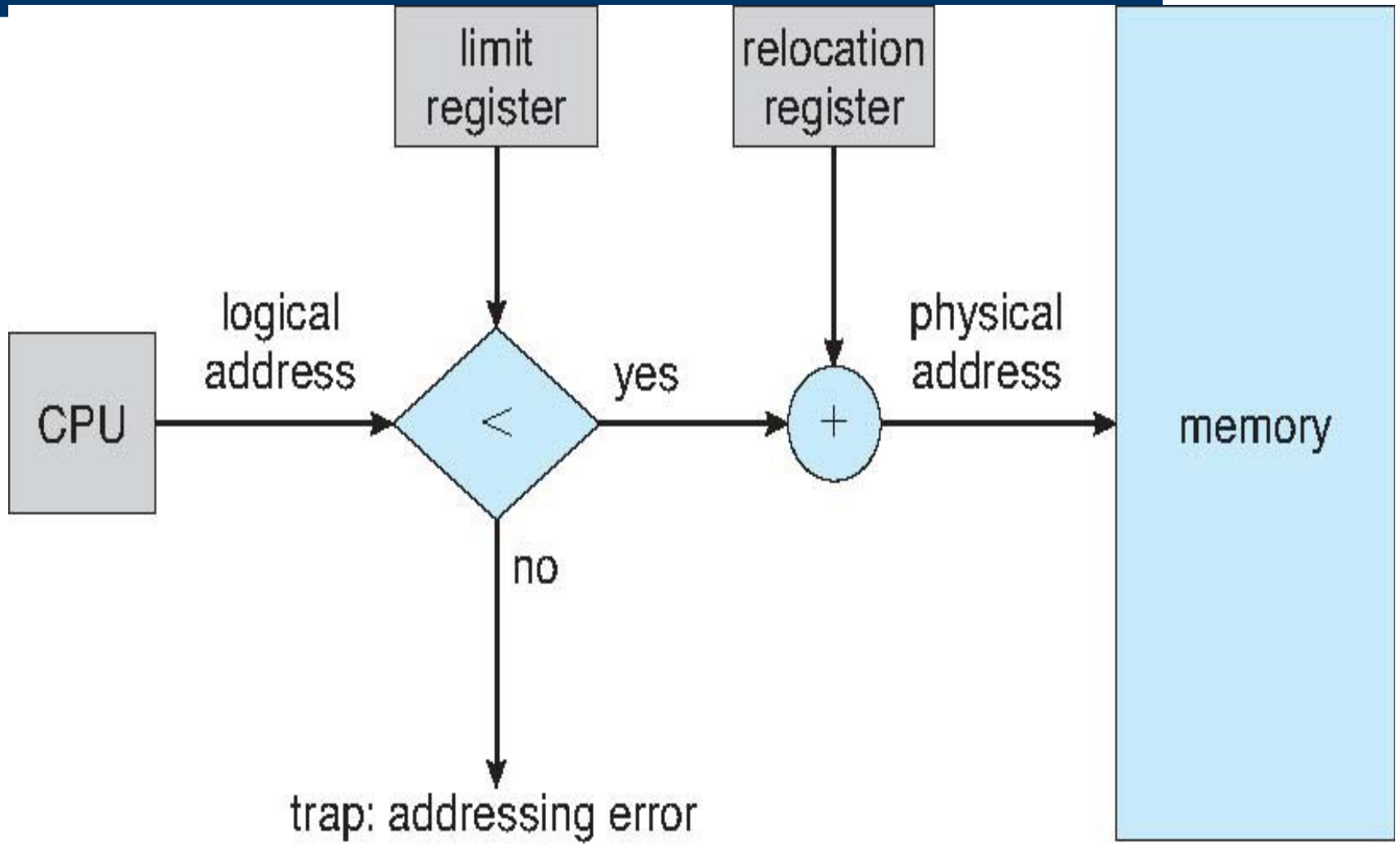
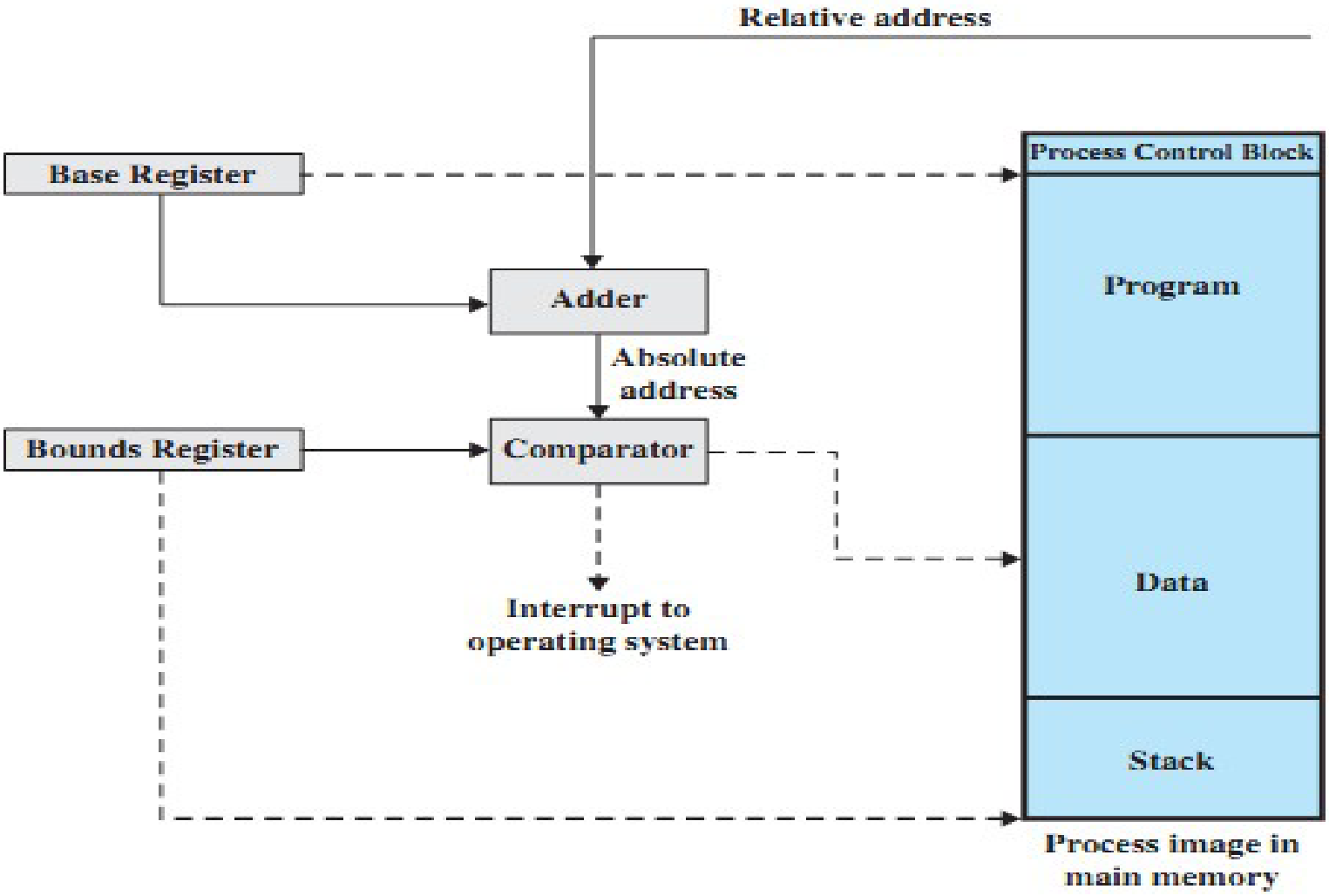
Hardware Support for Relocation and Limit Registers

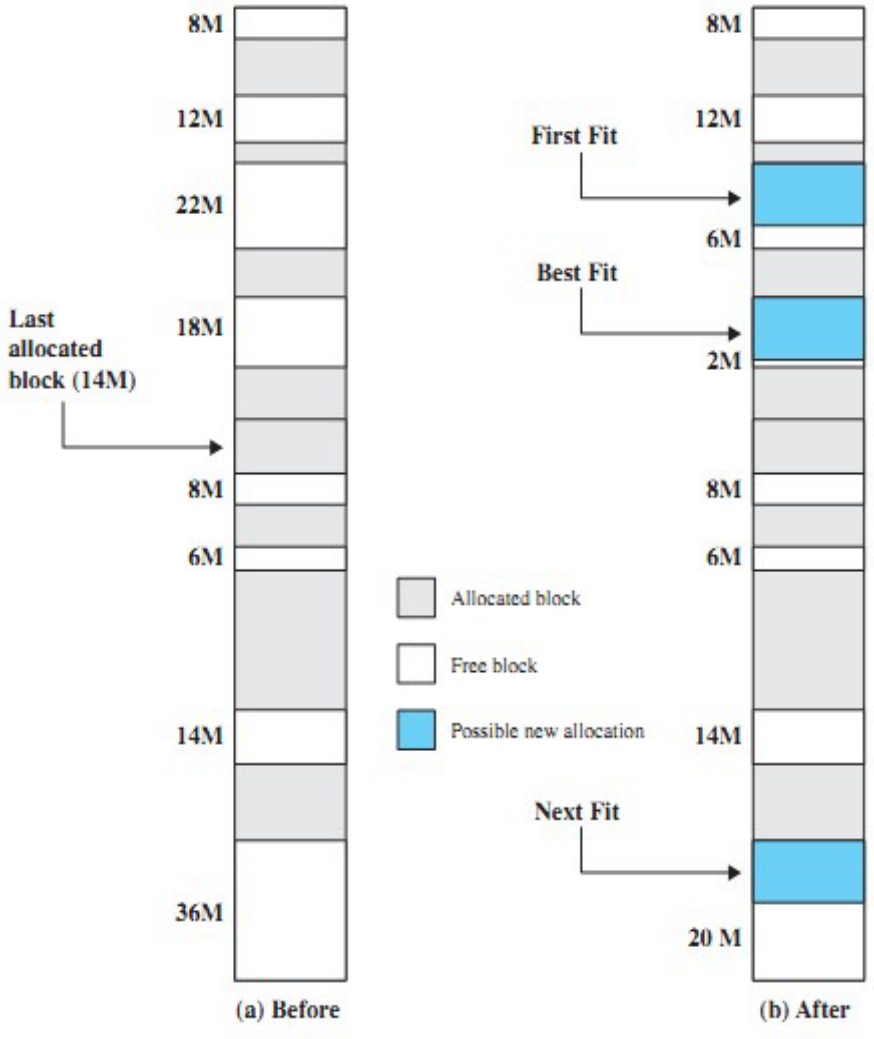


Example Hardware for Address Translation



Placement Algorithms

- Used to decide which free block to allocate to a process of 16MB.
- Goal: reduce usage of compaction procedure (its time consuming).
- Example algorithms:
  - First-fit
  - Next-fit
  - Best-fit
  - Worst-fit (to imagine)



Practice Exercise

Variable partitioning memory management is used by a system. At a given time there are the following holes in memory (in memory order): 10K, 4K, 20K, 18K, 7K, 9K, 12K, and 15K. Which hole is taken for successive process requests of 12K, 10K, and 9K using first fit, best fit, next fit, worst fit, best available fit strategy? Compute internal and external fragementation after applying each of the strategies on above process requests.

What do you think is the best strategy and why?

Practice Exercise 5

Consider a system with a 16KB memory. The sequence of processes loaded in and leaving the memory are given in the following.

P1	7K	loaded
P2	4K	loaded
P1		terminated and returned the memory space
P3	3K	loaded
P4	6K	loaded

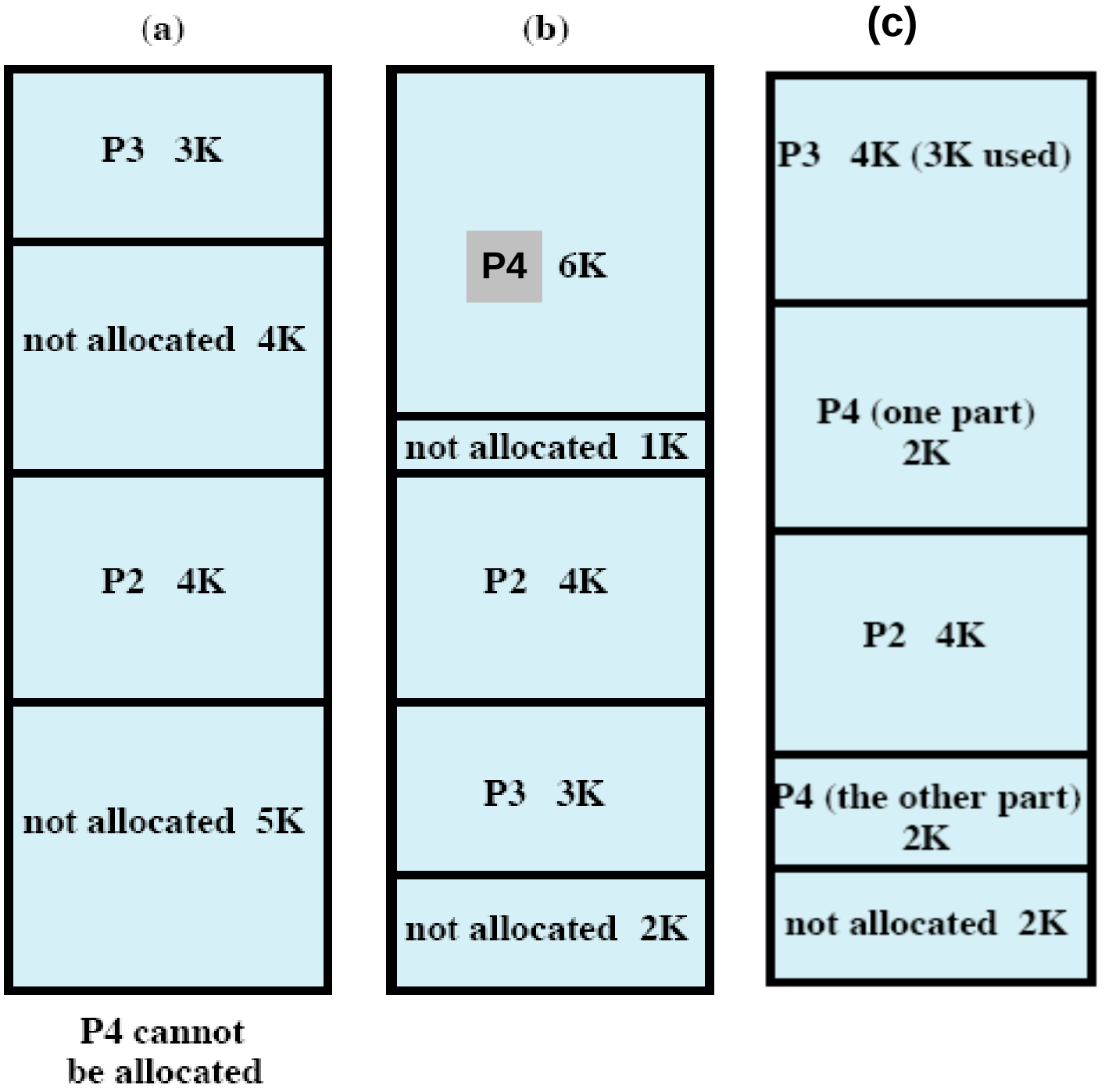
Assume that when a process is loaded to a selected "hole", it always starts from the smallest address. E.g. P1 will be loaded in memory locations 0 to 8K – 1 since the entire memory is one hole. Give the memory map showing allocated portion and free portion after the end of the sequence (if a process cannot be loaded, indicate that) for the following placement algorithms. Also, indicate the internal/external fragmentations.

a. first fit

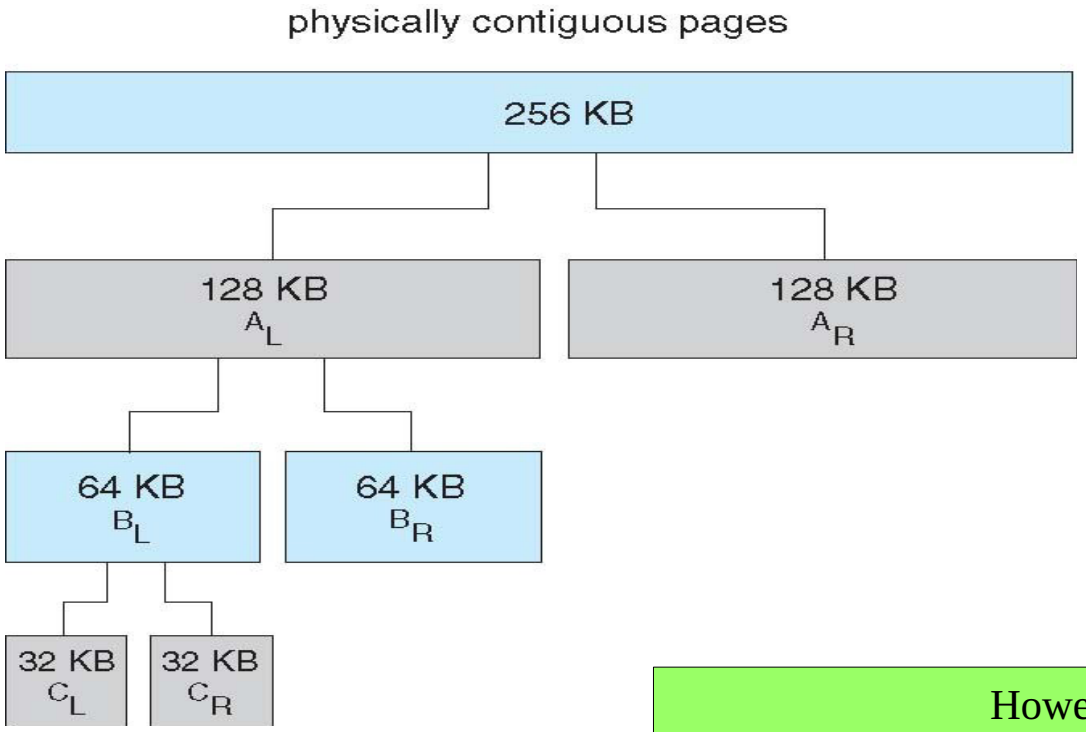
b. best fit

c. Simple paging (assume page size of 2K)

Solution 5


(a)
P3 3K
not allocated 4K
P2 4K
not allocated 5K
P4 cannot be allocated
(b)
P4 6K
not allocated 1K
P2 4K
P3 3K
not allocated 2K
(c)
P3 4K (3K used)
P4 (one part) 2K
P2 4K
P4 (the other part) 2K
not allocated 2K

Buddy System Allocation



Order 3 Buddy System
Order 0 – 256 KB (total memory – Given)
Order 1 – 128 KB
Order 2 – 64 KB
Order 3 – 32 KB

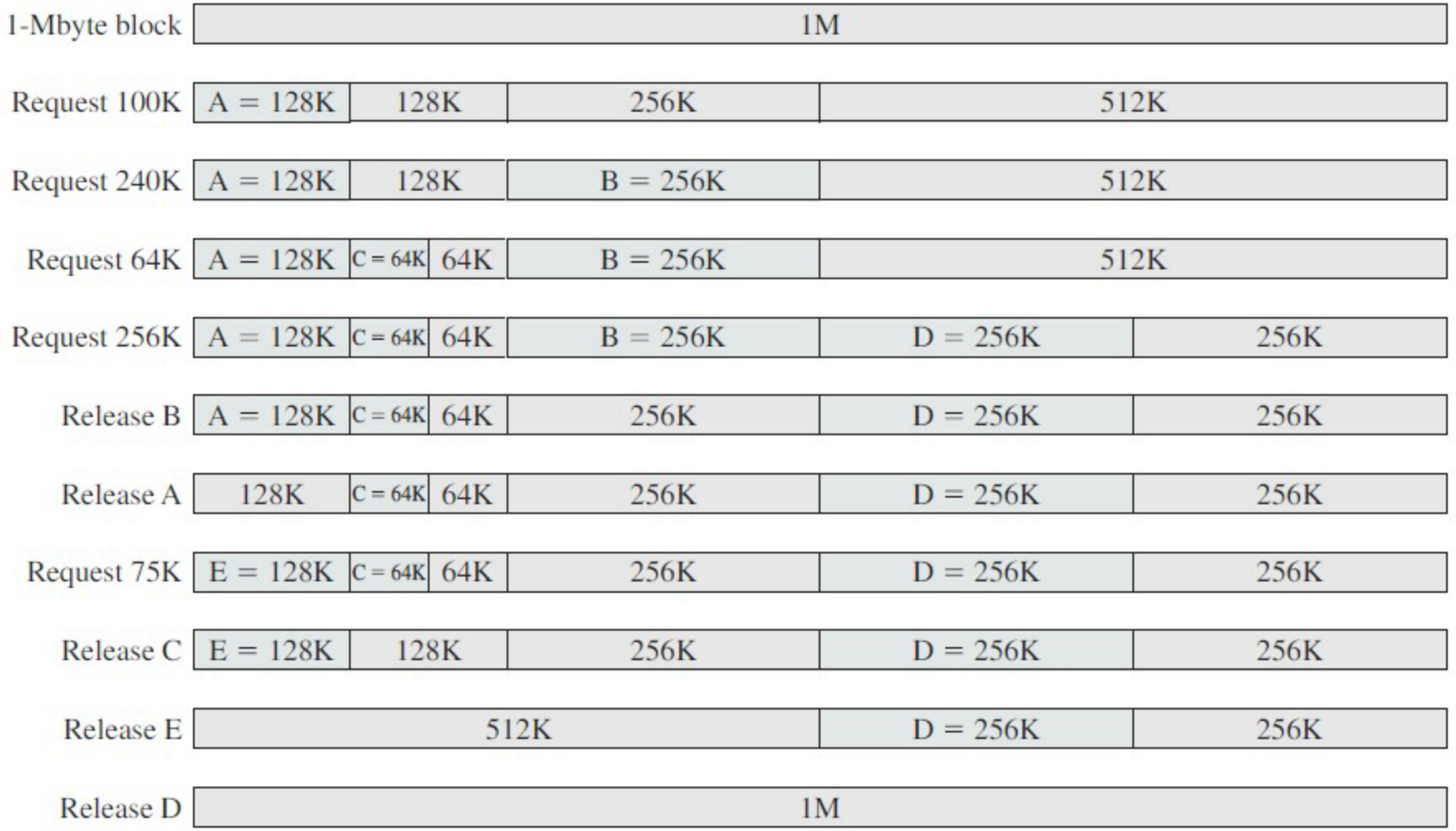
However, if it is Order 5 Buddy System
Order 0 – 256 KB (total memory – Given – largest buddy)
Order 1 – 128 KB
Order 2 – 64 KB
Order 3 – 32 KB
Order 4 – 16 KB
Order 5 – 8 KB (smallest buddy)

Practice Exercise

- Memory 1MB, Order (4)
- Requests in order – 100K(A), 240K(B), 64K(C), 256K(D)
- Release order – B, A
- New Request – 75K (E)
- Release order – C, E, D.
- Use Buddy system to showcase the above scenario. Give memory map and its tree representation.

Order(n) indicates that maximum n number of division operations permitted on a hole

Buddy System



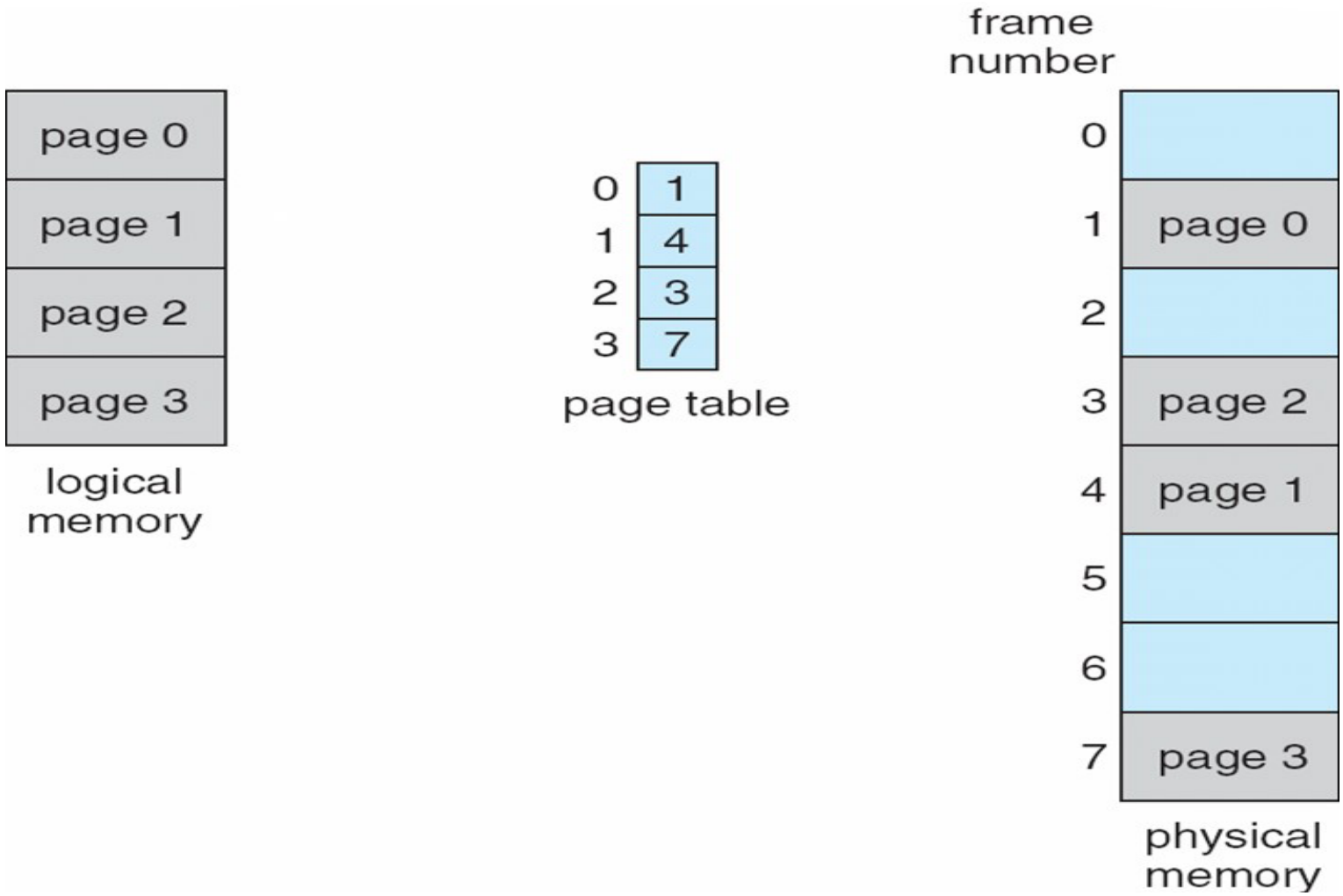
Practice Exercise

- Order 0 – 64K (indicates lower limit)
- Upper limit on order 4
- Memory requests (in order)
  - A (34K), B(66K), C(35K), D(67K)
- Release requests (in order)
  - B, D, A, C
- Show the process of allocation/deallocation using a buddy system approach for above memory requests considering system memory to be 1MB.

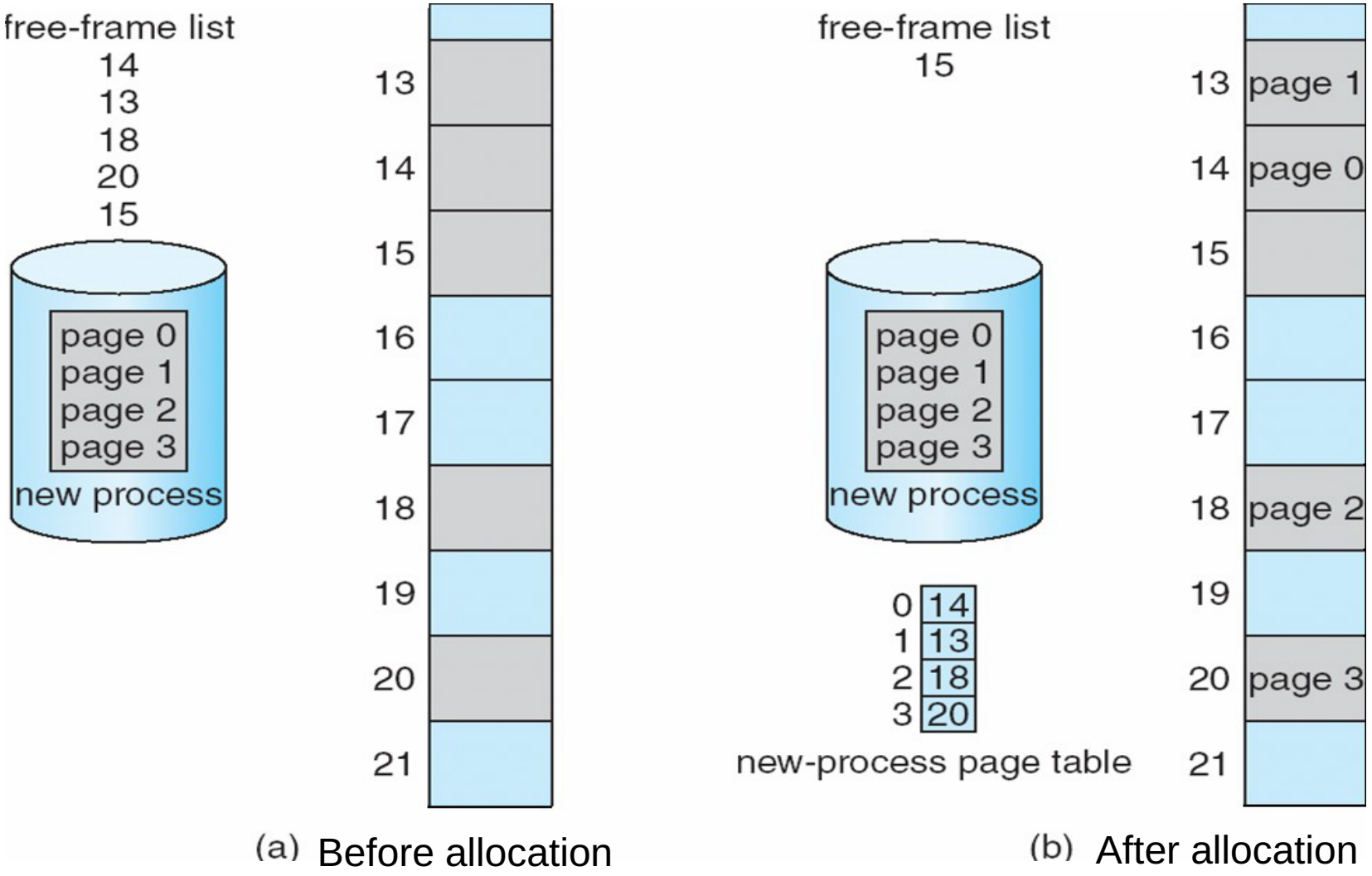
Solution

Step	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K
1	2(4)															
2.1	2(3)								2(3)							
2.2	2(2)				22				2(3)							
2.3	2(1)		2(1)		2(2)				2(3)							
2.4	2(0)	2(0)	2(1)		2(2)				2(3)							
2.5	A: 2(0)	2(0)	2(1)		2(2)				2(3)							
3	A: 2(0)	2(0)	B: 2(1)		2(2)				2(3)							
4	A: 2(0)	C: 2(0)	B: 2(1)		2(2)				2(3)							
5.1	A: 2(0)	C: 2(0)	B: 2(1)		2(1)		2(1)		2(3)							
5.2	A: 2(0)	C: 2(0)	B: 2(1)		D: 2(1)		2(1)		2(3)							
6	A: 2(0)	C: 2(0)	2(1)		D: 2(1)		2(1)		2(3)							
7.1	A: 2(0)	C: 2(0)	2(1)		2(1)		2(1)		2(3)							
7.2	A: 2(0)	C: 2(0)	2(1)		2(2)				2(3)							
8	2(0)	C: 2(0)	2(1)		2(2)				2(3)							
9.1	2(0)	2(0)	2(1)		2(2)				2(3)							
9.2	2(1)		2(1)		2(2)				2(3)							
9.3	2(2)				2(2)				2(3)							
9.4	2(3)								2(3)							
9.5	2(4)															

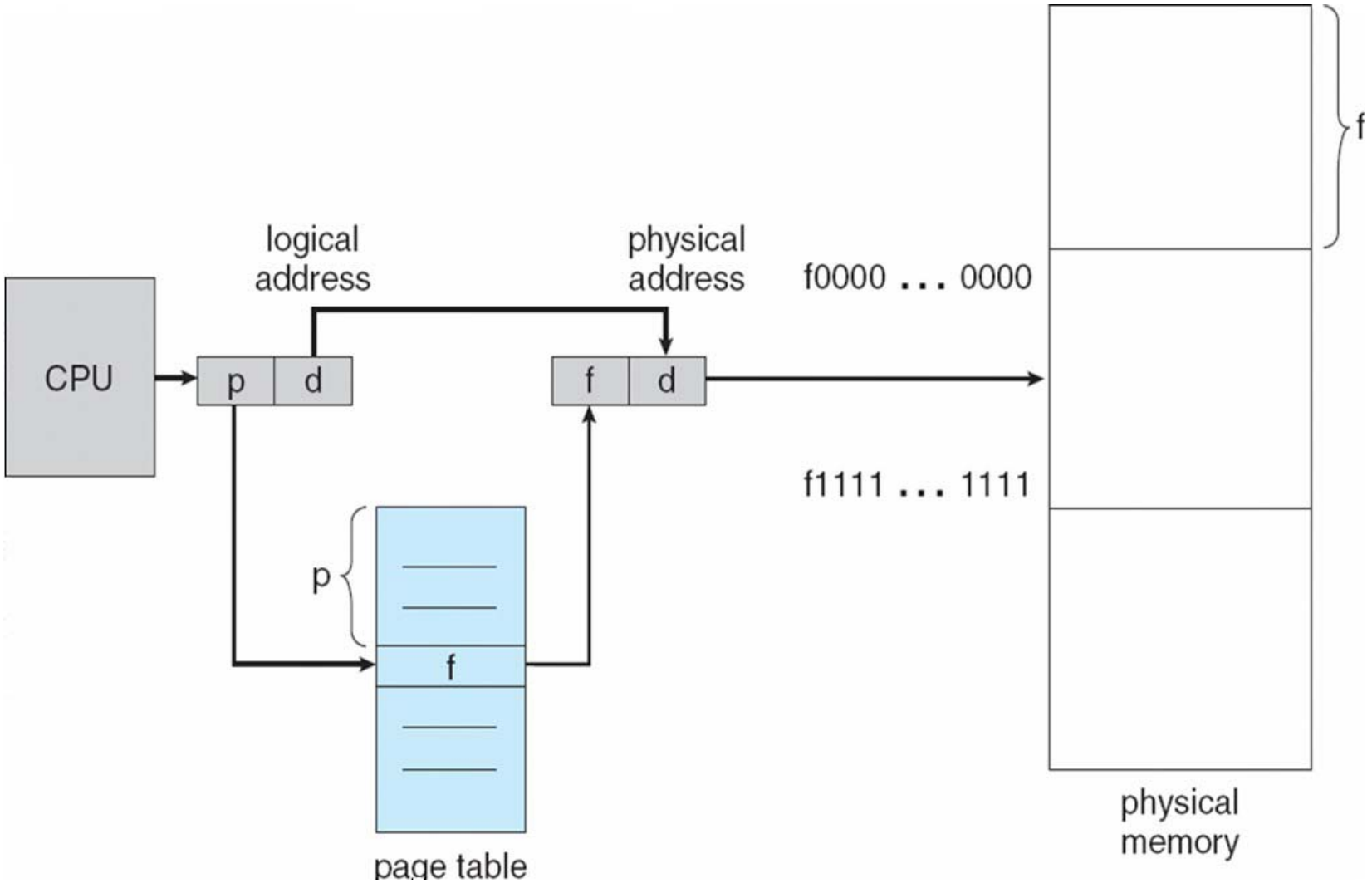
Paging Example



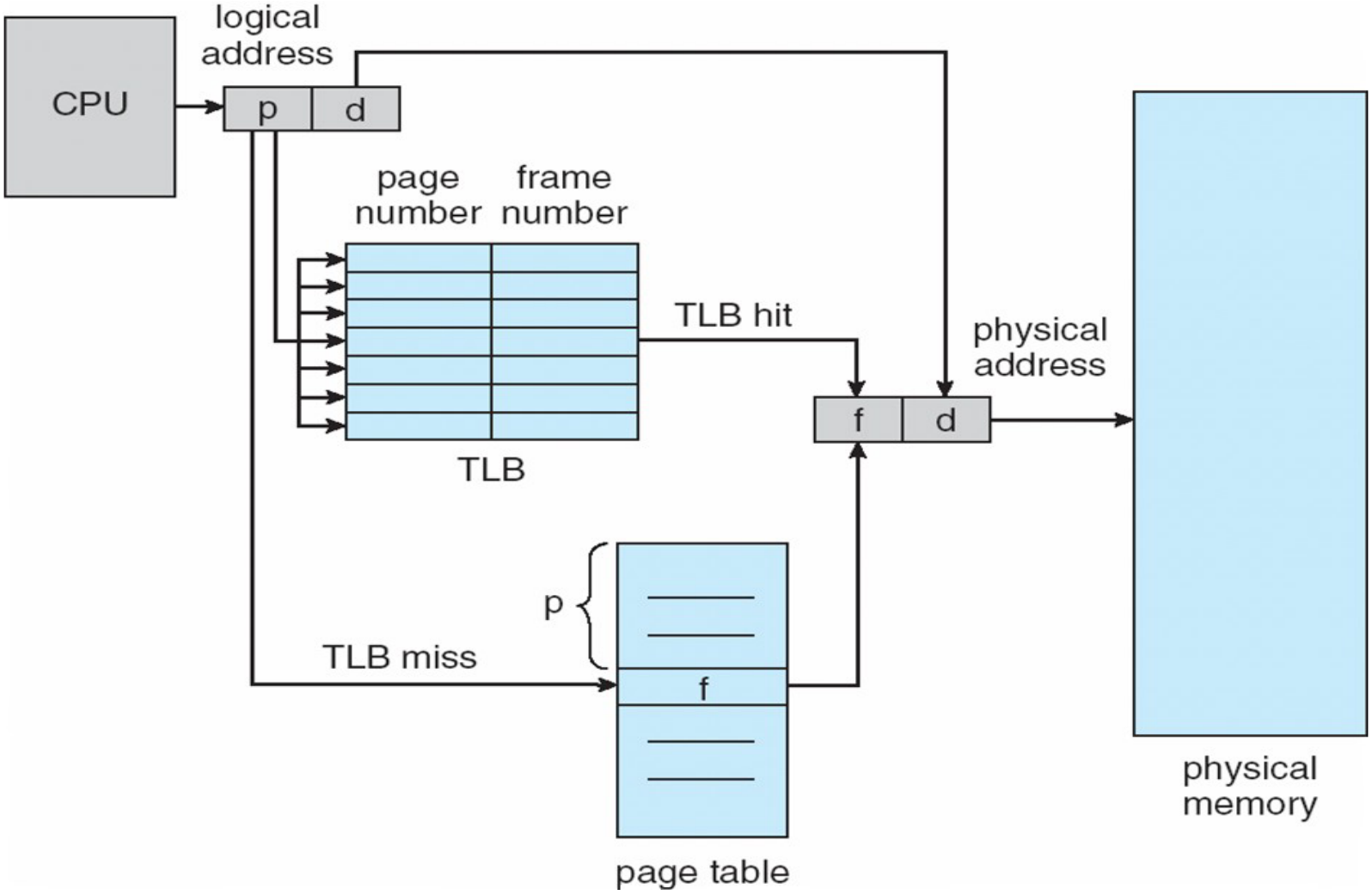
Free-Frame list example



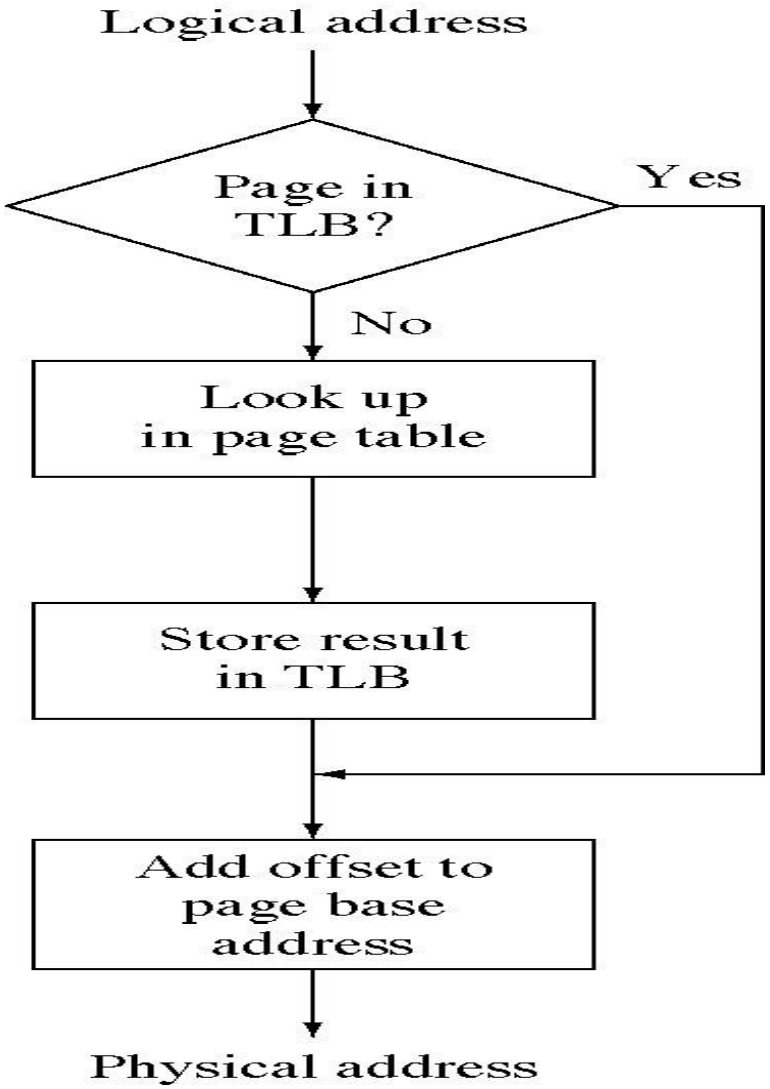
Address Translation Architecture



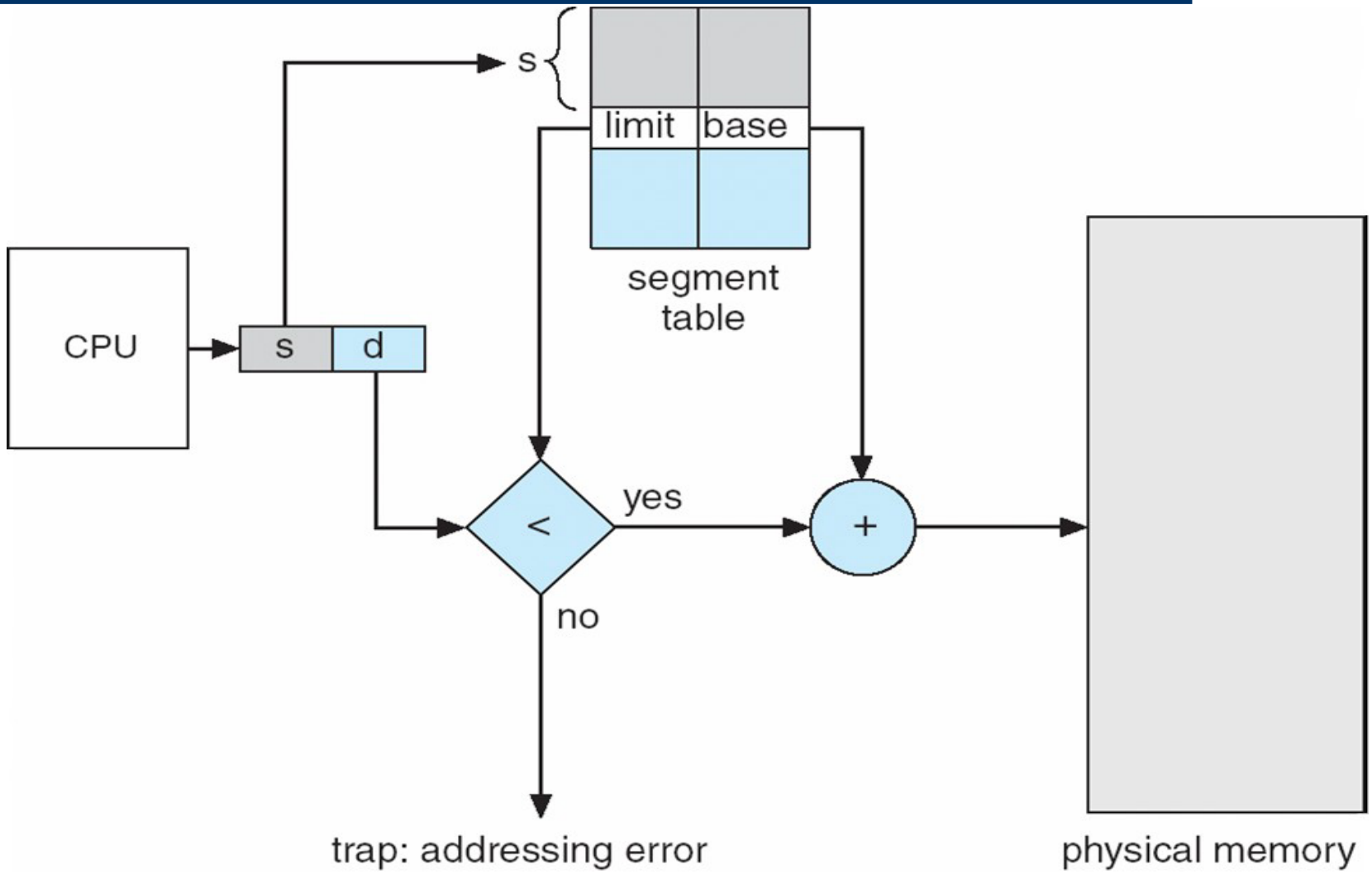
Paging Hardware With TLB



TLB Flow Chart

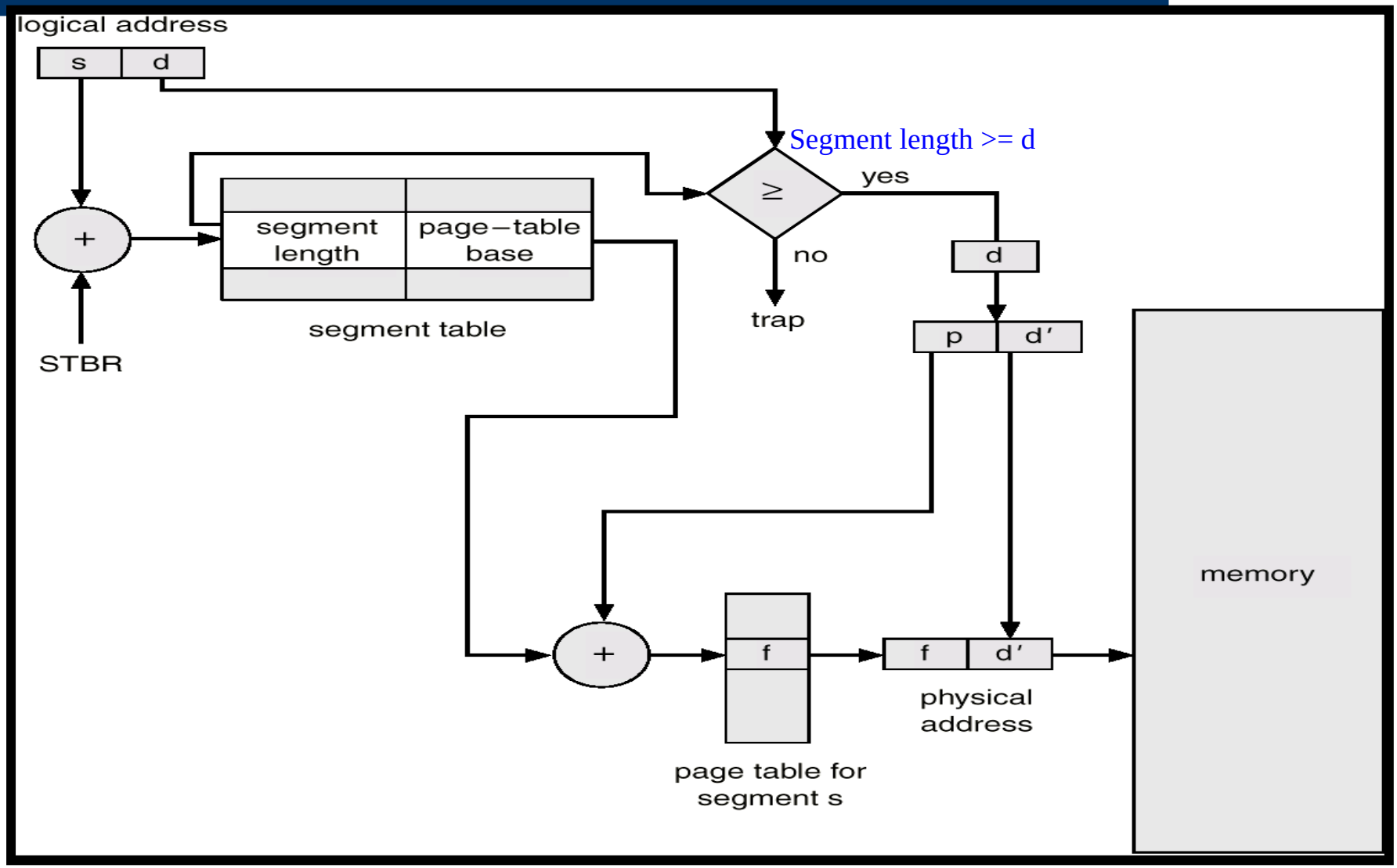


Address Translation Architecture



Case Study 1

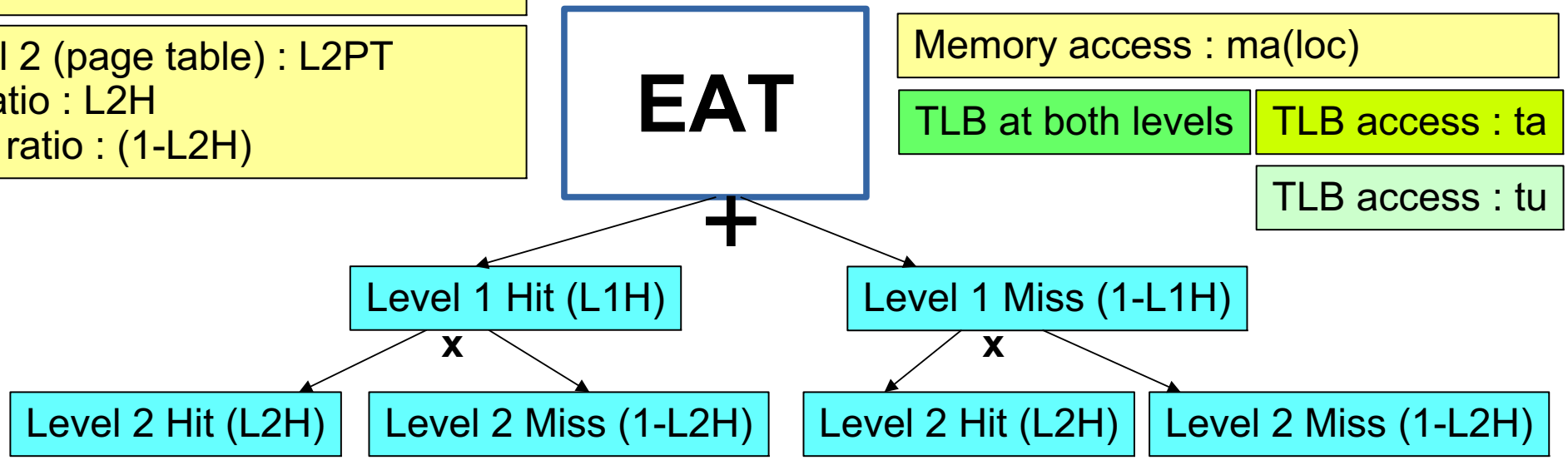
MULTICS Address Translation Scheme



(Real) Segmentation+Paging

Level 1 (segment table) : L1ST
Hit ratio : L1H
Miss ratio : (1-L1H)

Level 2 (page table) : L2PT
Hit ratio : L2H
Miss ratio : (1-L2H)



Path 1 :

L1 Hit, **L2 Hit** – ta(1) + ta(2) + ma(loc)

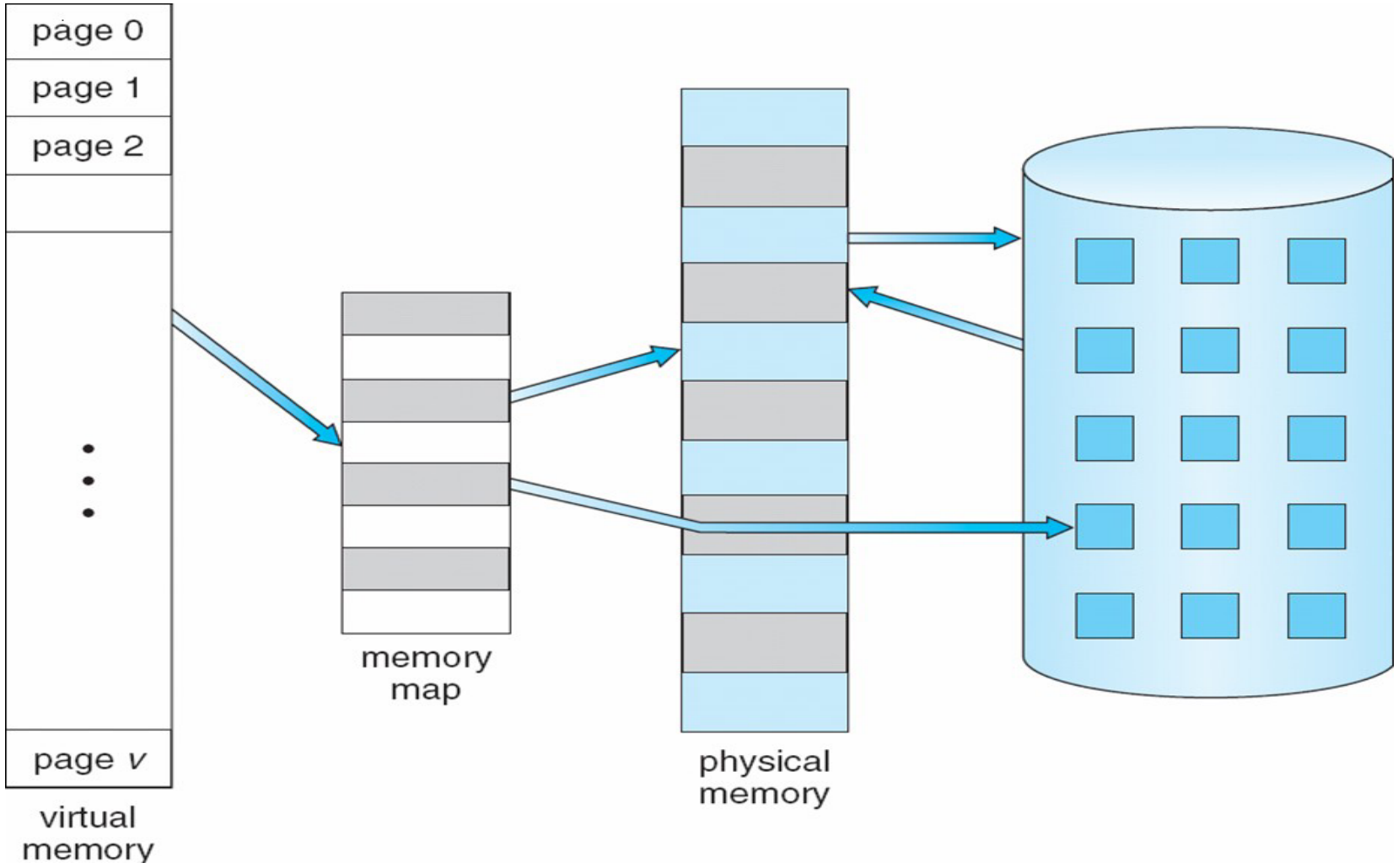
L1 Hit, **L2 Miss** – ta(1) + ta(2) + ma(L2PT) + **tu(2) +** ma(loc)

Path 2:

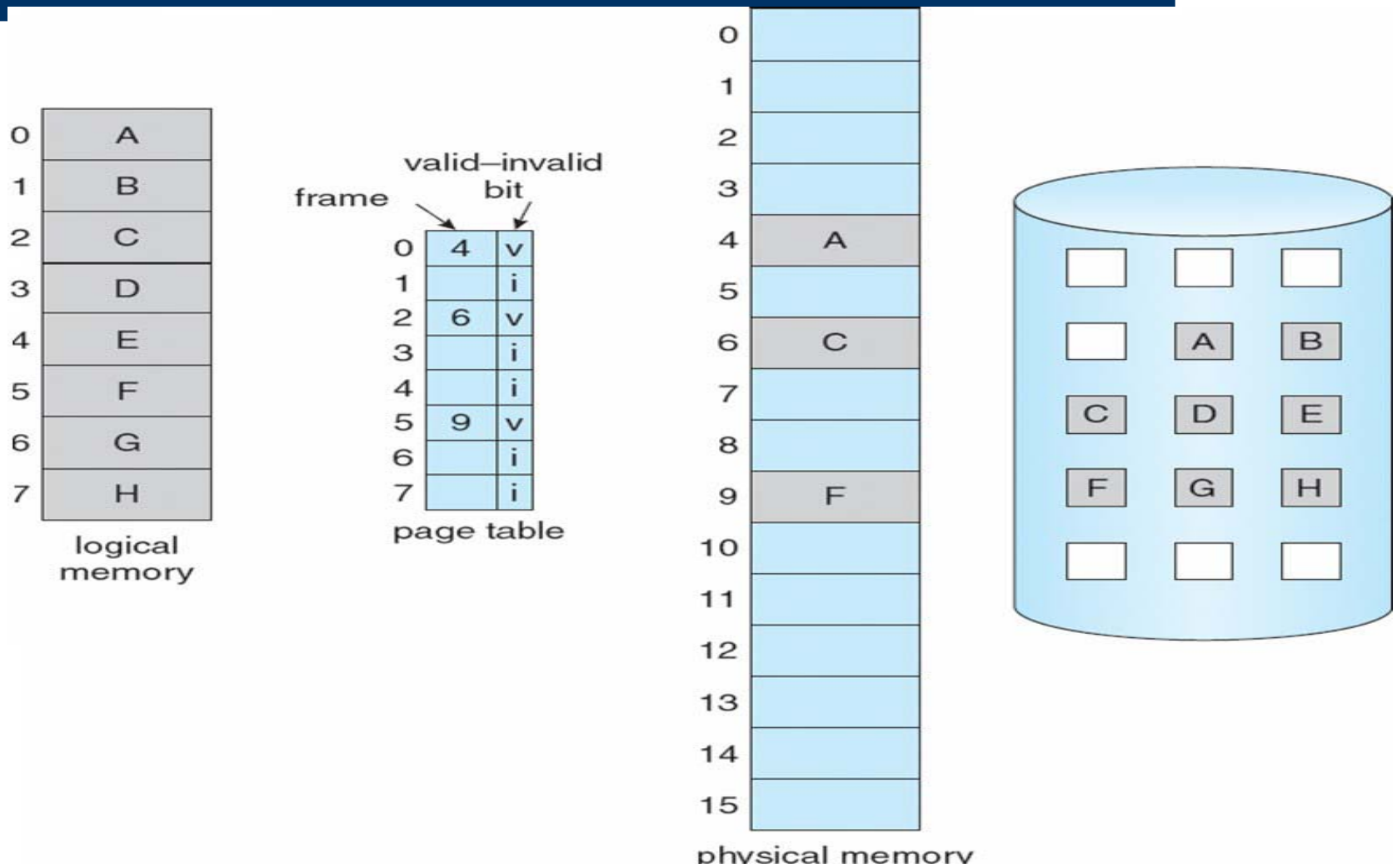
L1 Miss, **L2 Hit** – ta(1) + ma(L1ST) + **tu(1)** + ta(2) + ma(loc)

L1 Miss, **L2 Miss** – ta(1) + ma(L1ST) + **tu(1)** + ta(2) + ma(L2PT) + **tu(2) +** ma(loc)

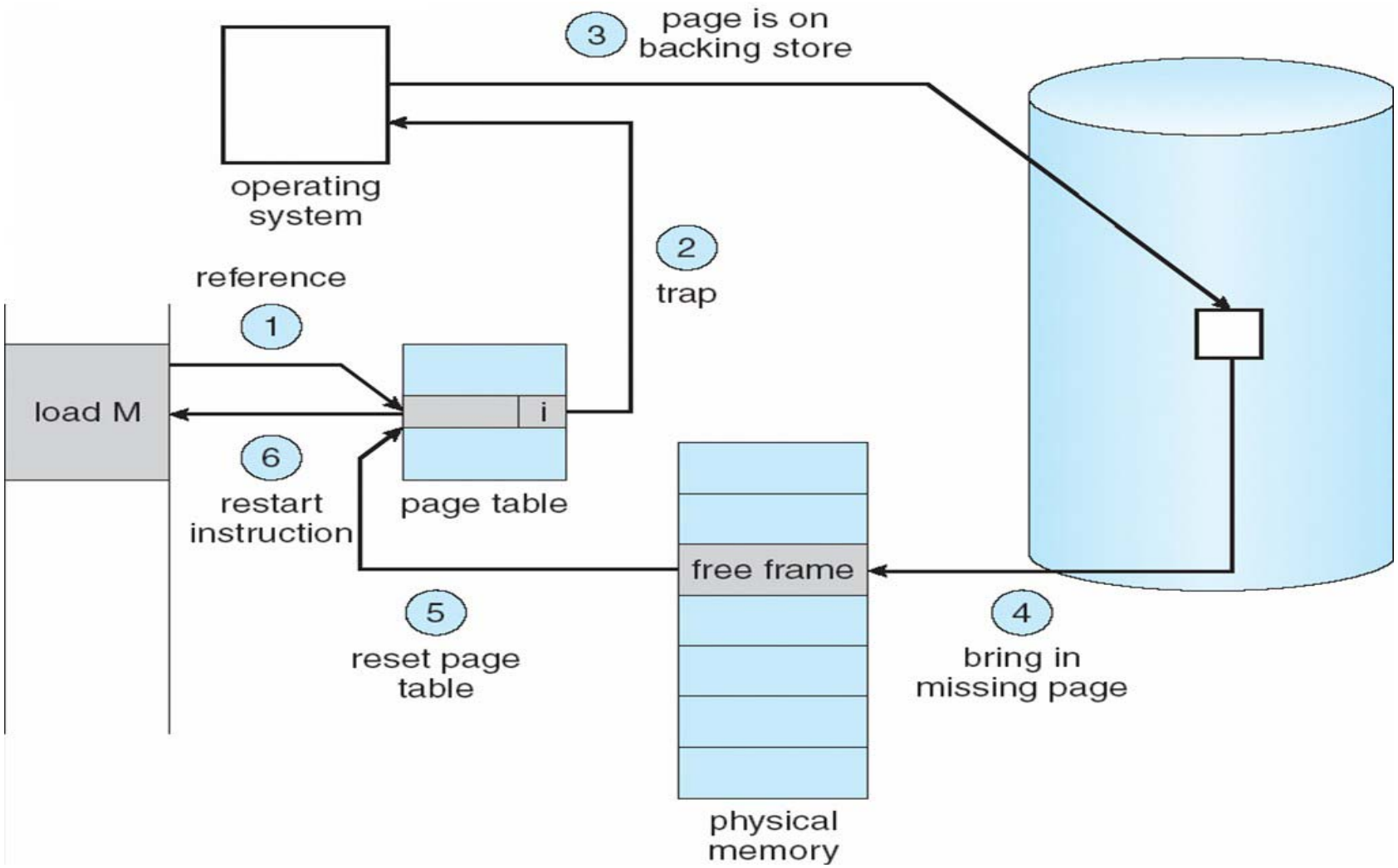
Virtual Memory that is larger than Physical Memory



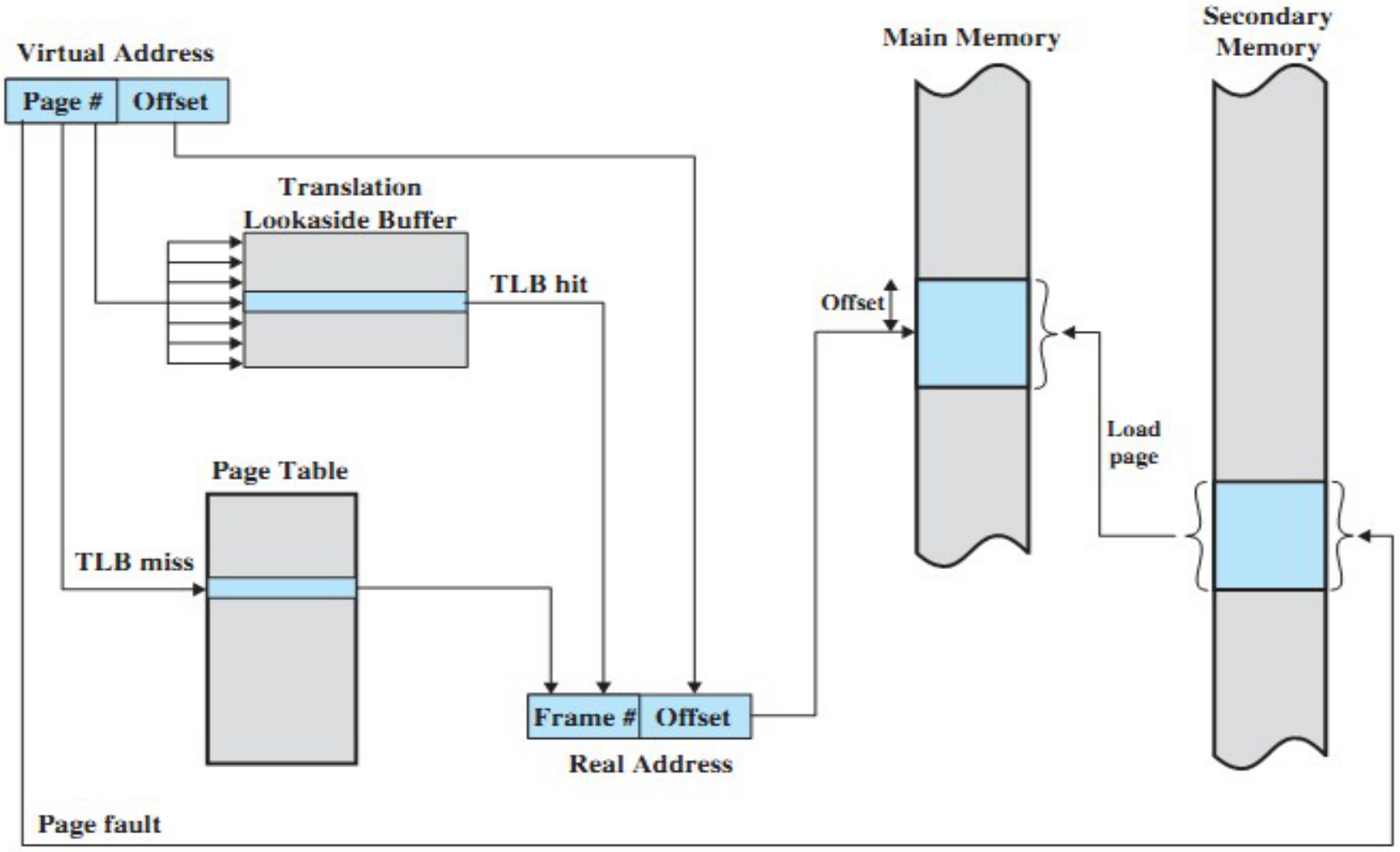
Page Table when some Pages are not in Main Memory



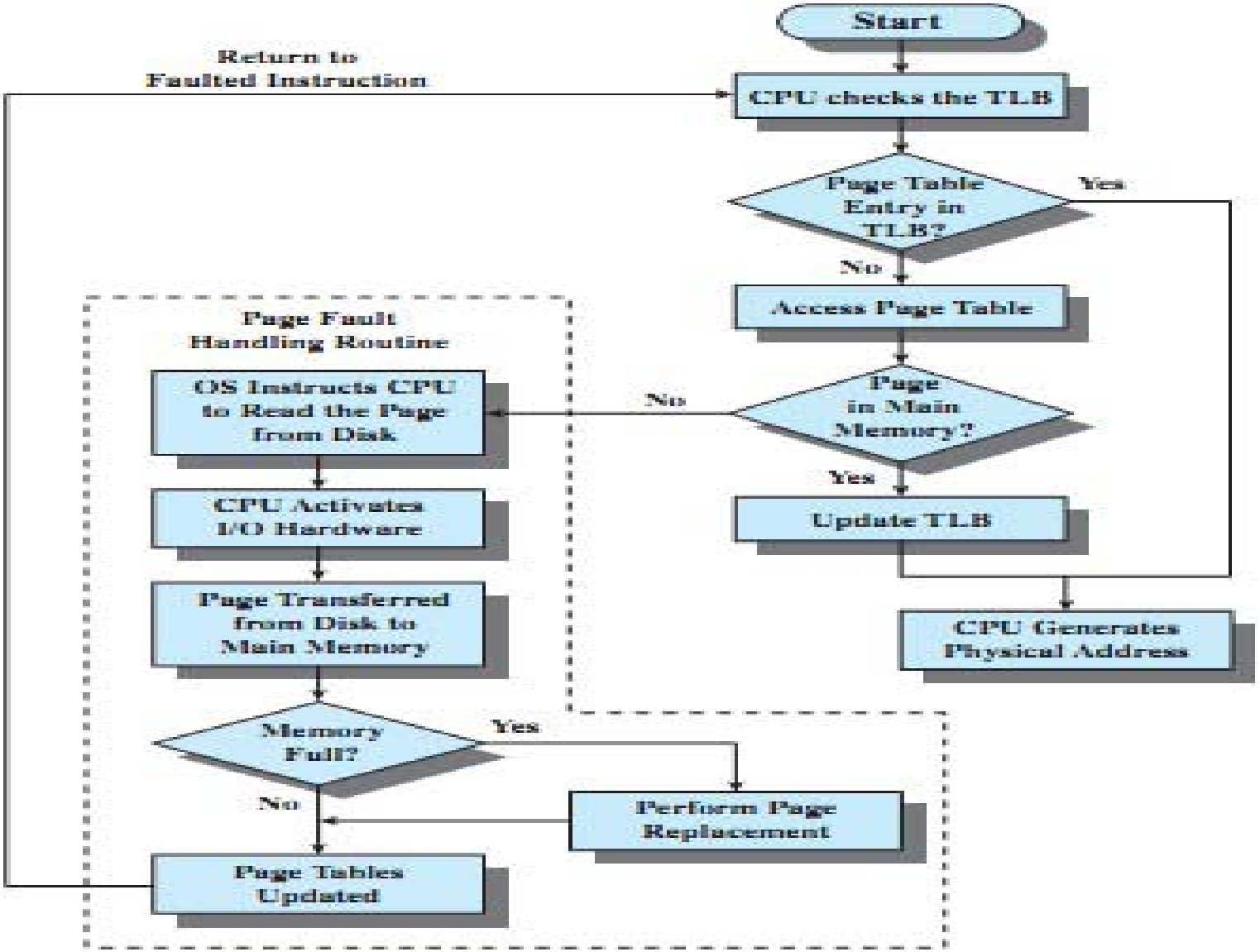
Steps in handling a Page Fault (1)



Use of a Translation Look-aside Buffer



Operation of Paging and TLB



Practice Exercise 11

Assume a demand paged memory management system with the following characteristics:

page size = 4K = 2^{12} bytes
physical address space = 2^{24} bytes
logical address space = 2^{32} bytes
TLB size = 2^{6} bytes

a. How many bits are needed to specify a logical address? Give the number of bits for the page number and offset.

b. How many page table entries can fit in the TLB if each entry only contains the information needed for logical to physical translation. Show your work.

c. How many page table entries are needed when a program uses its full logical address space?

d. Part (c) indicates a serious problem that arises from having a very large logical address space. What is this problem and how could an OS solve it? Discuss the consequences of your solution for runtime overhead during program execution (just a few sentences).

Solution 11

a. logical address space = 32 bits
page number = 20 bits; offset = 12 bits

b. In TLB, need page number (20 bits), frame number (12)
==> 32 bits for each entry
==> 4 bytes for each entry
TLB = 2^6 bytes or 64 bytes
==> 16 entries

c. 2^{20} page table entries are needed for the 2^{20} pages

d. PROBLEM: page table is very large ... it won't fit on one page and the OS won't want to keep the whole table in memory at all times
SOLUTION: page the page table
==> leads to page faults for page table pages
==> more I/O swapping

Practice Exercise 12

Suppose:

TLB lookup time = 20 ns
TLB hit ratio = 80%
memory access time = 75 ns
swap page time = 500,000 ns
50% of pages are dirty
OS uses a single level page table

a. What is the effective access time (EAT) if we assume the page fault rate is 0%? (Show your work.) Assume the cost to update the TLB, the page table, and the frame table (if needed) is negligible.

b. What is the effective access time (EAT) if we assume the page fault rate is 10%? (Show your work.) Assume the cost to update the TLB, the page table, and the frame table (if needed) is negligible.

Solution 12

a. EAT = 0.8 (TLB + MEM) + 0.2 (TLB + MEM + MEM) (As asked)

Or

a. EAT = 0.8 (TLB + MEM) + 0.2 (TLB + MEM + MEM + TLB)

Update

b. EAT = 0.8 (TLB + MEM) + 0.2 (0.9 (TLB + MEM + MEM)
+ 0.1 (TLB + MEM + 0.5 (DISK) + 0.5 (2 DISK) + MEM)) (As asked)

Or

b. EAT = 0.8 (TLB + MEM) + 0.2 (0.9(TLB + MEM + MEM + TLB)
+ 0.1 (TLB + MEM + 0.5 (DISK) + 0.5 (2 DISK) + MEM + MEM + TLB))

Update

Update

Solution 12

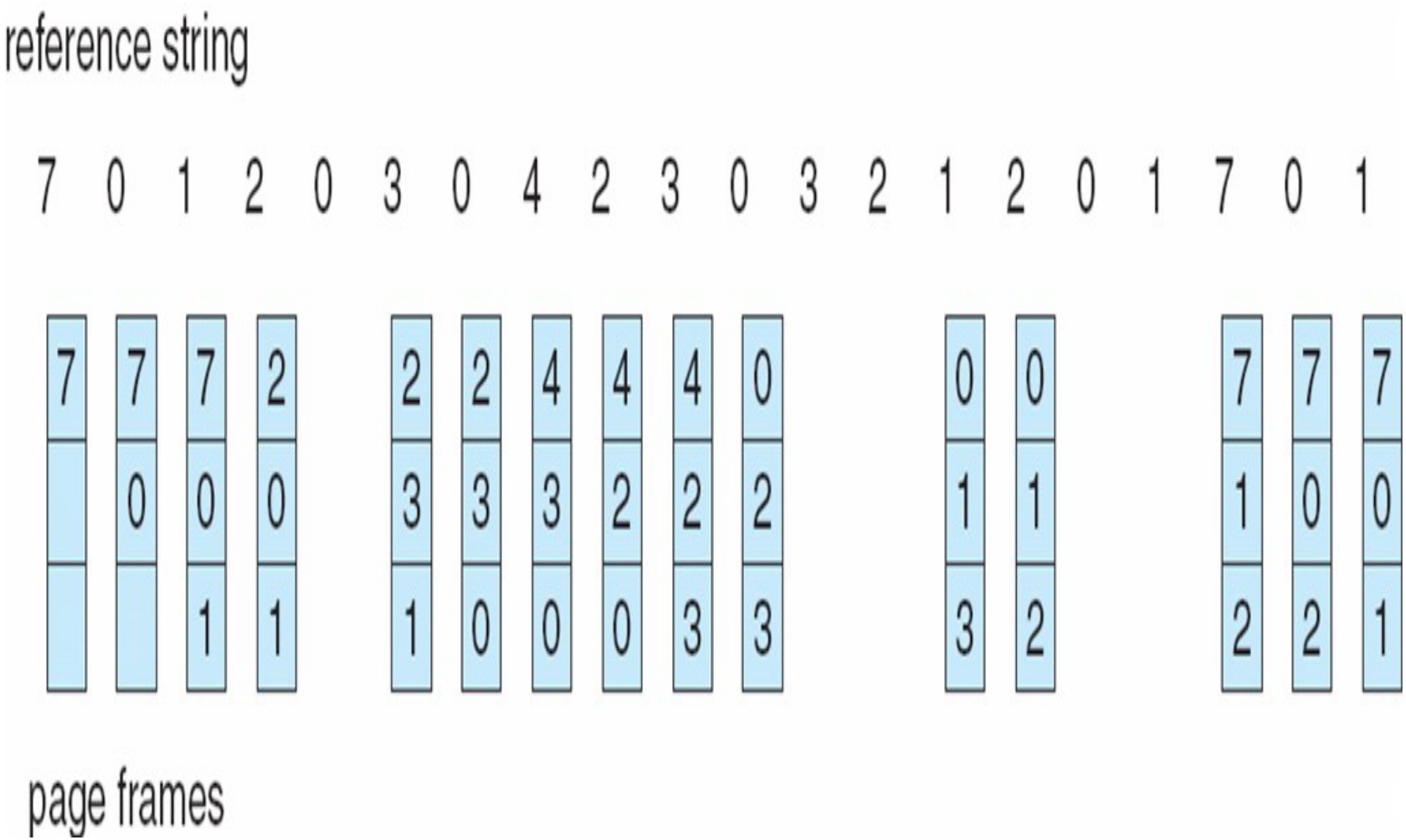
a. EAT = 0.8 (TLB + MEM) + 0.2 (TLB + MEM + MEM) (As asked)
Or
a. EAT = 0.8 (TLB + MEM) + 0.2 (TLB + MEM + MEM + TLB)

b. EAT = 0.8 (TLB + MEM) + 0.2 (0.9 (TLB + MEM + MEM)
+ 0.1 (TLB + MEM + 0.5 (DISK) + 0.5 (2 DISK) + MEM)) (As asked)
Or
b. EAT = 0.8 (TLB + MEM) + 0.2 (0.9(TLB + MEM + MEM + TLB)
+ 0.1 (TLB + MEM + 0.5 (DISK) + 0.5 (2 DISK) + MEM + MEM + TLB))

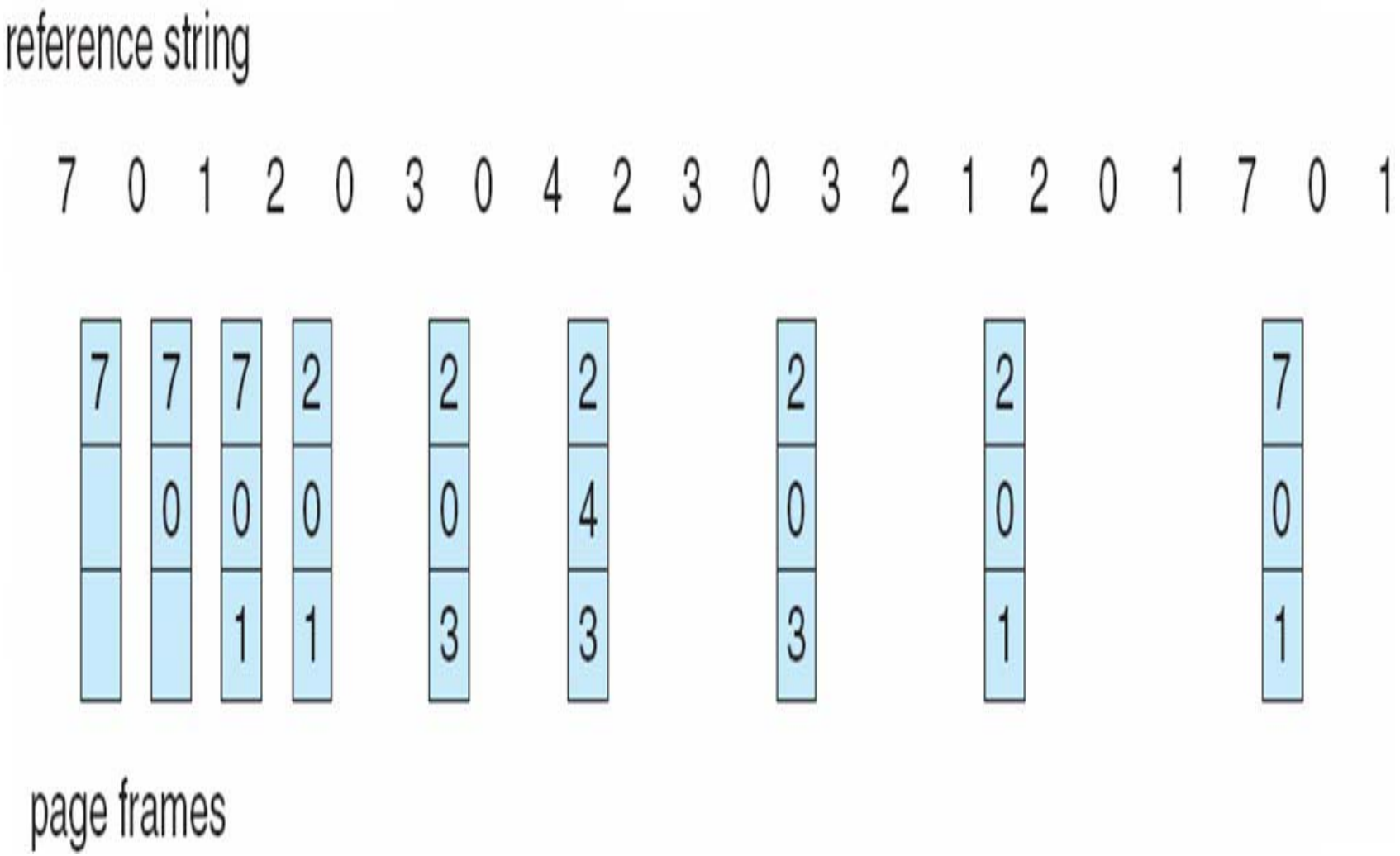
a. EAT = 0.8 (TLB + MEM) + 0.2 (TLB + MEM + MEM)
EAT = 0.8 (20 + 75) + 0.2 (20 + 75 + 75)
EAT = 76 + 34 = 110 ns

b. EAT = 0.8 (TLB + MEM) + 0.2 (0.9(TLB + MEM + MEM) + 0.1 (TLB + MEM + 0.5 (DISK) + 0.5 (2 DISK) + MEM))
EAT = 0.8 (20 + 75) + 0.2 (0.9 (20 + 75 + 75) + 0.1 (20 + 75 + 0.5 (500000)
+ 0.5 (1000000) + 75))
EAT = 76 + 0.2 (153 + .1 (750170)) = 76 + 15034 = 15110 ns

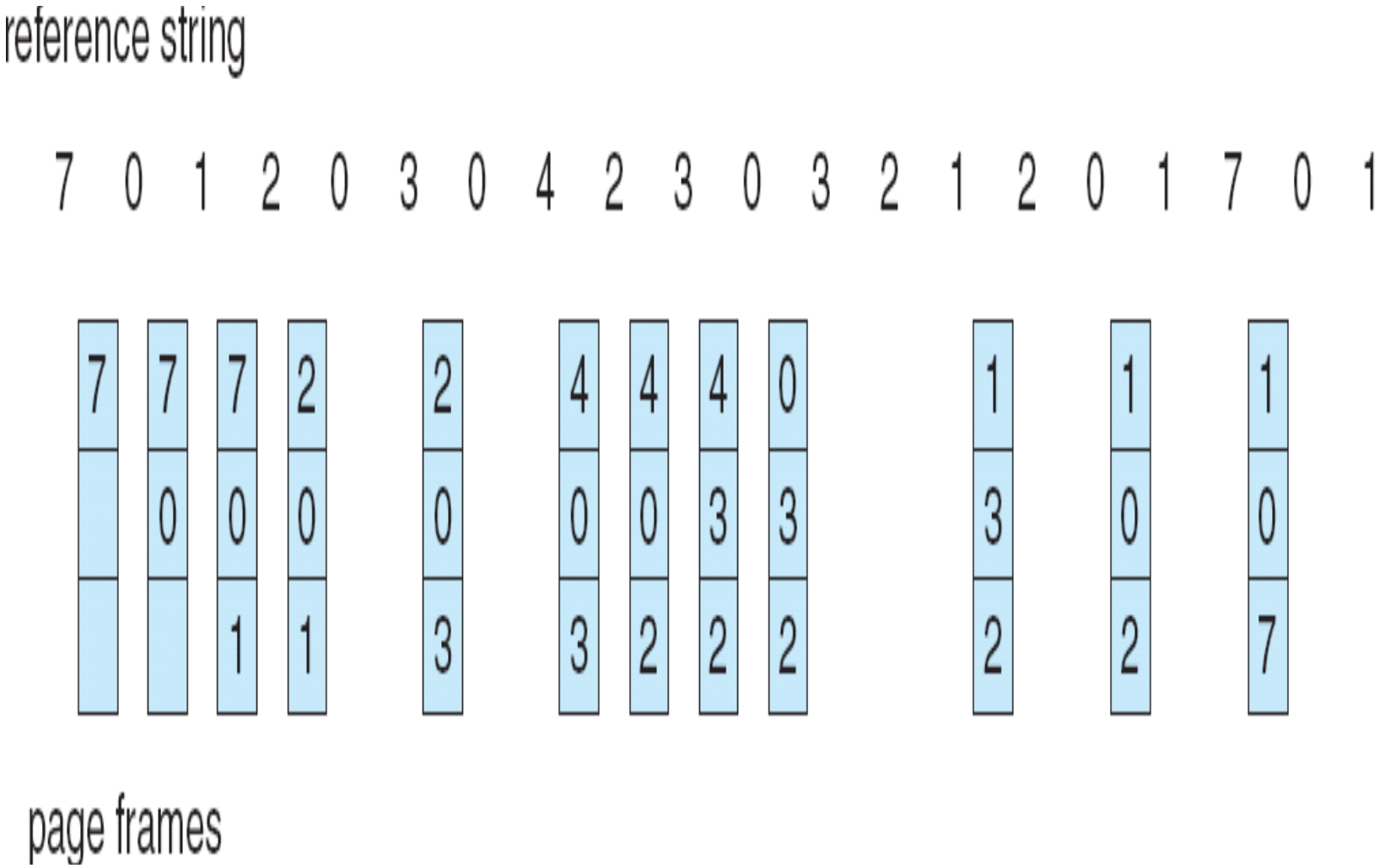
FIFO Page Replacement



Optimal Page Replacement



LRU Page Replacement



Local vs. Global Replacement Policies