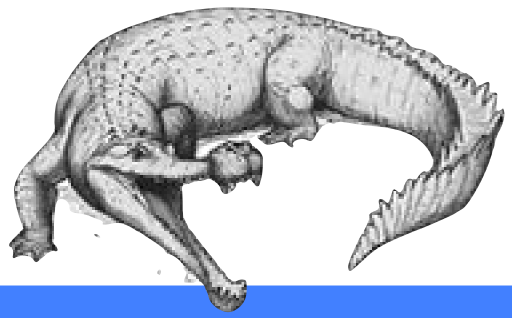
2020春

LINUX OPERATING SYSTEM

YANG

LINUX操作系统（双语）

双语课→课件内容中英混排



Lecture 14

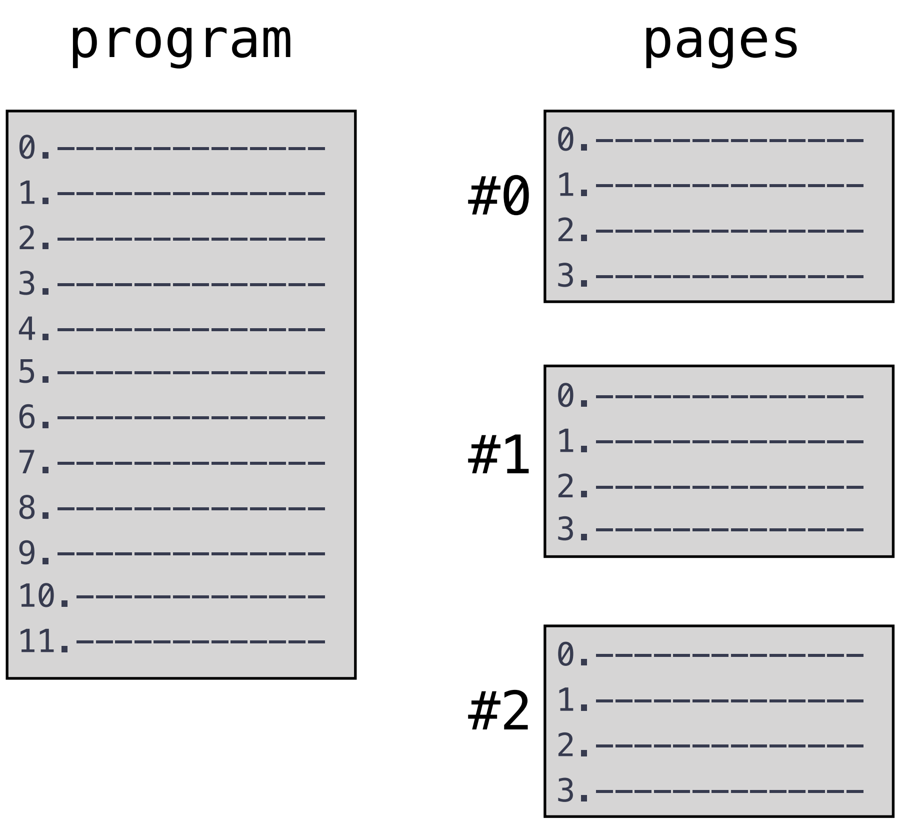
Page Table

本讲内容

- 页表
- 快表
- 基于页的保护与共享
- 多级页表

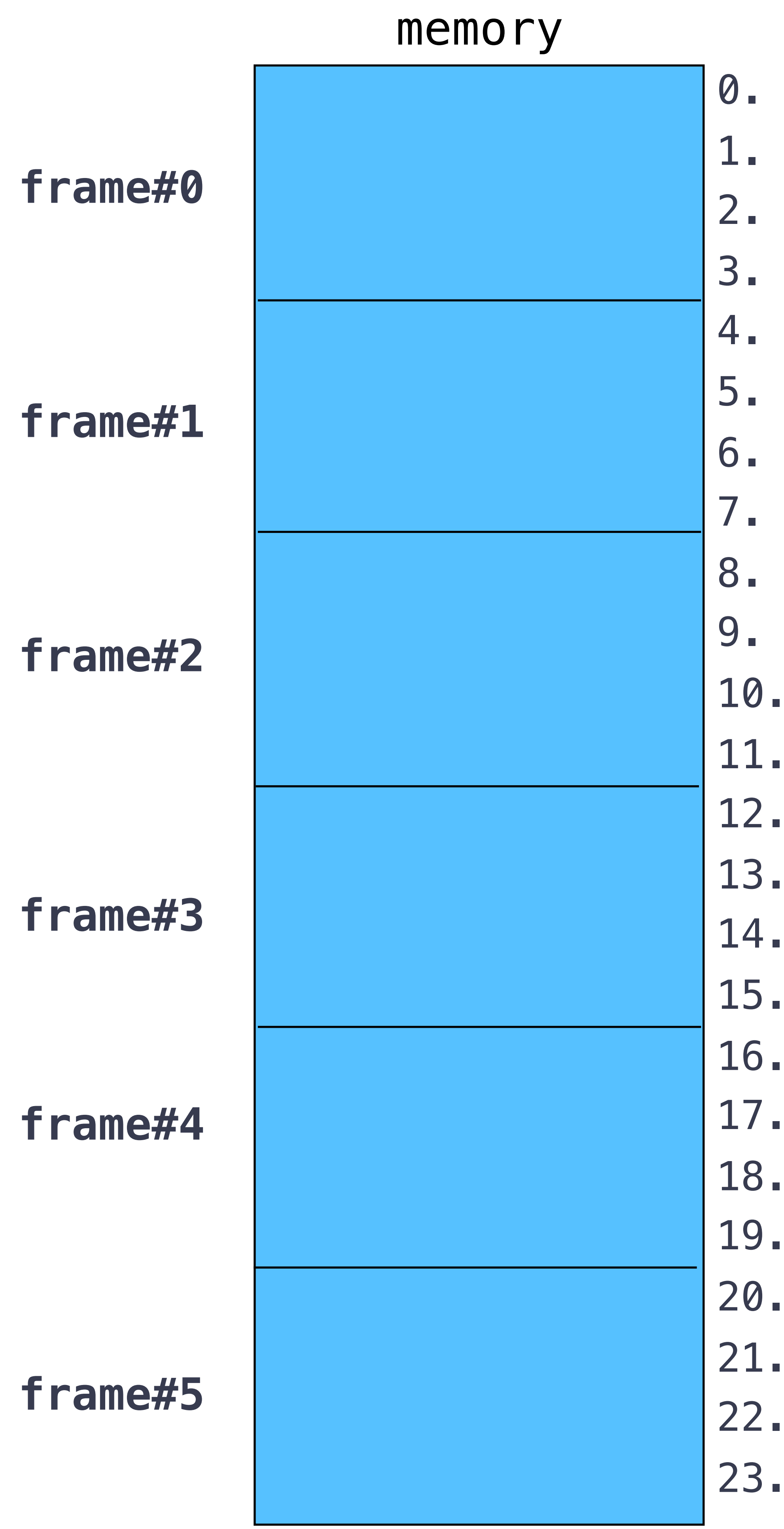
页表

分页

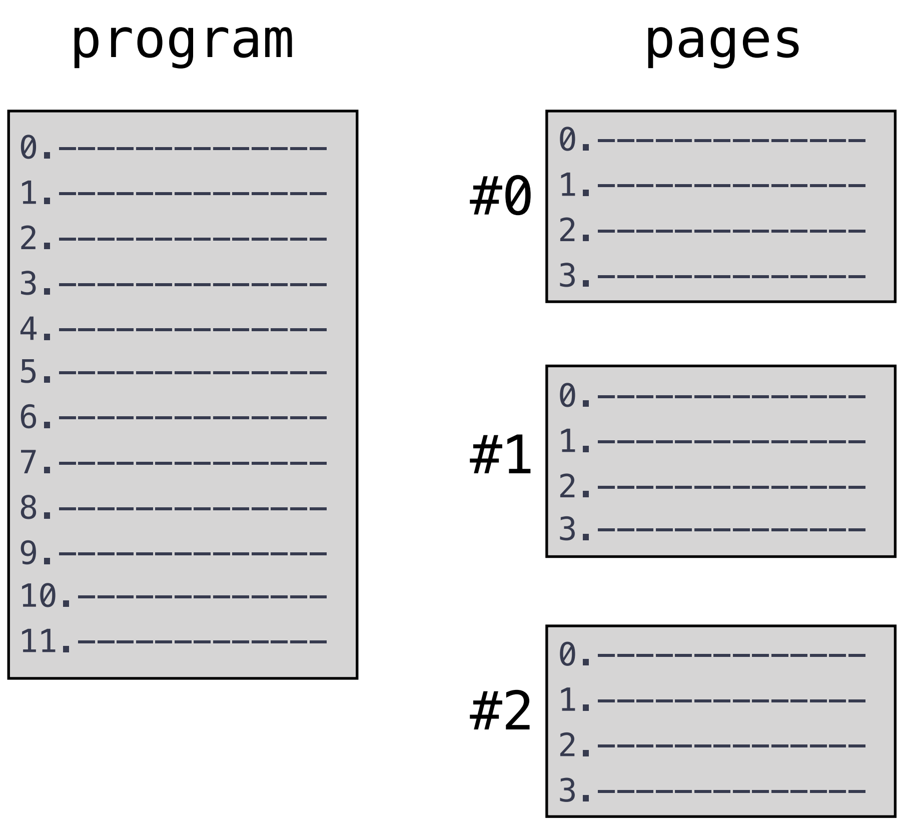


Frame Table

frame no.	state
0	0
1	0
2	0
3	0
4	0
5	0



分页



Frame Table

frame no.	state
0	1
1	0
2	1
3	0
4	0
5	1

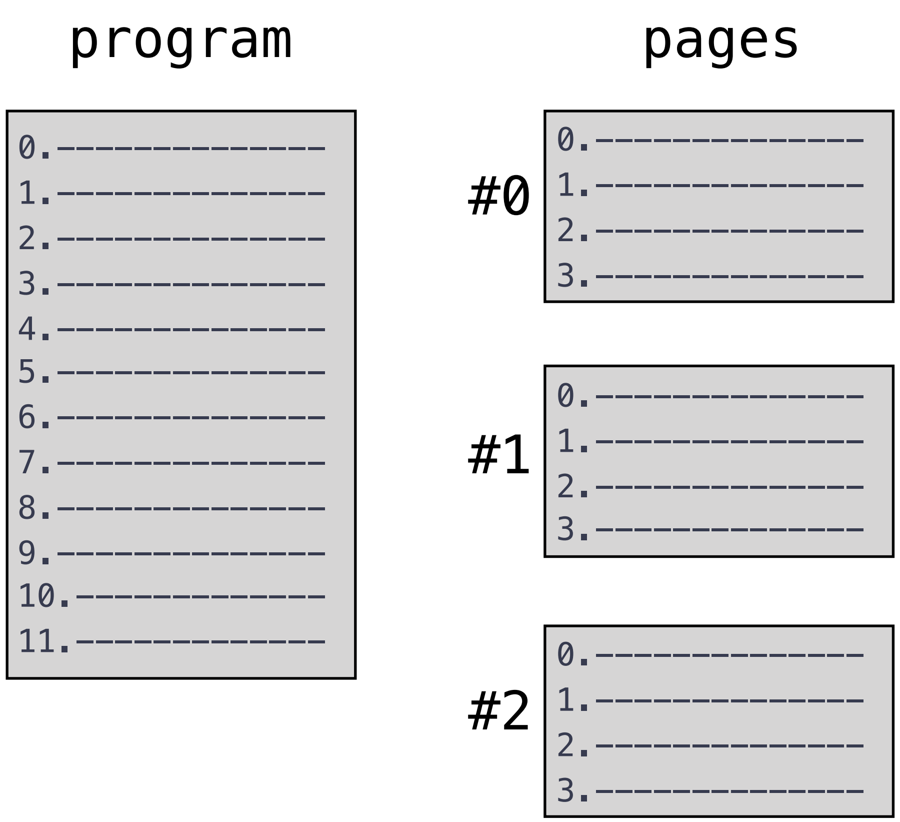
Page Table

page no.	frame no.
0	5
1	0
2	2

页号	页内位移

逻辑地址

分页

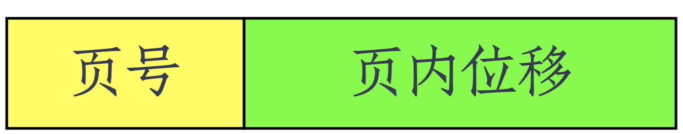


Frame Table

frame no.	state
0	1
1	0
2	1
3	0
4	0
5	1

Page Table

	frame no.
0	5
1	0
2	2



逻辑地址

页面大小

页号	页内位移

逻辑地址

- 若逻辑地址长度为m bits，页面大小：2^n Bytes
  - 页内位移占n bits
  - 页号占m-n bits
- 获取Linux系统页大小的命令

```
youngyt@youngyt-PC:/$ uname -m
x86_64
youngyt@youngyt-PC:/$ getconf PAGESIZE
4096
```

  - 请分别给出m和n的值

PAGE TABLE

- The operating system maintains a copy of the page table for each process.
- This copy is used to translate logical addresses to physical addresses.
- It is also used by the CPU dispatcher to define the hardware page table when a process is to be allocated the CPU.
- Paging therefore increases the context-switch time.

快表

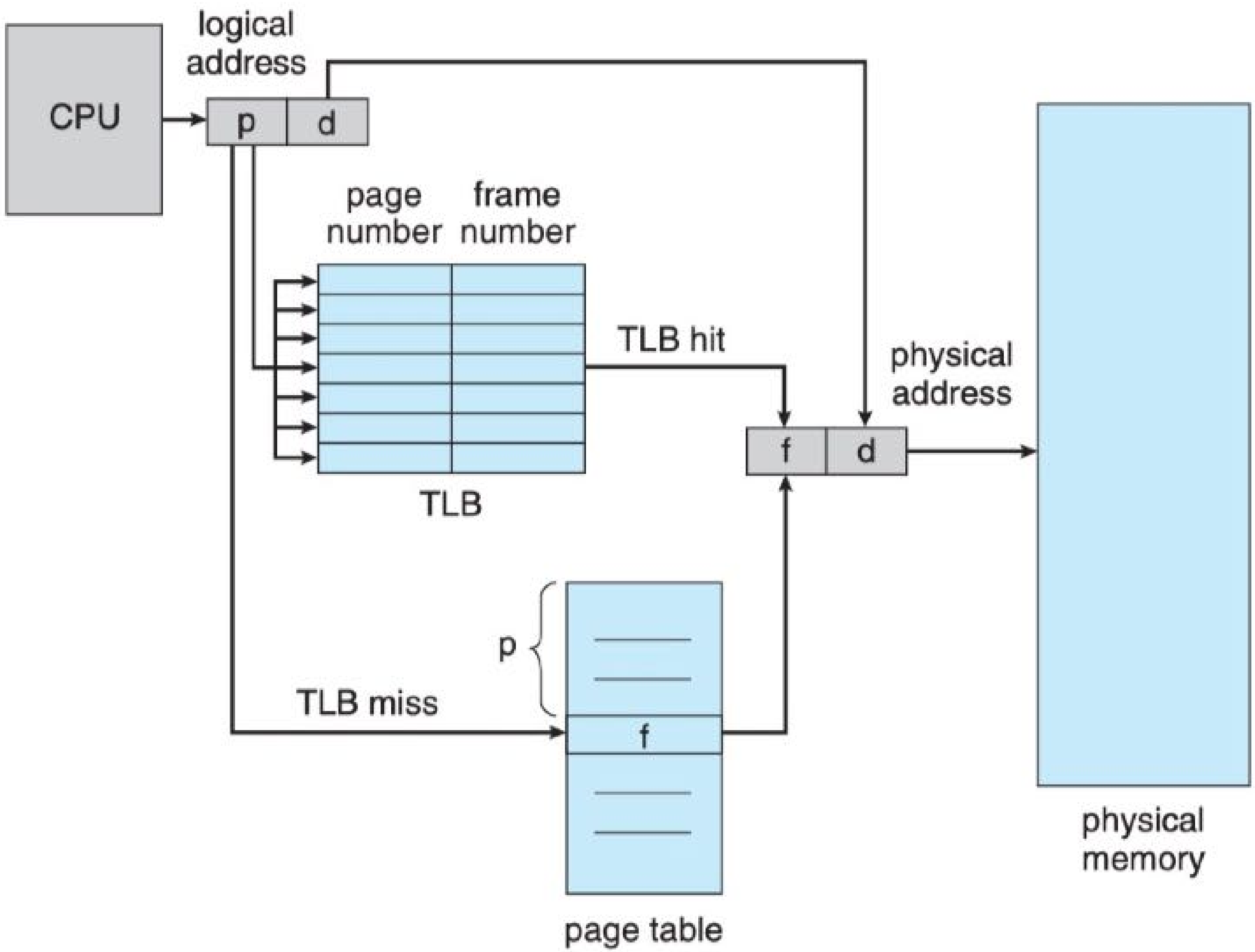
HARDWARE PAGE TABLE

- The page table is kept in main memory, and a page-table base register (PTBR) points to the page table.
- Changing page tables requires changing only this one register, substantially reducing context-switch time.
- With this scheme, two memory accesses are needed to access a byte (one for the page-table entry, one for the byte).

TLB

- TLB(Translation Look-aside Buffer) is a kind of small, fast-lookup hardware cache. It is used with page tables in the following way.
  - The TLB contains only a few of the page-table entries.
  - When a logical address is generated by the CPU, its page number is presented to the TLB.
  - If the page number is found, its frame number is immediately available and is used to access memory.
  - If TLB miss, a memory reference to the page table must be made.

PAGING WITH TLB



TLB HIT RATIO

- The percentage of times that the page number of interest is found in the TLB is called the hit ratio.
- An 80-percent hit ratio, for example, means that we find the desired page number in the TLB 80 percent of the time. If it takes 100 nanoseconds to access memory, please find the effective memory-access time.

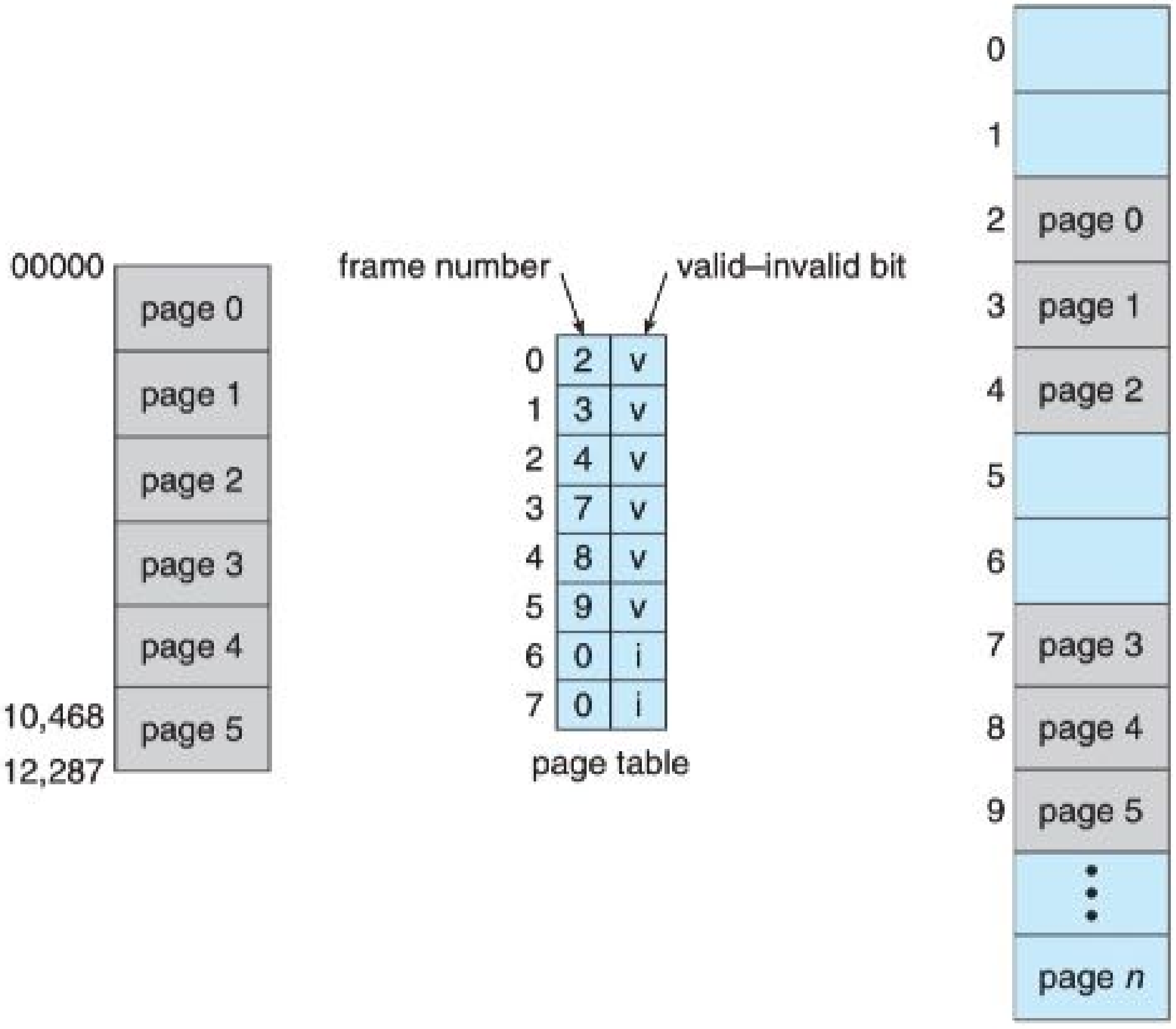
```
effective access time = 0.80 × 100 + 0.20 × 200 = 120 ns
```

- How about 99% hit ratio?

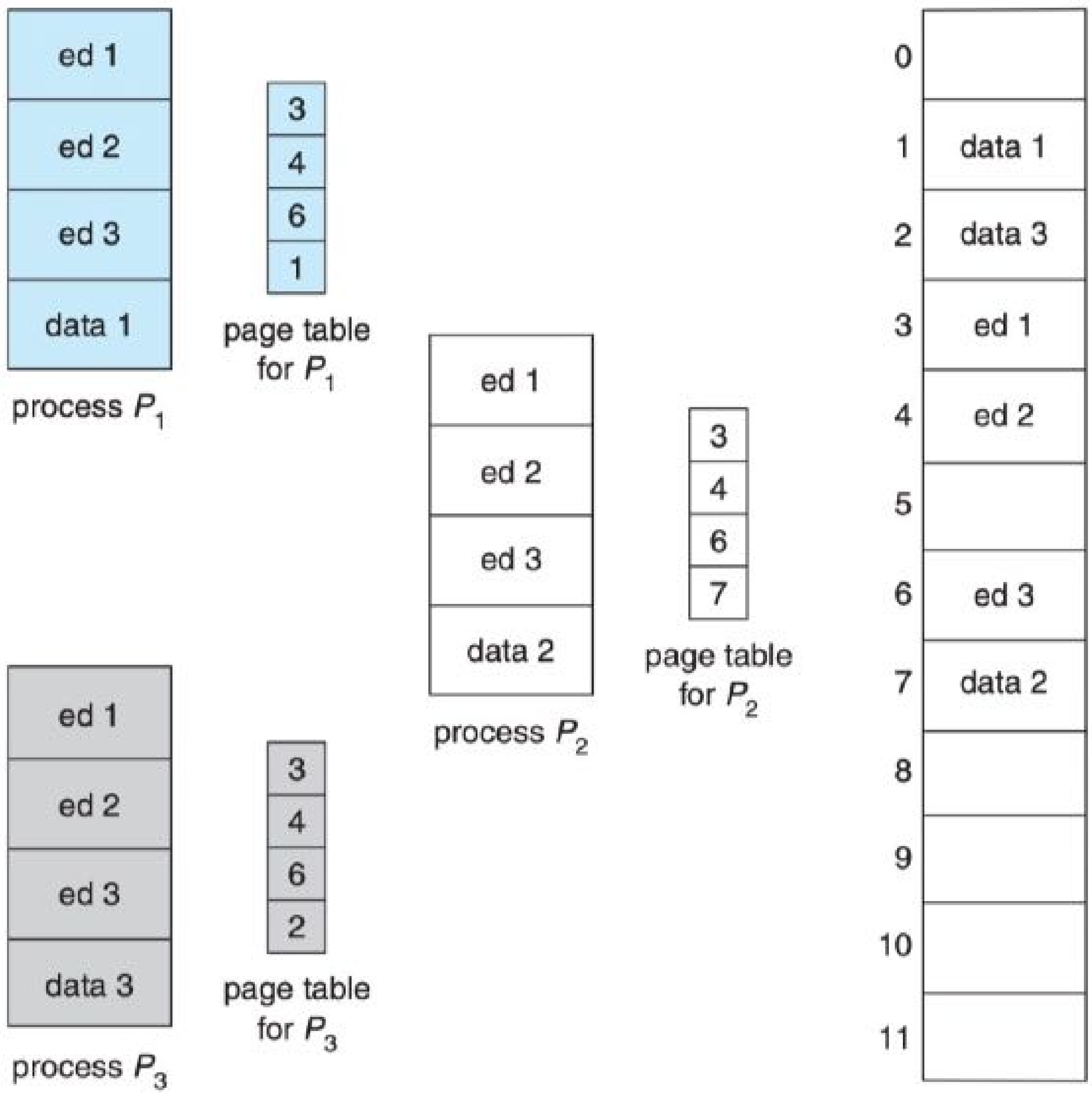
基于页的保护与共享

保护

- 为了防止地址转换时出现异常，可在页表每个条目设置一个“valid-invalid”比特位，用于表示该页的有效性。
- 这个方法可以被轻松扩展以提供更好的保护级别，如“只读”、“读写”、“可执行”等。



共享



ed 1
ed 2
ed 3
data 1
process P_1
3
4
6
1
page table for P_1
ed 1
ed 2
ed 3
data 2
process P_2
3
4
6
7
page table for P_2
ed 1
ed 2
ed 3
data 3
process P_3
3
4
6
2
page table for P_3
0
1 data 1
2 data 3
3 ed 1
4 ed 2
5
6 ed 3
7 data 2
8
9
10
11

多级页表

页表大小

- 假设CPU是32bits，采用的逻辑地址是32bits，那么进程的逻辑地址空间大小为2^{32}Bytes，即4G Bytes。
  - 若页面大小是4K Bytes，则一个进程最多被分成1M个页面，也就是说进程的页表最多有1M个页表项；
  - 若每个页表项占用4Bytes，则每个页表最多占用4MBytes空间（1K个连续页框）。
  - 如何解决“连续”的困扰？

页表页

pages
P#0
P#1
P#2
P#3
P#4
P#5
P#6
P#7

frames
F#0
F#1
F#2
F#3
F#4
F#5
F#6
F#7
F#8
F#9
F#10
F#11
F#12
F#13
F#14
F#15
F#16
F#17
F#18
F#19
F#20
F#21
F#22
F#23

页表页

pages

P#0
P#1
P#2
P#3
P#4
P#5
P#6
P#7

page table

P#0	F#3
P#1	F#6
P#2	F#12
P#3	F#9
P#4	F#1
P#5	F#8
P#6	F#11
P#7	F#0

frames

F#0
F#1
F#2
F#3
F#4
F#5
F#6
F#7
F#8
F#9
F#10
F#11
F#12
F#13
F#14
F#15
F#16
F#17
F#18
F#19
F#20
F#21
F#22
F#23

页表页

pages

P#0
P#1
P#2
P#3
P#4
P#5
P#6
P#7

page table

P#0	F#3
P#1	F#6
P#2	F#12
P#3	F#9
P#4	F#1
P#5	F#8
P#6	F#11
P#7	F#0

page table directory

PPT#0	F#7
PPT#1	F#10

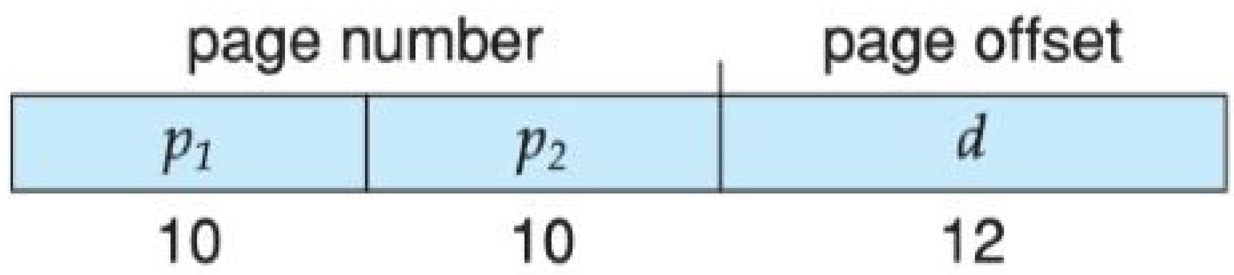
frames

F#0
F#1
F#2
F#3
F#4
F#5
F#6
F#7
F#8
F#9
F#10
F#11
F#12
F#13
F#14
F#15
F#16
F#17
F#18
F#19
F#20
F#21
F#22
F#23

页表页号	页号	页内位移

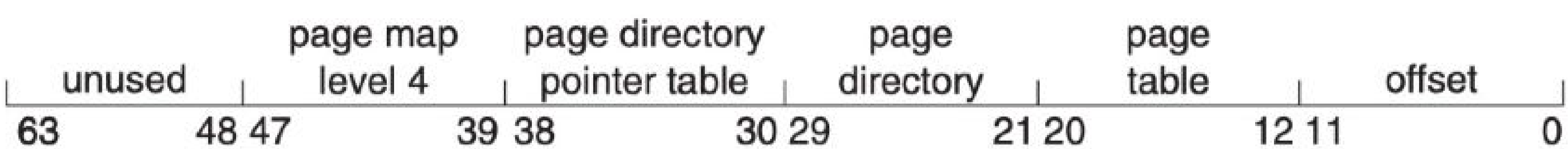
逻辑地址

多级页表



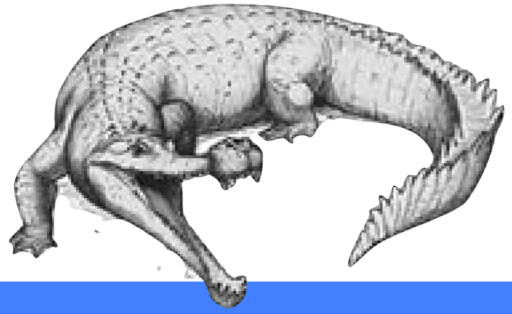
- 上面是一个32位地址采用两级页表的例子，页面大小是4KBytes，第一级页表页的数量是1K个，每个页表页中包含的页面数量也是1K个。

- 下面是x86-64架构CPU采用的四级页表方案:



下期预告

- 下次直播时间：3月25日 上午9:30
- 课程内容
  - Lecture 15 Virtual Memory
- Q&A



Lecture 14

The End