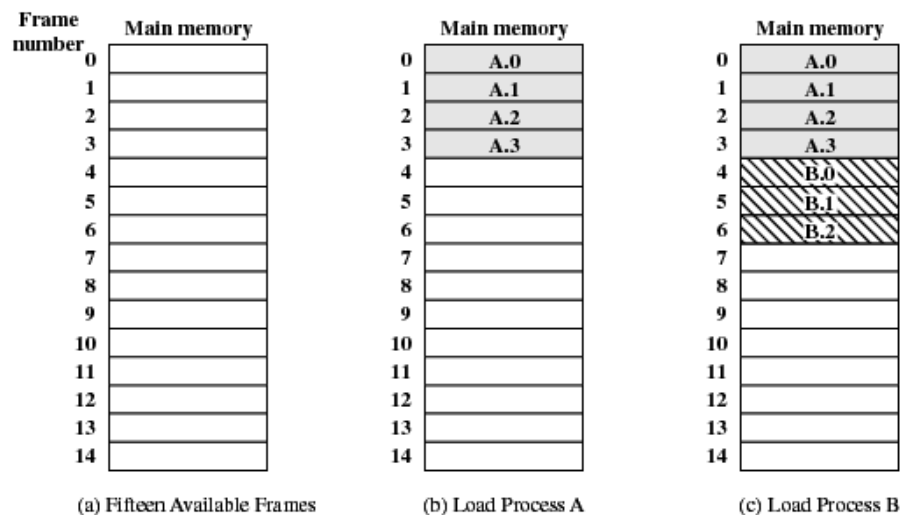# Paging

- Partition memory into small equal fixed-size chunks called page **frames**.
- Processes divided into **pages** as well
- Page frames and pages are of equal size
  - try "pagesize" command
- Operating system maintains a page table for each process
  - Contains the frame location for each page in the process
  - Memory address consist of a page number and offset within the page

1

Assignment of Process Pages to Free Frames

Frame number	Main memory		Main memory		Main memory
0		0	A.0	0	A.0
1		1	A.1	1	A.1
2		2	A.2	2	A.2
3		3	A.3	3	A.3
4		4		4	B.0
5		5		5	B.1
6		6		6	B.2
7		7		7	
8		8		8	
9		9		9	
10		10		10	
11		11		11	
12		12		12	
13		13		13	
14		14		14	
	(a) Fifteen Available Frames		(b) Load Process A		(c) Load Process B

2

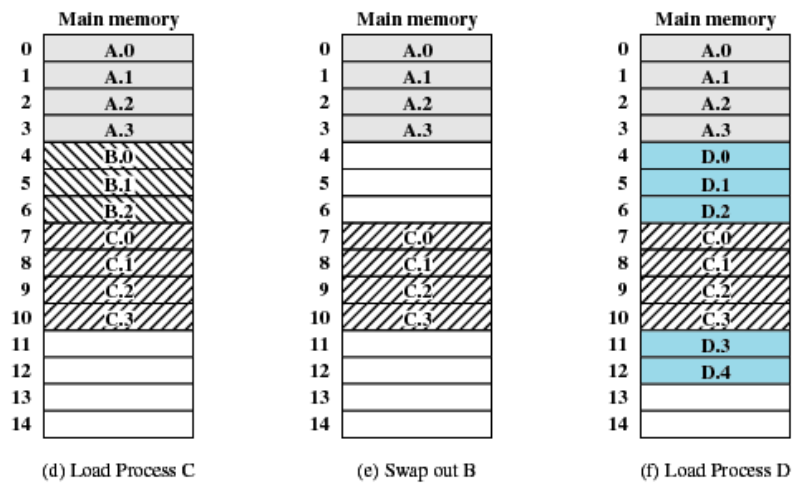# Assignment of Process Pages to Free Frames

Figure 7.9 Assignment of Process Pages to Free Frames
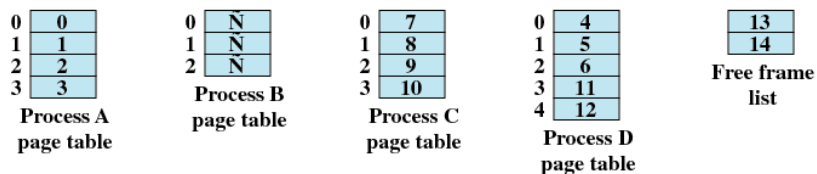
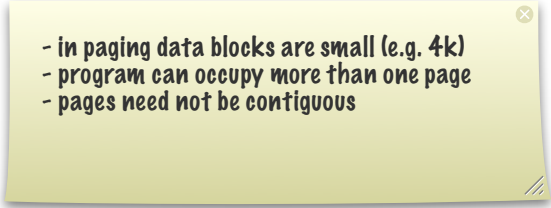# Page Tables for Example

Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Paging

- The page frames are of equal size.

Is this the same as fixed partitioning?

- in paging data blocks are small (e.g. 4k)
- program can occupy more than one page
- pages need not be contiguous

Segmentation

- All segments of all programs do not have to be of the same length
- There is a maximum segment length
- Addressing consist of two parts
  - a segment number and
  - an offset

Segmentation

- Since segments are not equal, segmentation may look a bit like dynamic partitioning...

    So is it the same or is something different?

    - program may occupy more than one segment
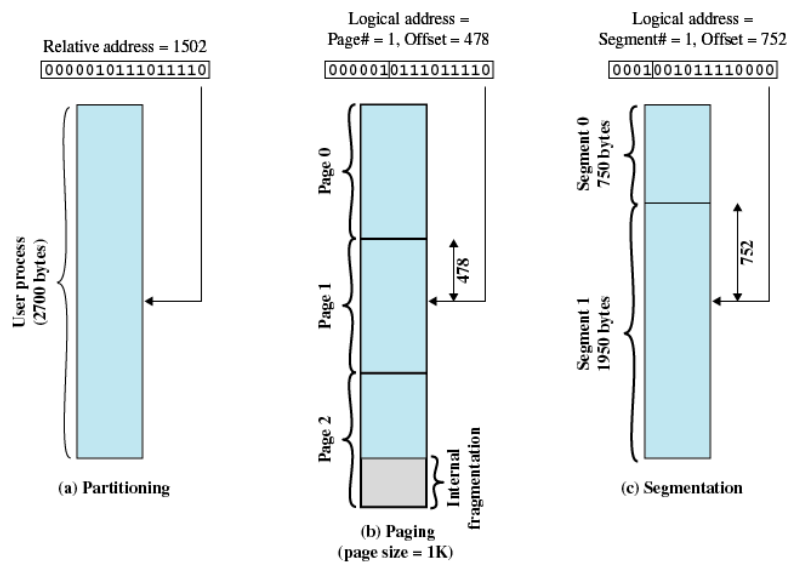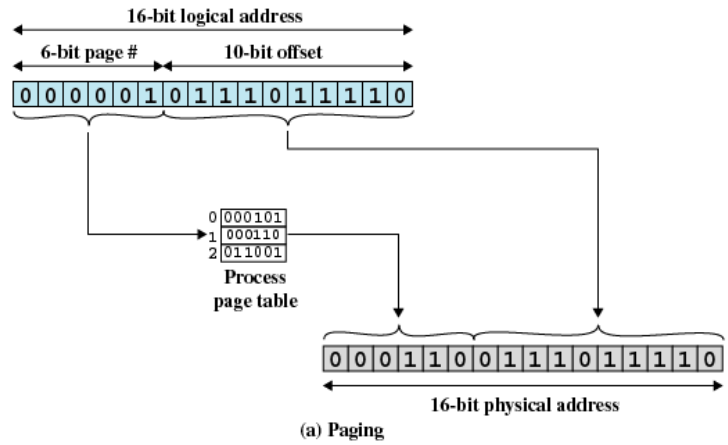   - segments need not be contiguous

---

Figure 7.11   Logical Addresses
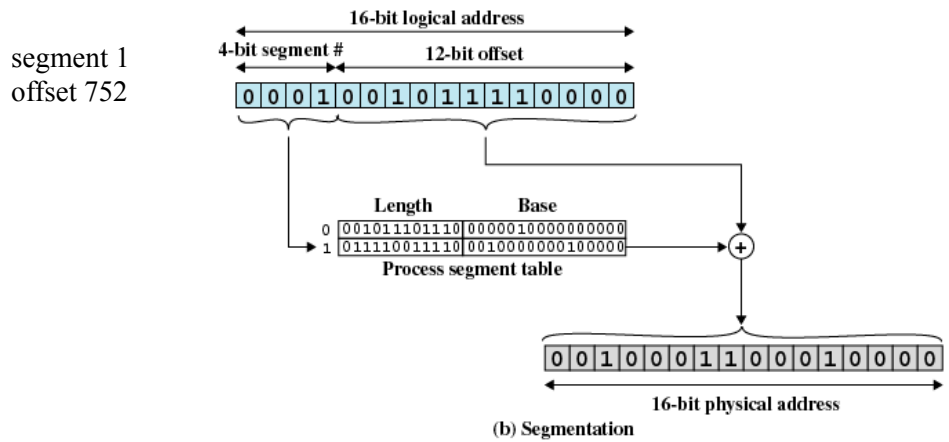
Logical-to-Physical Address Translation

16-bit logical address

6-bit page # | **10-bit offset**

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

Process page table
0 | 000101
1 | 000110
2 | 011001

| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

16-bit physical address

(a) Paging

segment 1
offset 752

16-bit logical address

4-bit segment # | **12-bit offset**

| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Length | Base
0 | 001011101110 | 0000010000000000
1 | 011110011110 | 0010000000100000

Process segment table

+

| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

16-bit physical address

(b) Segmentation

Figure 7.12 Examples of Logical-to-Physical Address Translation

Thus physical address is computed as:

```
    0010000000100000
+        001011110000
=   0010001100010000
```