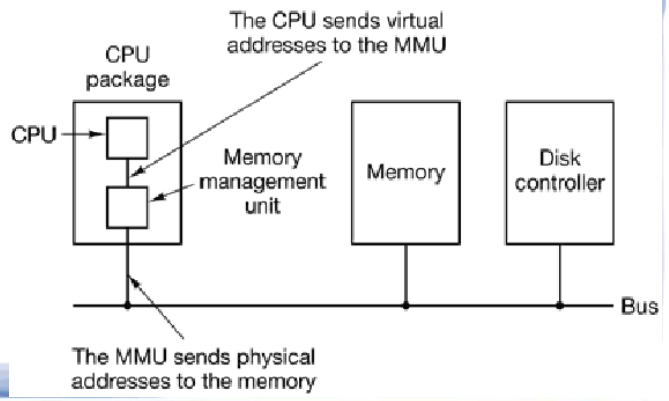
# Memory Management (2)

## Paging

- **Virtual addresses** : program-generated addresses, it form **Virtual address space**.
- **Memory Management Unit (MMU)** : maps the virtual addresses onto the physical memory addresses

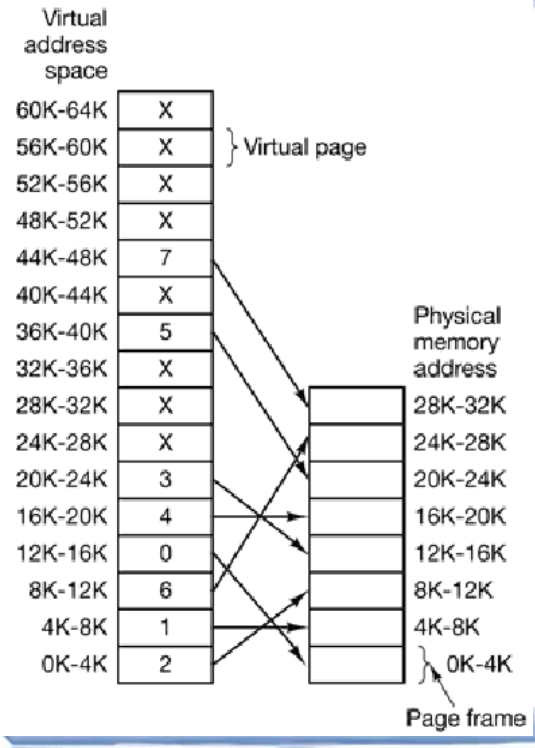



## VIRTUAL MEMORY

- Problem : programs were too big to fit in the available memory
- Solution :
  - 1. overlay (split the program into pieces)
    - Swapping overlay done by system, however splitting the program has to done by the programmer
  - 2. Virtual memory
    - Basic idea : programs may exceed the amount of physical memory available for it
    - Most virtual memory system use a technique called paging,

## Paging

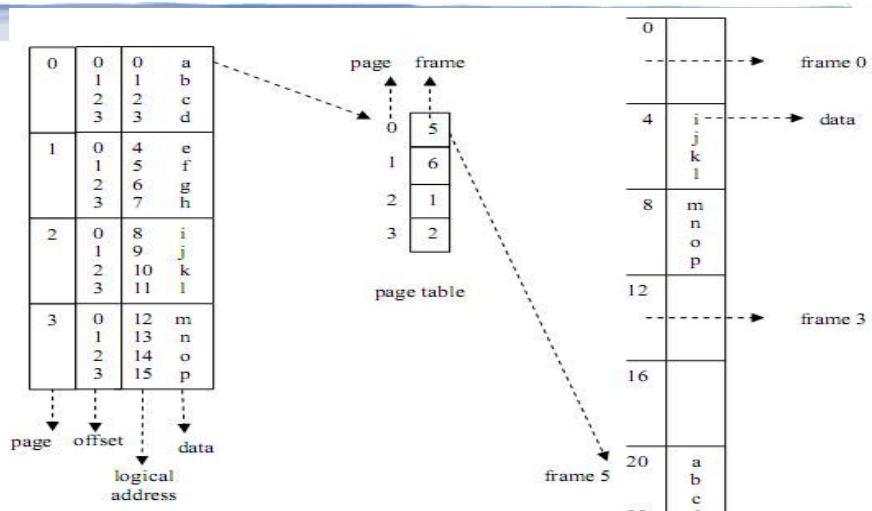
- Virtual address space is divided up into units called **pages**
- The corresponding units in the physical memory are called **page frames**
- **Page fault / fault** = happened if program tries to use an unmapped page.
  - **The operating system picks a little-used page frame and writes its contents back to the disk.**
  - **It then fetches the page just referenced into the page frame just freed, changes the map, and restarts the trapped instruction.**




**MOV REG,0**
**0 (virtual address)**
**Page 0**
**Frame 2**
**Physical address is 8192**

**MOV REG,20500**
**20 bytes from the start of virtual page 5 (virtual addresses 20480)**
**Frame 3**
**Physical address = 12288+20 = 12308**

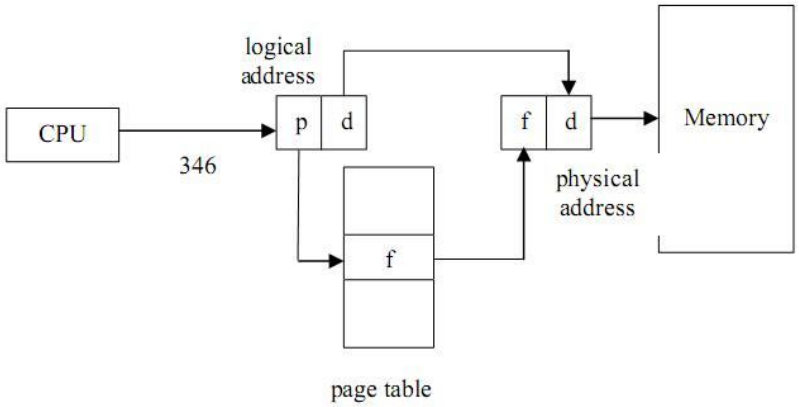
**MOV REG, 33000 – PAGE FAULT**




- Virtual address 0 mapped to physical address 20 ((5 x 4) + 0).
- Virtual address 5 mapped to physical address 25 ((6 x 4) +1).

# Paging

- Virtual address consist of 2 part :
  - page number p
  - offset page d




Page table = table for all processes in the memory

# Page Replacement Algorithm

- When a page fault occurs, the operating system has to choose a page to remove from memory to make room for the page that has to be brought in.
- Objective : Minimum page fault

# Page Replacement Algorithm

- Random
- FIFO (First In First Out)
- Optimal
- LRU (Least Recently Used)

# Optimal

- Replace the page that will not be used in the near future

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| F | F | F | F | | F | | F | | | F | | | F | | | | F | | |

- Impossible to implement because the operating system has no way of knowing when each of the pages will be referenced next.

# FIFO (First In First Out)

- Replace the oldest page

| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 7 | 7 |
| | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 |
| F | F | F | F | | F | F | F | F | F | F | | | F | F | | | F | F | F |

- Number of faults : 15

# LRU (Least Recently Used)

- Replace the page that is not recently used

| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 |
| | | 1 | 1 | 1 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 |
| F | F | F | F | | F | | F | F | F | F | | | F | | F | | F | | |