2020春

LINUX OPERATING SYSTEM

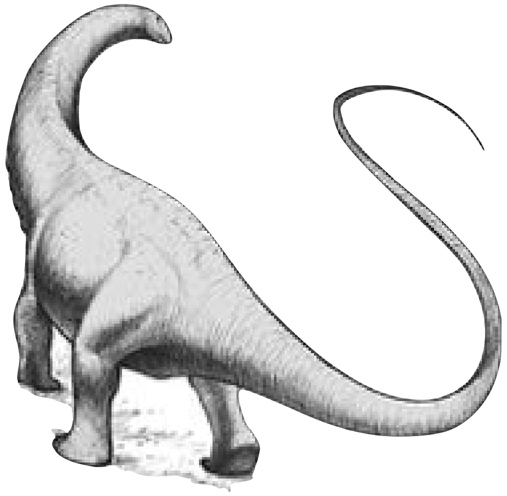
YANG

LINUX操作系统（双语）

双语课→课件内容中英混排

Lecture 13

Segmentation & Paging



本讲内容

- 动机
- 分段
- 分页

动机

MOTIVATION

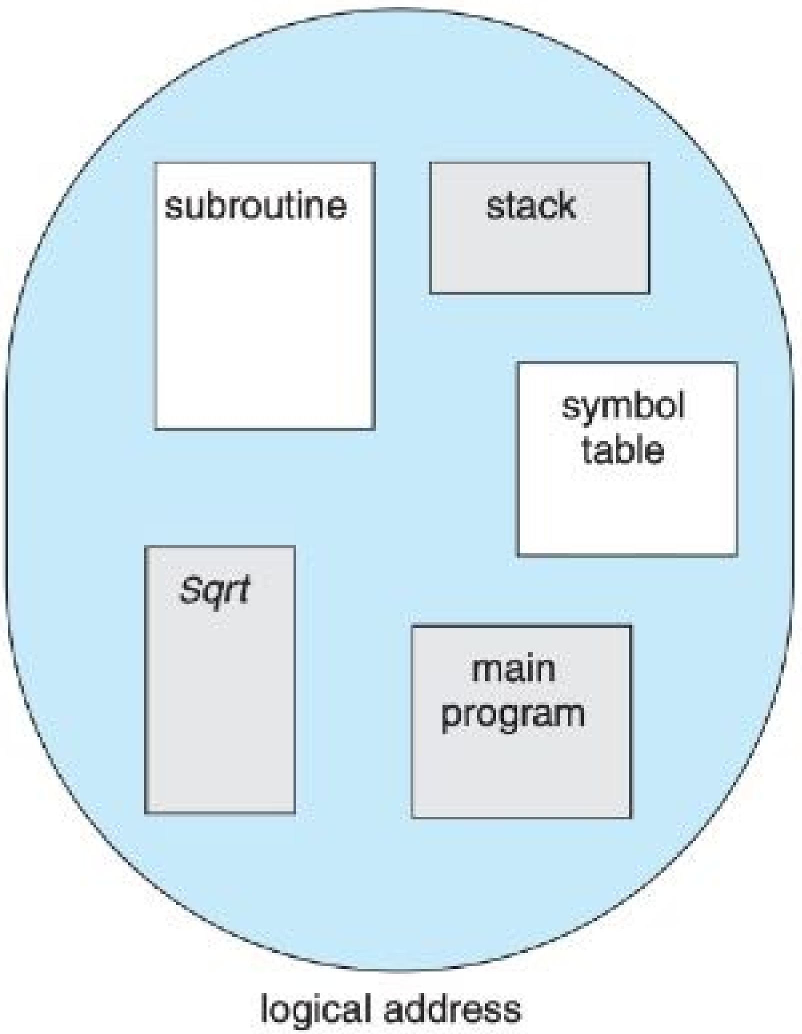
- Solution to fragmentation: permit the logical address space of processes to be noncontiguous.
- The view of memory is different between
  - logical (programmer's): a variable-sized segments
  - physical : a linear array of bytes
- The hardware could provide a memory mechanism that mapped the logical view to the actual physical memory.

分段

程序🦧眼中的内存世界

程序猿看到的:

- 主函数和一组其他函数
- 各种数据结构：变量、结构体、对象、数组等
- 所有的模块都是名字来引用的



- 因此他们认为在内存中，程序是由若干个大小不等的段构成的，每个段都有专门的用途，段的大小和用途相关。
- 段和段之间不必连续存放（离散）

划分段

```
0.void f(){
1.//......
2.//......
3.//......
4.}
5.void g(){
6.//......
7.//......
8.}
9.int main(){
10.//......
11.//......
12.return 0;
13.}
```

```
0.void f(){
1.//......
2.//......
3.//......
4.}
```

```
0.void g(){
1.//......
2.//......
3.}
```

```
0.int main(){
1.//......
2.//......
3.return 0;
4.}
```

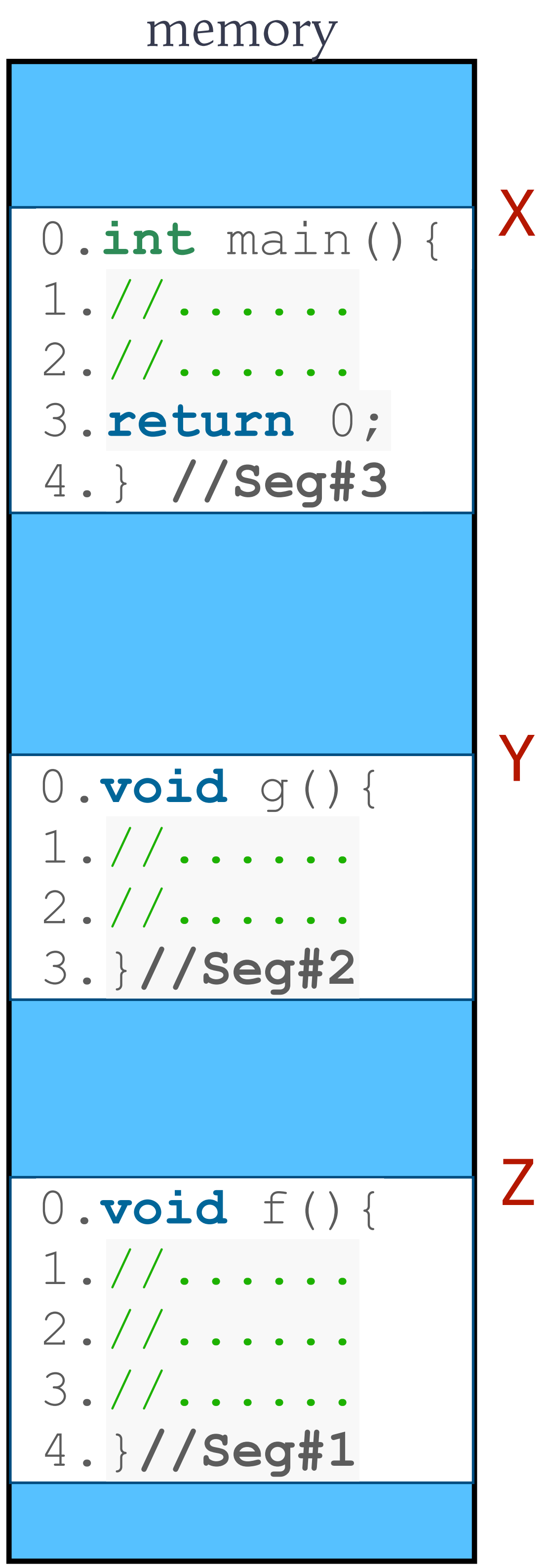
$\langle$segment-number, offset$\rangle$

分段硬件

- 段基址
- 段限长
- 段表

段号	段内位移

逻辑地址



16位段式地址转换实例

```
0x240    main:     la $a0, varx
0x244              jal strlen
  …                   …
0x360    strlen:   li   $v0, 0
0x364    loop:     lb   $t0, ($a0)
0x368              beq  $r0,$t1, done
  …                   …
0x4050   varx      dw   0x314159
```

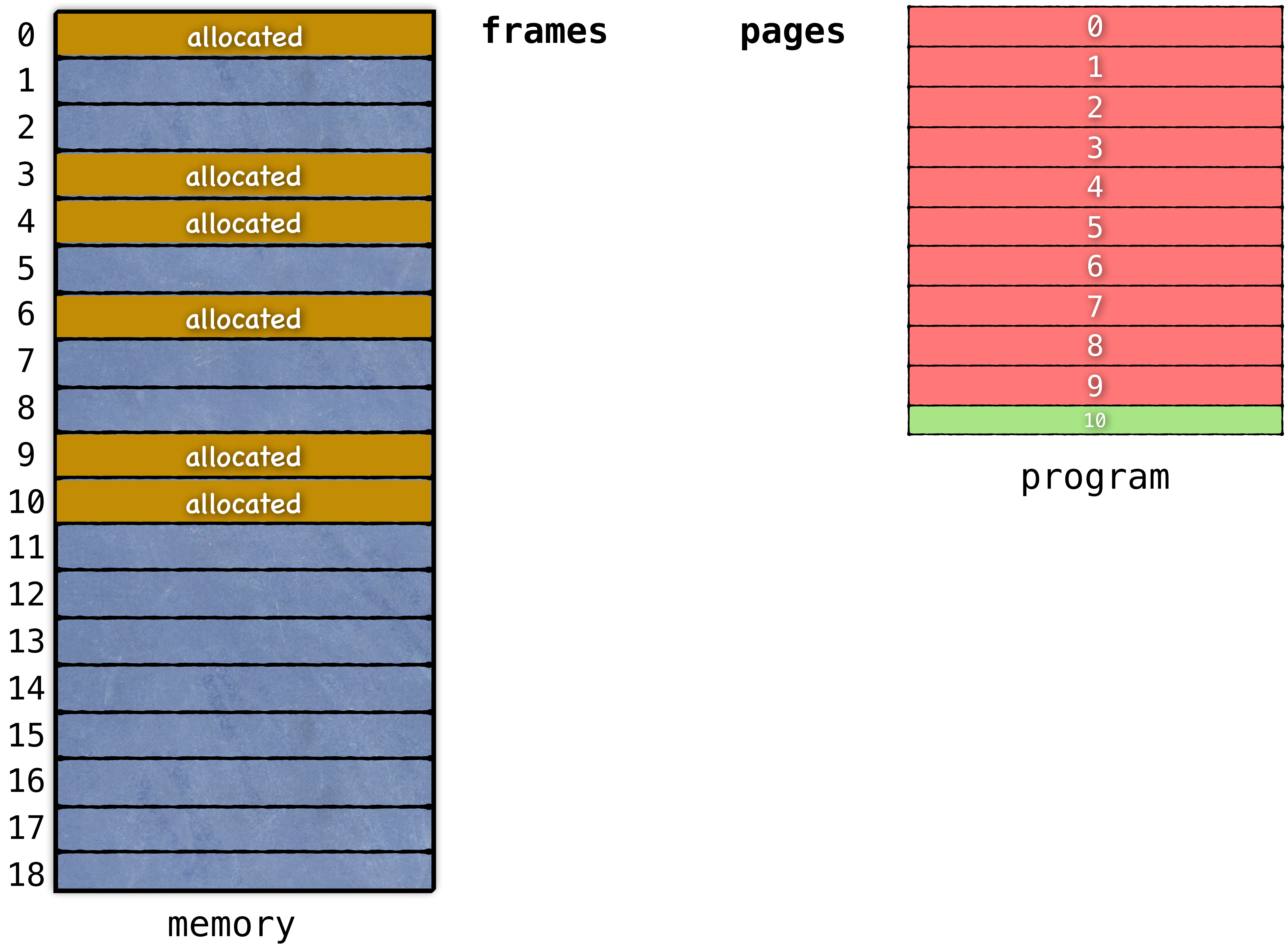
段号	基址	限长
0 (code)	0x4000	0x0800
1 (data)	0x4800	0x1400
2 (shared)	0xF000	0x1000
3 (stack)	0x0000	0x3000

- 假设逻辑地址的段号占2bits，段内位移占14bits，此时 PC寄存器的值为0x240
- 下一条指令的物理地址为：____________
- Move 0x4050 → $a0， Move PC+4 → PC，下条指令物理地址为：___________
- Move 0x0248 → $ra (return address!)， Move 0x0360 → PC，下条指令物理地址为：___________
- Move 0x0→$v0， Move PC+4→PC，下条指令物理地址为：____________
- "lb $t0,($a0)"(将a0寄存器中所示内存地址处取出1个字节存到寄存器t0中)，该内存地址为：___________， 即 Load Byte from 0x4850 → $t0, Move PC+4→PC ……

逻辑地址

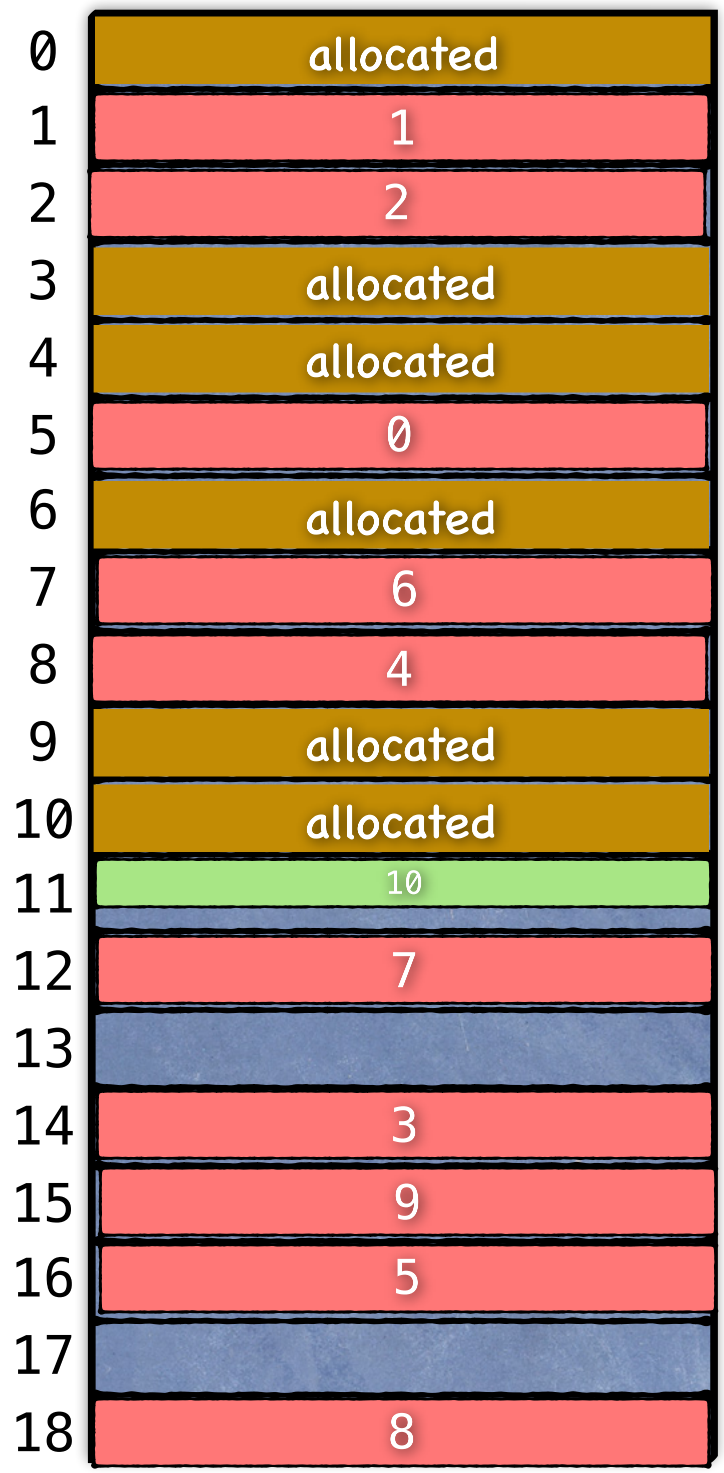
分页

基本方法



0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
allocated
allocated
allocated
allocated
allocated
allocated
memory
frames
pages
0
1
2
3
4
5
6
7
8
9
10
program

基本方法



memory

page no	frame no
0	5
1	1
2	2
3	14
4	8
5	16
6	7
7	12
8	18
9	15
10	11

页号与页内位移

计算物理地址

页号	页内位移

逻辑地址

请计算以下逻辑地址对应的物理地址（10进制）：

1. |0|3| →
2. |2|3| →
3. |1|3| →

计算物理地址

页号	页内位移

逻辑地址

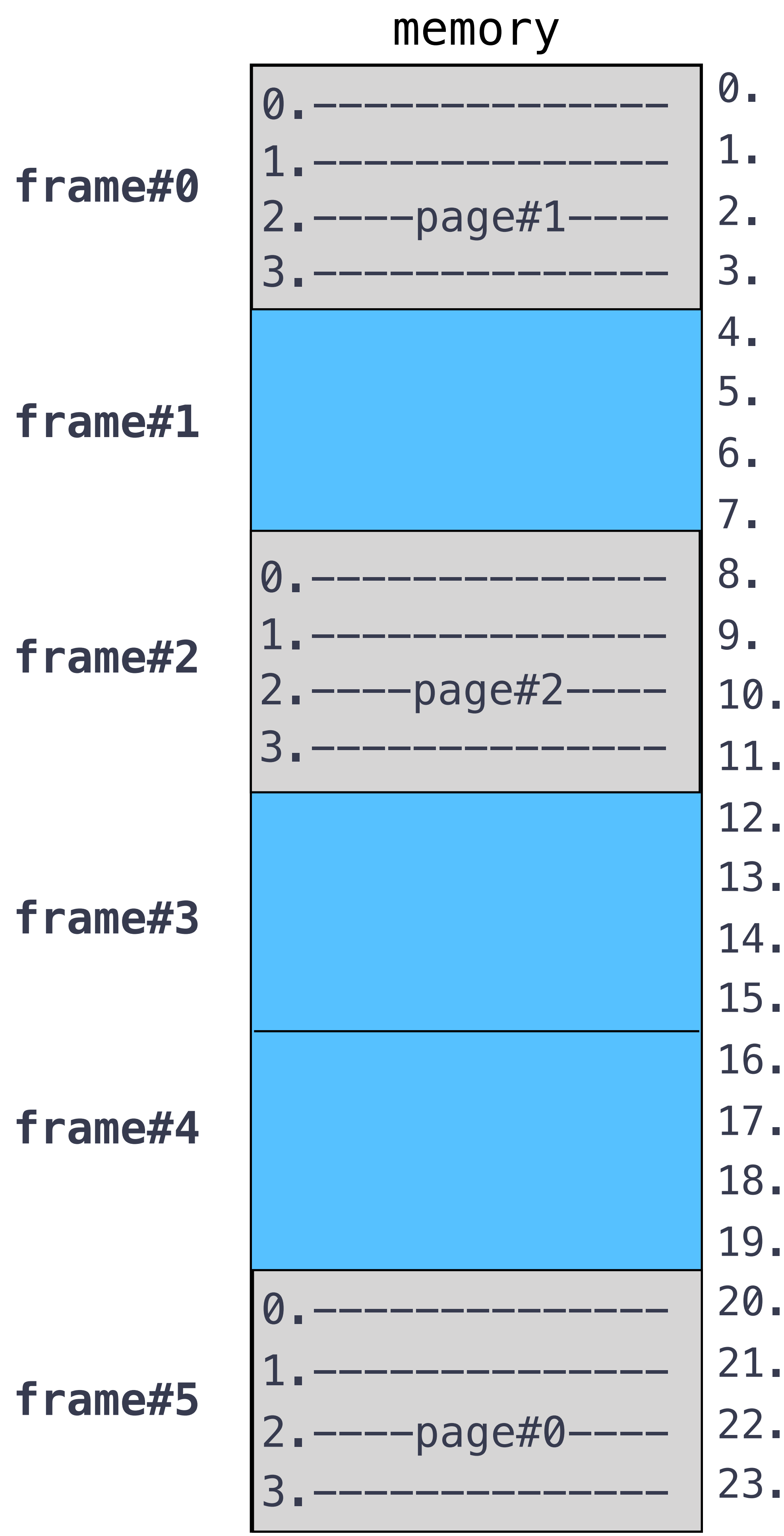
请计算以下逻辑地址对应的物理地址（10进制）：

1. |0|3| →

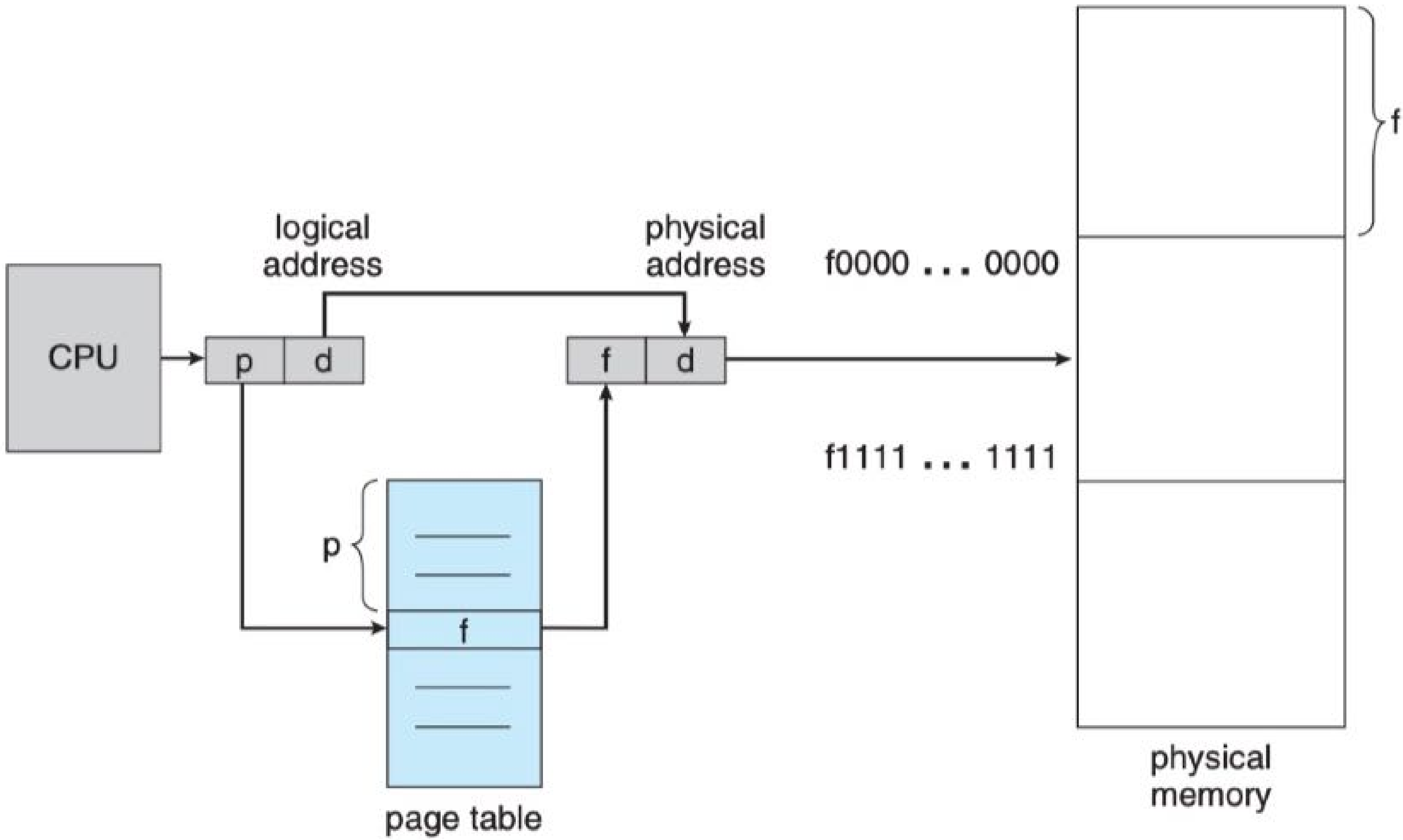
2. |2|3| →

3. |1|3| →

physical address = frame_no * pagesize + offset



分页硬件


CPU
logical address
p
d
physical address
f
d
f0000 ... 0000
f1111 ... 1111
f
p
f
page table
physical memory

LOGICAL ADDRESS

- The page size (like the frame size) is defined by the hardware. The size of a page is a power of 2, varying between 512 bytes and 1 GB per page, depending on the computer architecture.
- The selection of a power of 2 as a page size makes the translation of a logical address into a page number and page offset particularly easy.
- If the size of the logical address space is 2^m, and a page size is 2^n bytes, then the high-order m – n bits of a logical address designate the page number, and the n low-order bits designate the page offset. Thus, the logical address is as follows:

分段与分页的区别

分段	分页
信息的逻辑单位	信息的物理单位
段长是任意的	页长由系统确定
段的起始地址可以从主存任一地址开始	页框起始地址只能以页框大小的整数倍开始
(段号，段内位移)构成了二维地址空间	(页号，页内位移)构成了一维地址空间
会产生外部碎片	消除了外部碎片，但会出现内部碎片

下期预告

- 下次直播时间：3月20日 上午9:30
- 课程内容
  - Lecture 14 Page Table
- Q&A

Lecture 13

The End