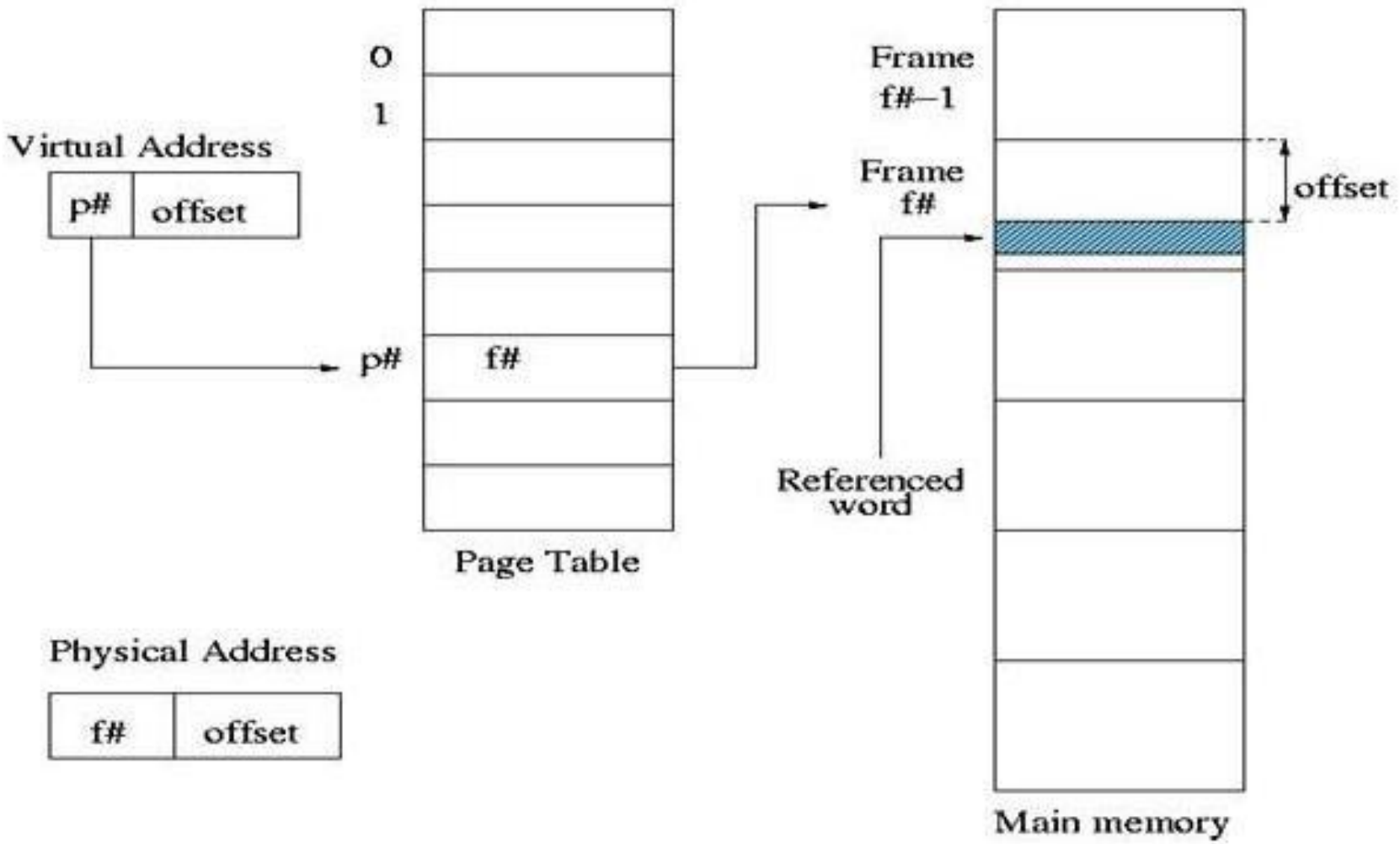
운영체제론 실습 13주차

정보보호연구실 @ 한양대학교

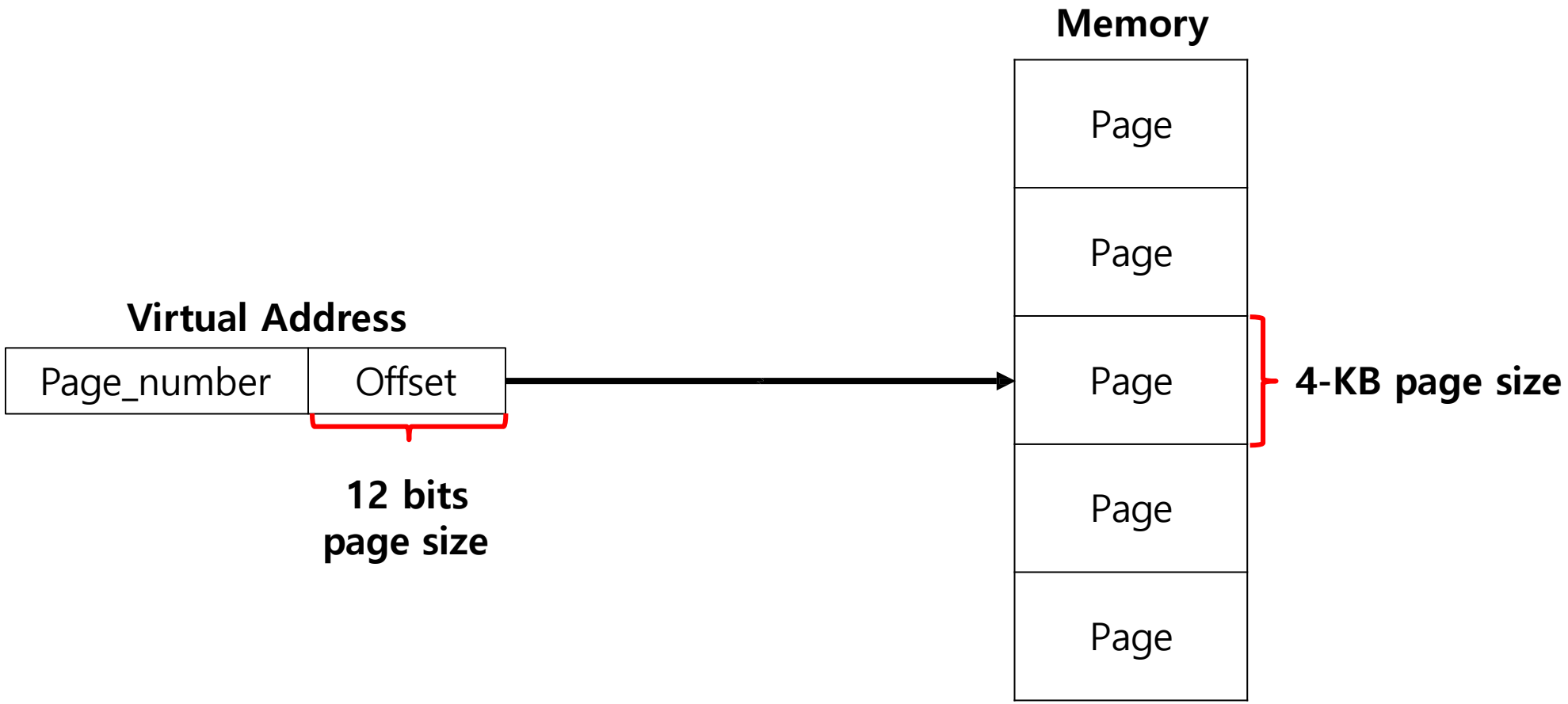
Paging

- **프레임과 페이지**
  - 프레임과 페이지는 메모리를 일정한 크기의 공간으로 나누어 관리하는 단위
    - 페이지는 가상메모리를 일정한 크기로 나눈 블록 단위
    - 프레임은 물리메모리를 일정한 크기로 나눈 블록 단위
  - Paging이란 페이지를 가상기억장치에 편성해 운용하는 기법
    - 프레임에 요청한 페이지가 이미 있으면 Page Hit
    - 프레임에 요청한 페이지가 없으면 Page fault
  - Paging-replacement란 페이지가 없을 경우 페이지를 교체하기 위한 기법
    - First in First Out(FIFO)
    - Least Recently Used(LRU)
    - Optimal Algorithm

Paging


0
1
p#
f#
Page Table
Virtual Address
p#
offset
Physical Address
f#
offset
Frame
f#–1
Frame
f#
offset
Referenced
word
Main memory

Paging


Memory
Page
Page
Page
Page
Page
Virtual Address
Page_number
Offset
12 bits
page size
4-KB page size

Programming Problems

9.28 Assume that a system has a 32-bit virtual address with a 4-KB page size. Write a C program that is passed a virtual address (in decimal) on the command line and have it output the page number and offset for the given address. As an example, your program would run as follows:

```
./addresses 19986
```

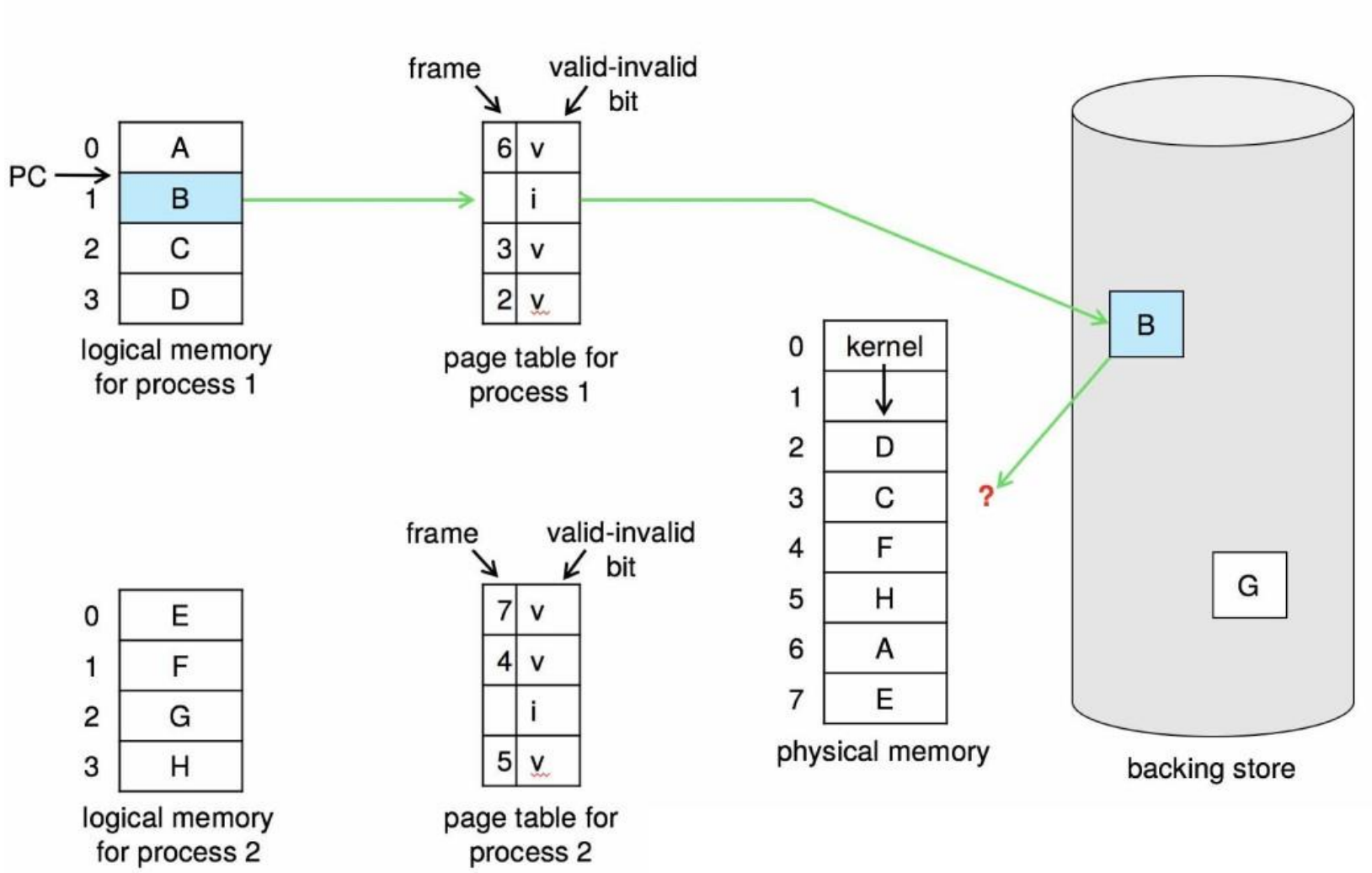
Your program would output:

```
The address 19986 contains:
page number = 4
offset = 3602
```

Writing this program will require using the appropriate data type to store 32 bits. We encourage you to use `unsigned` data types as well.

주어진 메모리 주소의 page number와 offset을 계산하는 프로그램을 작성해 보자.

Page-replacement



Page-replacement(FIFO)

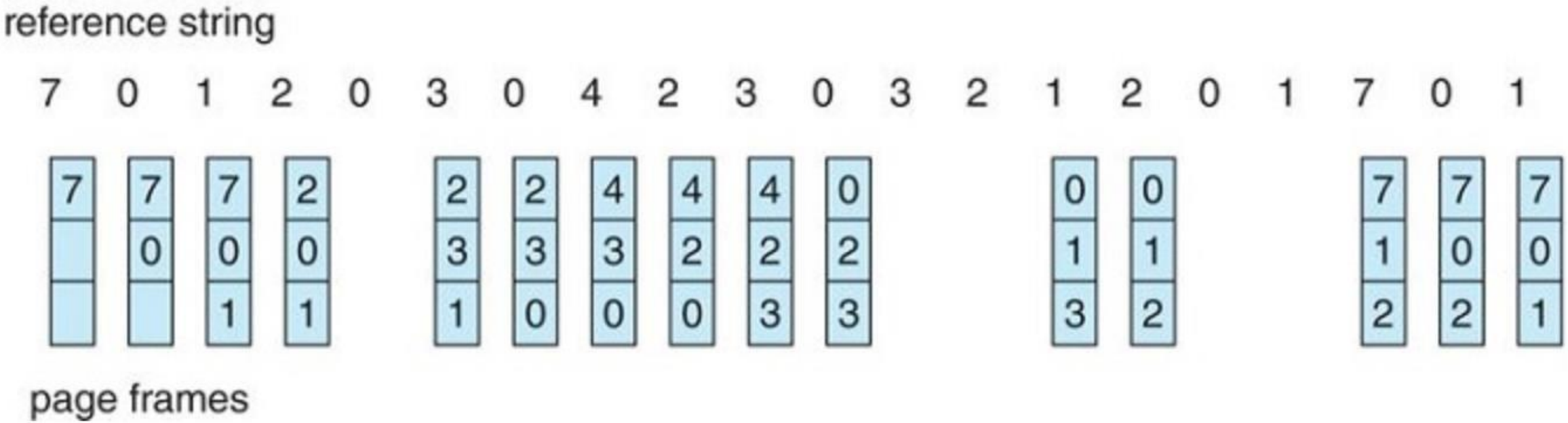


Figure 10.12 FIFO page-replacement algorithm.

Page-replacement(FIFO)

```python
def FIFO(size, pages):
    SIZE = size
    memory = []
    faults = 0
    for page in pages:
        if memory.count(page) == 0 and len(memory) < SIZE:
            memory.append(page)
            faults += 1
        elif memory.count(page) == 0 and len(memory) == SIZE:
            memory.pop(0)
            memory.append(page)
            faults += 1
    return faults
```

```
FIFO
[7]
[7, 0]
[7, 0, 1]
[0, 1, 2]
[1, 2, 3]
[2, 3, 0]
[3, 0, 4]
[0, 4, 2]
[4, 2, 3]
[2, 3, 0]
[3, 0, 1]
[0, 1, 2]
[1, 2, 7]
[2, 7, 0]
[7, 0, 1]
15 page faults.
```

Page-replacement(LRU)

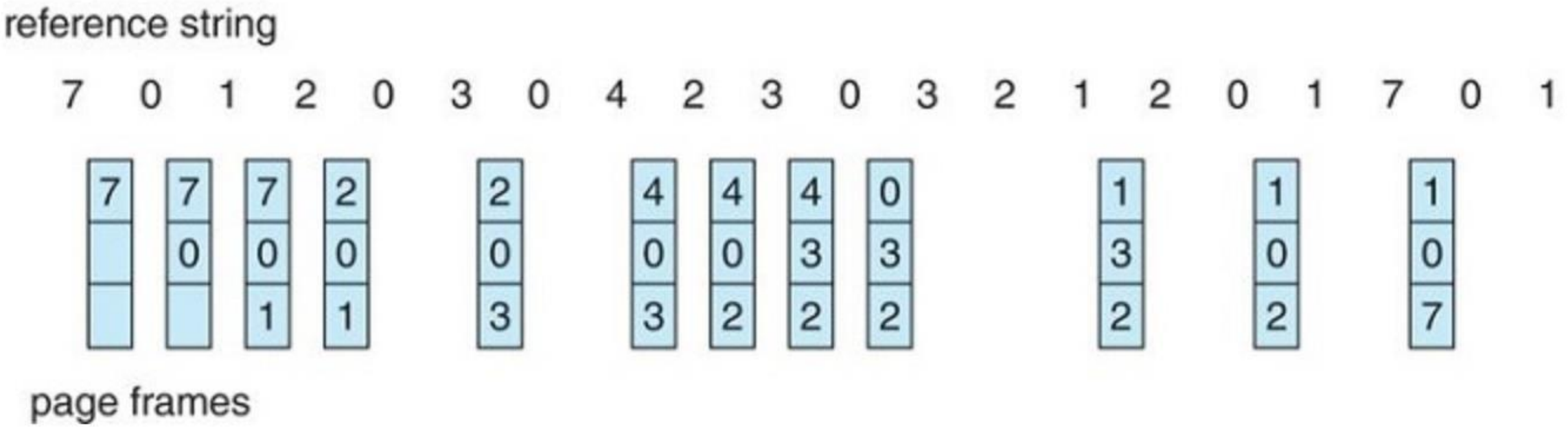


Figure 10.15 LRU page-replacement algorithm.

Page-replacement(LRU)

```python
def LRU(size, pages):
    SIZE = size
    memory = []
    faults = 0
    for page in pages:
        if memory.count(page) == 0 and len(memory) < SIZE:
            memory.append(page)
            faults += 1
        elif memory.count(page) == 0 and len(memory) == SIZE:
            memory.pop(0)
            memory.append(page)
            faults += 1
        elif memory.count(page) > 0:
            memory.remove(page)
            memory.append(page)
    return faults
```

```
LRU
[7]
[7, 0]
[7, 0, 1]
[0, 1, 2]
[2, 0, 3]
[3, 0, 4]
[0, 4, 2]
[4, 2, 3]
[2, 3, 0]
[3, 2, 1]
[1, 2, 0]
[0, 1, 7]
12 page faults.
```

Page-replacement

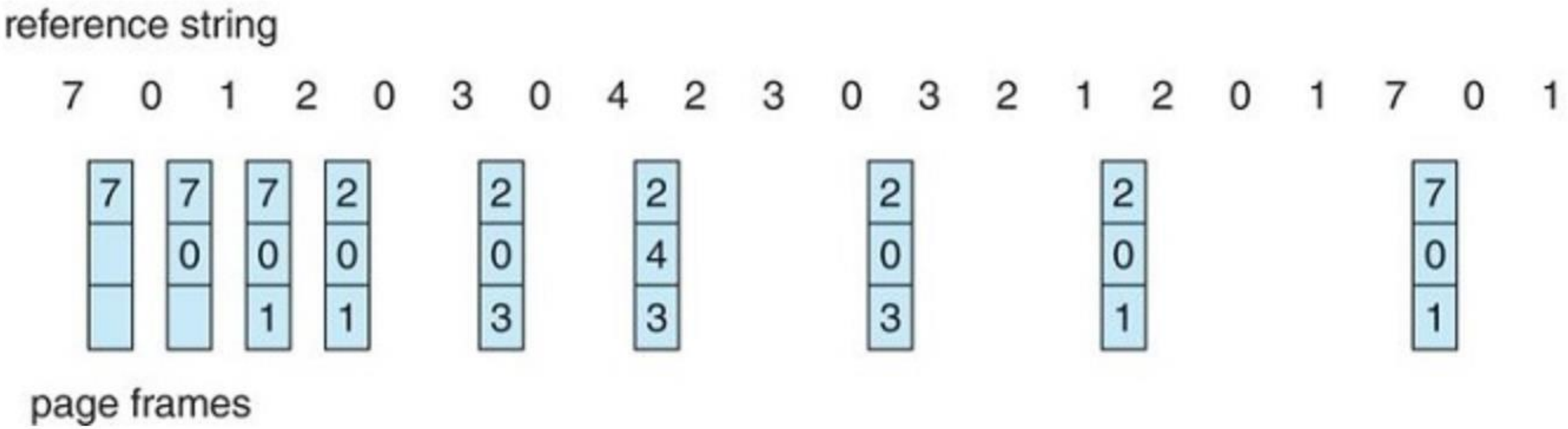


Figure 10.14 Optimal page-replacement algorithm.

Programming Problems

10.44 Write a program that implements the FIFO, LRU, and optimal (OPT) page-replacement algorithms presented in Section 10.4. Have your program initially generate a random page-reference string where page numbers range from 0 to 9. Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm. Pass the number of page frames to the program at startup. You may implement this program in any programming language of your choice. (You may find your implementation of either FIFO or LRU to be helpful in the virtual memory manager programming project.)

FIFO, LRU 알고리즘을 참고하여 OPT 알고리즘을 구현해 보자.