CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

More Paging

More Paging

Physical memory and logical memory are split into frames and pages

Any page can be mapped to any frame

Mapping is transparent to process

To translate addresses, MMU and OS must work together (as has been the case to some degree for all of our schemes)

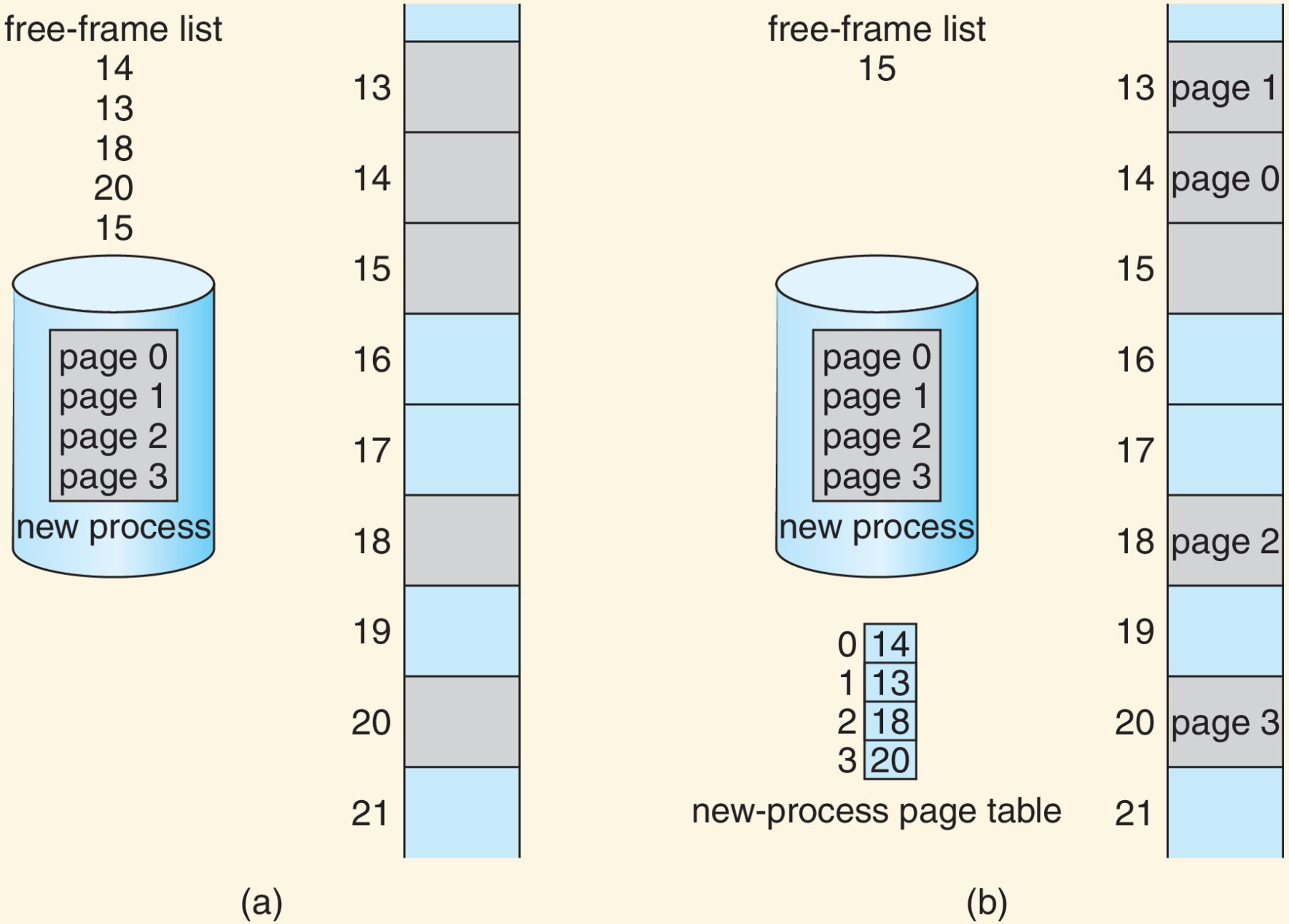
MMU handles actual translation, but OS is responsible for managing page table

There is a page table associated with each process

OS is responsible for filling page tables

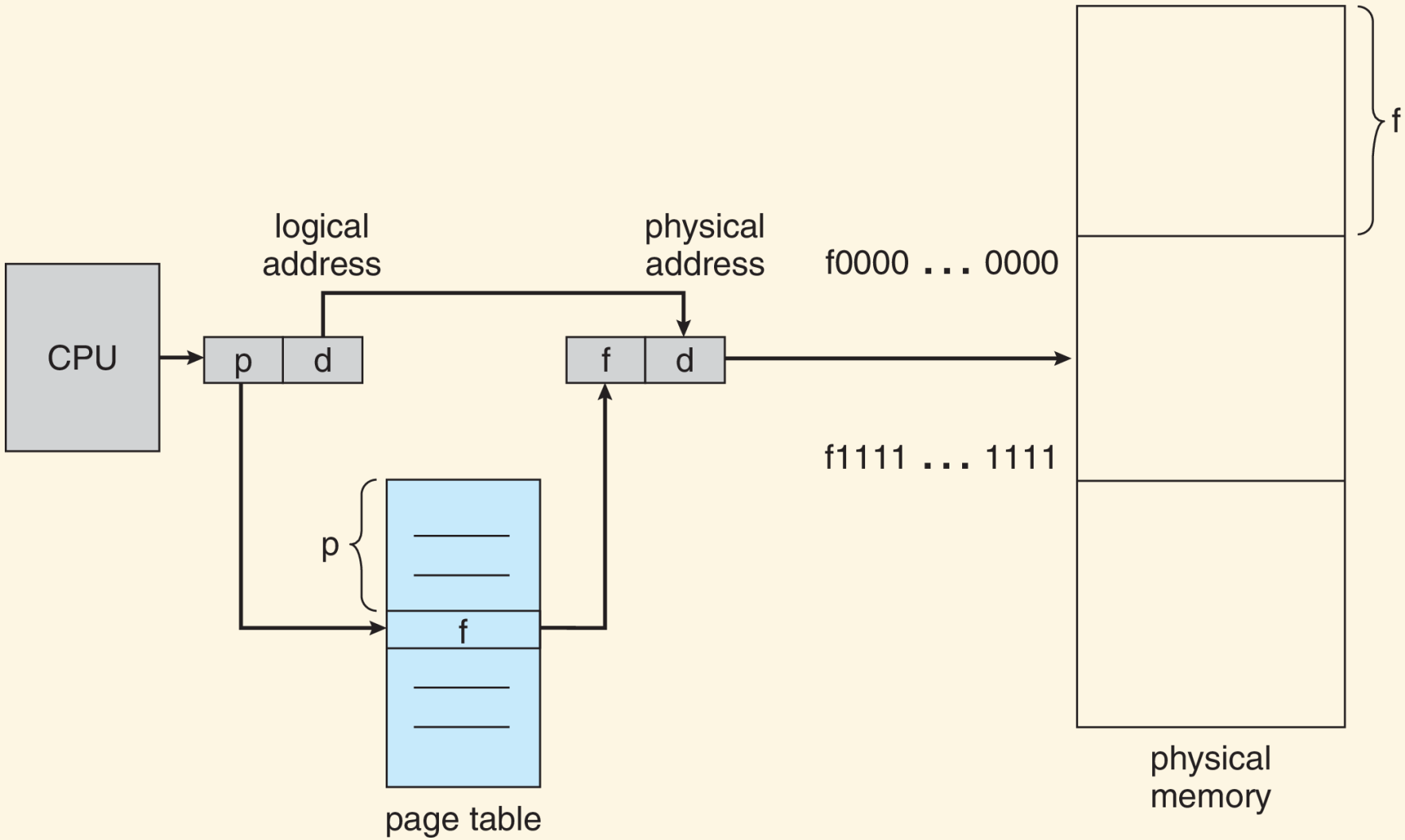
OS must know about available frames to do its job

Keeps frame information in **frame table**


free-frame list
14
13
18
20
15
page 0
page 1
page 2
page 3
new process
13
14
15
16
17
18
19
20
21
(a)
free-frame list
15
page 0
page 1
page 2
page 3
new process
0 14
1 13
2 18
3 20
new-process page table
13 page 1
14 page 0
15
16
17
18 page 2
19
20 page 3
21
(b)

Memory access is protected by design

Physical address determined by page table, which is controlled by OS


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

Paging can also make implementing shared memory simple

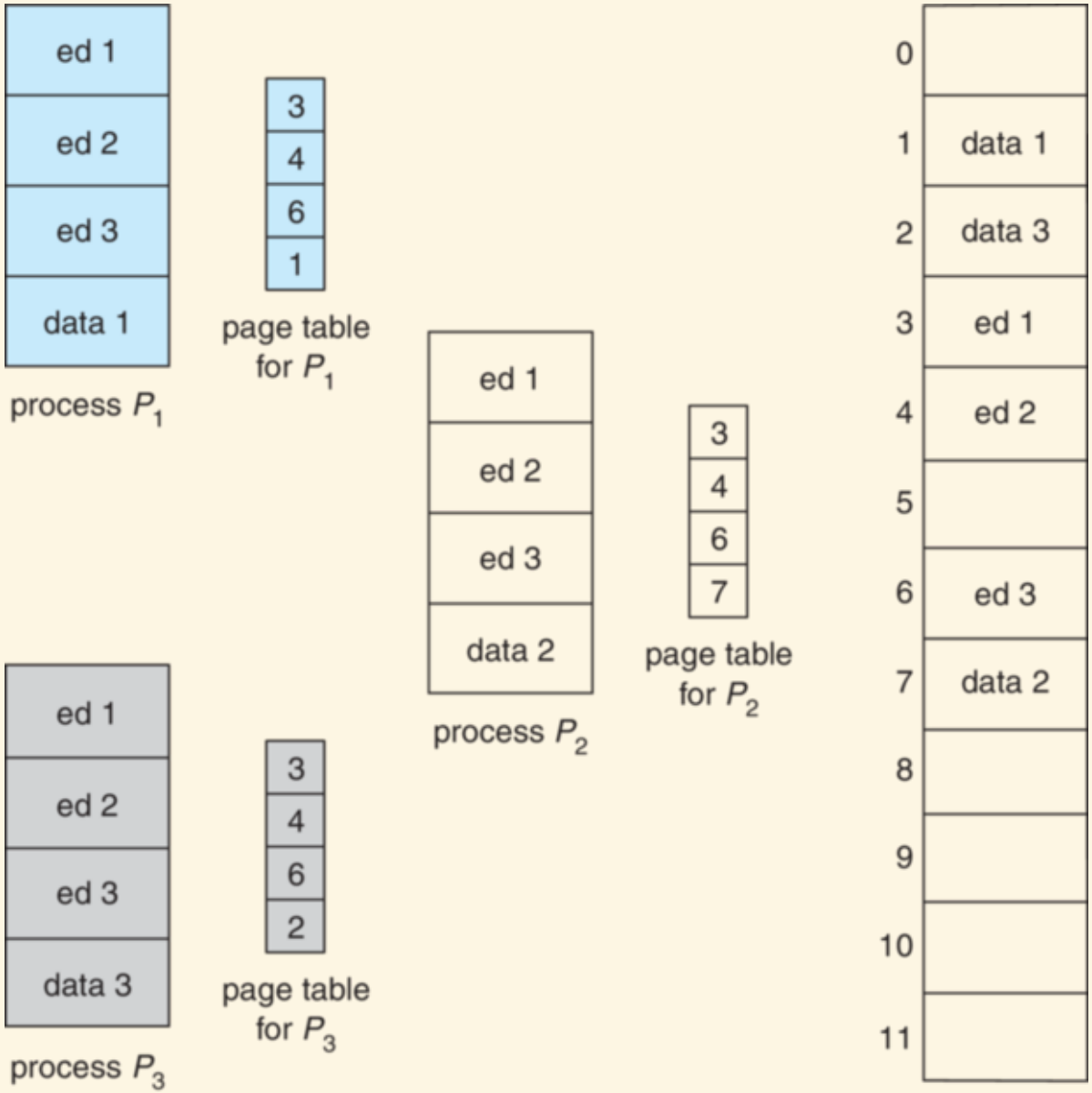
If OS puts same frame into page table of two (or more) processes, then they will both be allowed to access same memory

Note that it is a shared *frame* (physical memory), not a shared *page* (logical memory) that matters here

Shared pages can also be useful outside of IPC

Imagine 40 users all want to use same editor, and editor requiers 150 KB of code and 50 KB of data in memory -- total memory demand is 8,000 KB

Since code is read-only, it can be shared and only one copy need be kept in memory


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

Total memory for previous example is now 40*50 KB + 150 KB = 2,150 KB

Contrasted with previous 8,000 KB, this is a large savings

Benefits:

- No external fragmentation
- Maps well to backing store
- Simple protection
- Easy sharing of memory

Backing store will have similar issues as memory

Backing now store split into blocks of the same size as frames/pages, making management of backing store itself simpler

One downside is internal fragmentation

On average, each process will have (page_size/2) bytes of internal fragmentation

In terms of fragmentation, smaller pages are better

However, additional pages require additional storage overhead

Also, disks are generally more efficient when transferring large amounts of data

Page sizes in examples were obviously unreasonable

Typical memory for personal computers is several gigabytes, and page sizes are several kilobytes to megabytes (or sometimes gigabytes)

```
$ getconf PAGESIZE
4096
```

Most OSes use paging