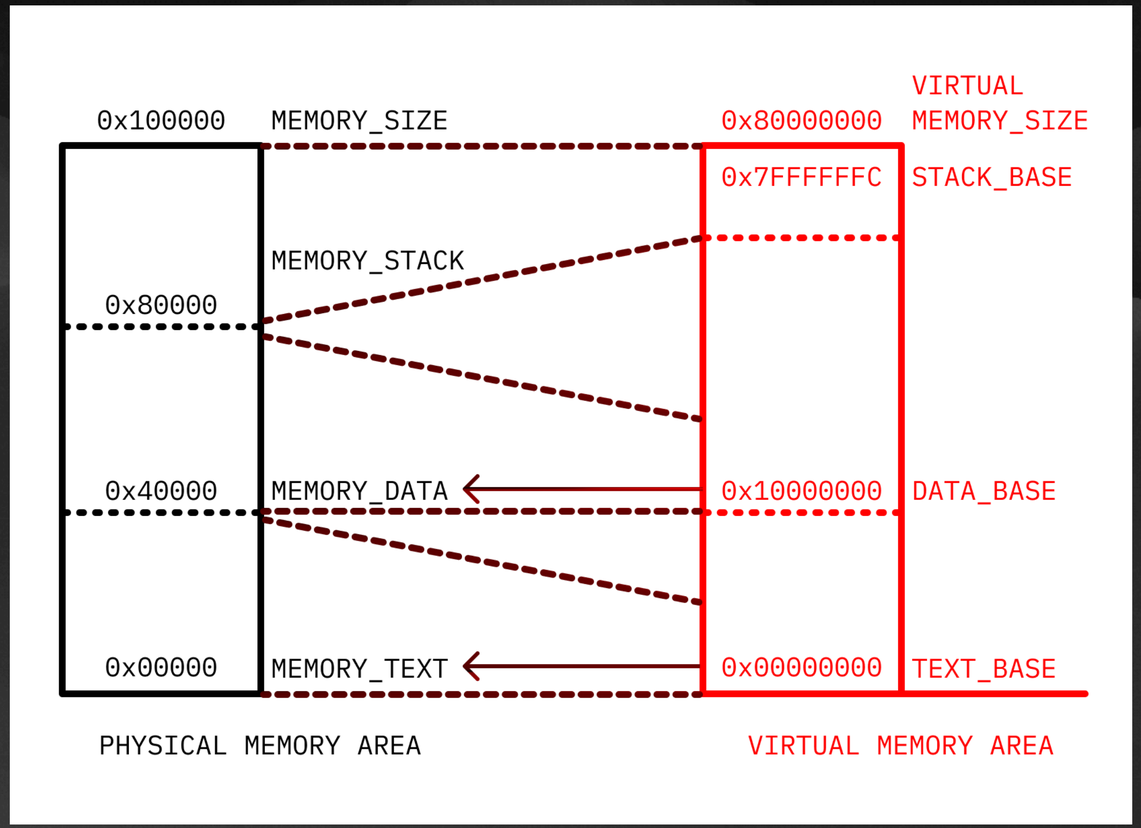


Suting Chen

Virtual Memory

Base and Bound

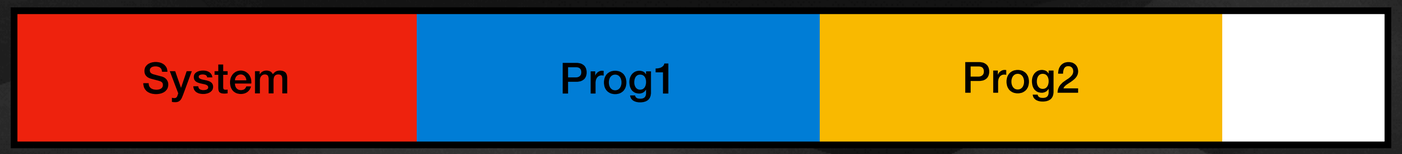
Segment	Virtual Base	Physical Base	Bound
Text	0x00000000	0x00000	0x40000
Data	0x10000000	0x40000	0x40000
stack	0x7FF80000	0x80000	0x20000
heap	-	-	-



Suting Chen

Virtual Memory — Base and Bound

Memory fragmentation



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

Program	System	Prog1	Prog2			
Size	300M	300M	300M			

Virtual Memory — Base and Bound

Memory fragmentation

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

Program	System	Prog1	Prog2			
Size	300M	-	300M			

Suting Chen

Virtual Memory — Base and Bound

Memory fragmentation

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

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

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		

Why not partition the whole device?

Introducing Pages

Pages

Something similar with cache blocks

- See memory in pages (blocks) instead of bytes

- When a process need more memory, simply allocate one page

Pages

On my machine

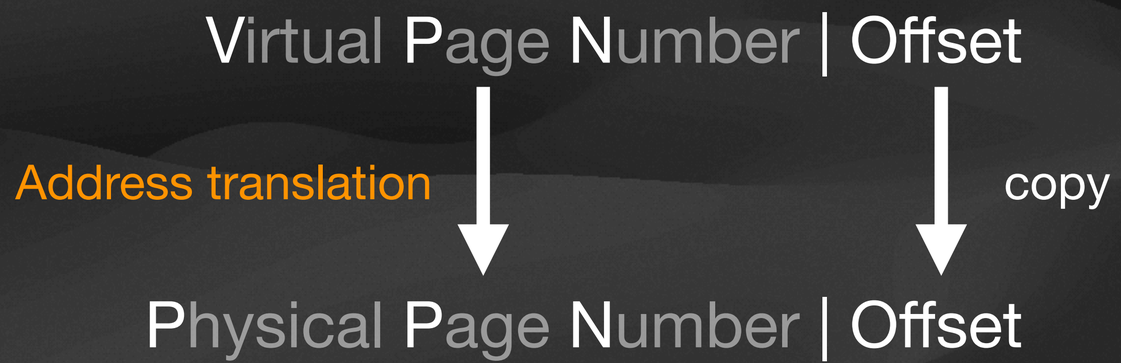
Whole memory (32GB)

A page (4096 Bytes)

A cache block (64 Bytes)

Virtual Memory — Pages

Harness pages in address translation

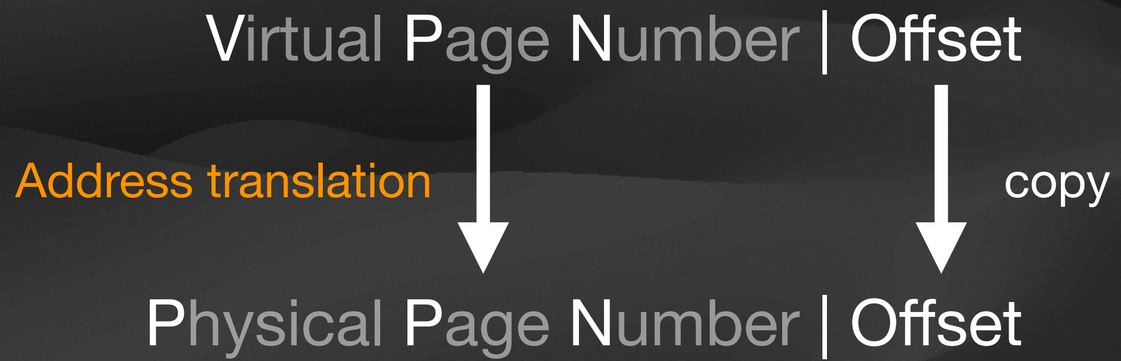


Virtual	Physical
0b00000	0b0110
0b00001	0b0100
0b00010	-
0b00011	-
0b00100	0b0001
…	…
0b11111	0b0000

Suting Chen

Virtual Memory — Pages

Harness pages in address translation



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

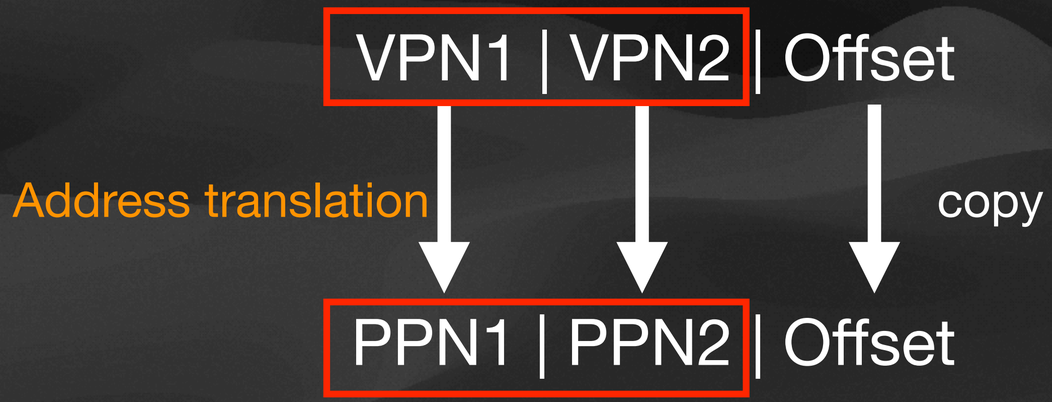
- Calculate the size of simple page table:

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!

Virtual Memory — Pages

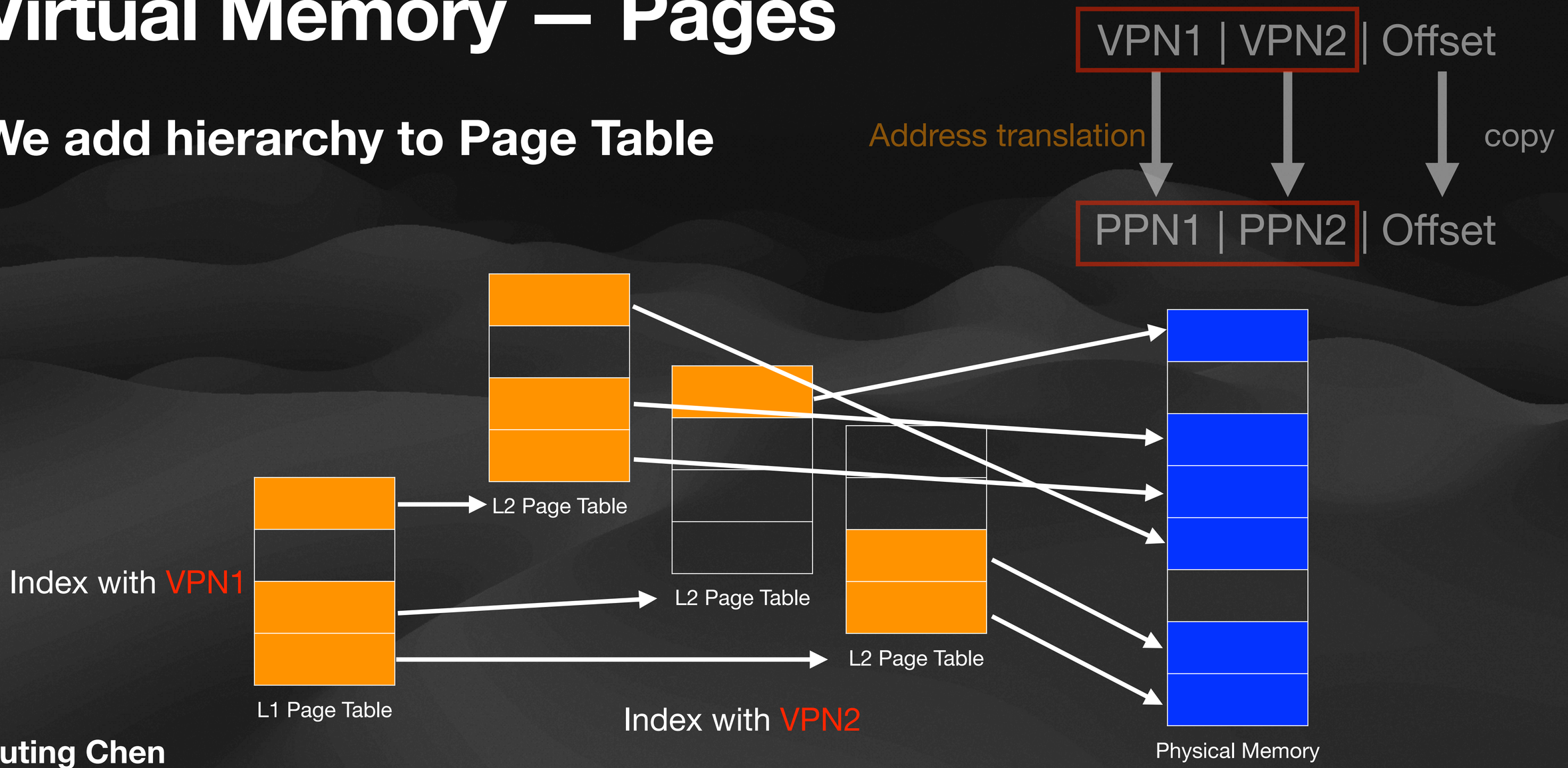
We add hierarchy to Page Table



Suting Chen

Virtual Memory — Pages

We add hierarchy to Page Table



Suting Chen

Virtual Memory — Pages

Let’s do some easy calculation!

- 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. $Size\ of\ a\ L2\ page\ table = 2^{10} \times 4 = 2^{12} = 4096\ Bytes$

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.

https://www.geeksforgeeks.org/multilevel-paging-in-operating-system/

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

Virtual Memory — Translation Lookaside Buffer

(Simple) page table & TLB

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

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

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
  - $TLB\ reach = 64 \times 4\ KB = 256\ KB$

Remainders

1. P&H (our textbook) chapter 5
2. More on OS next week
3. There will be exercise on VM, so don’t worry.
4. All my slides are here
5. No plagiarism! Very serious consequences!