EC-252: COMPUTER ARCHITECTURE AND MICROPROCESSORS

Vaskar Raychoudhury

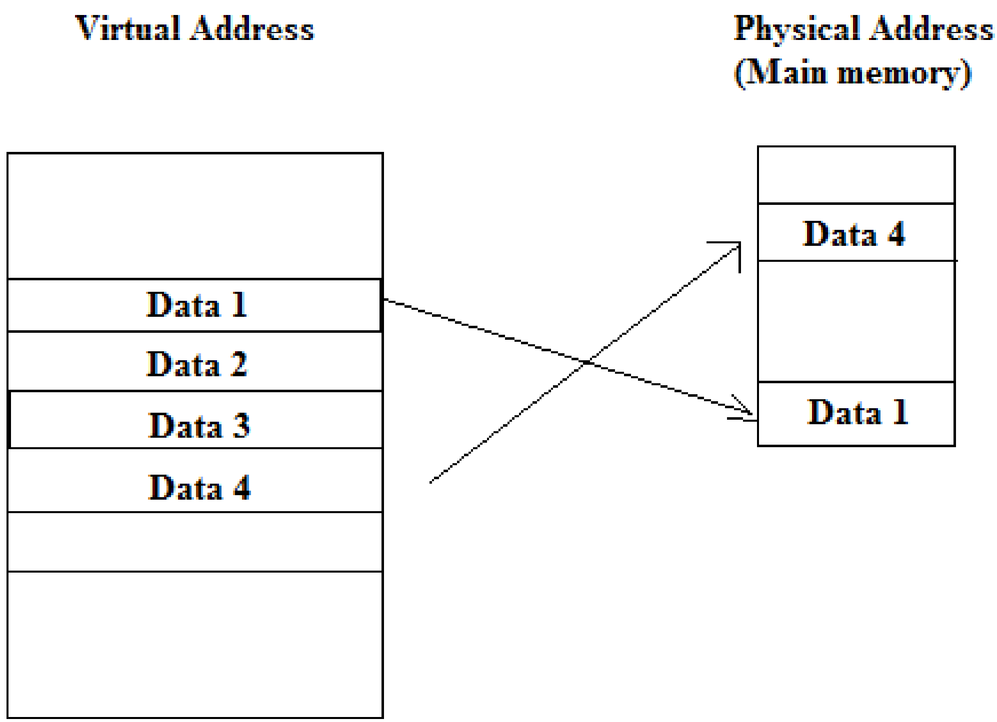
Indian Institute of Technology Roorkee

Virtual Memory

- Virtual memory is an illusion
  - Programmers imagine of a vast memory space
- Each address referenced by CPU goes through an address mapping
  - From the virtual address to a physical address in main memory
- Virtual memory system provides a mechanism-
  - dynamically translating program-generated addresses into correct main memory locations

Virtual Memory (2)

- The address used by a programmer will be called a virtual address or logical address.
- An address in main memory is called a physical address

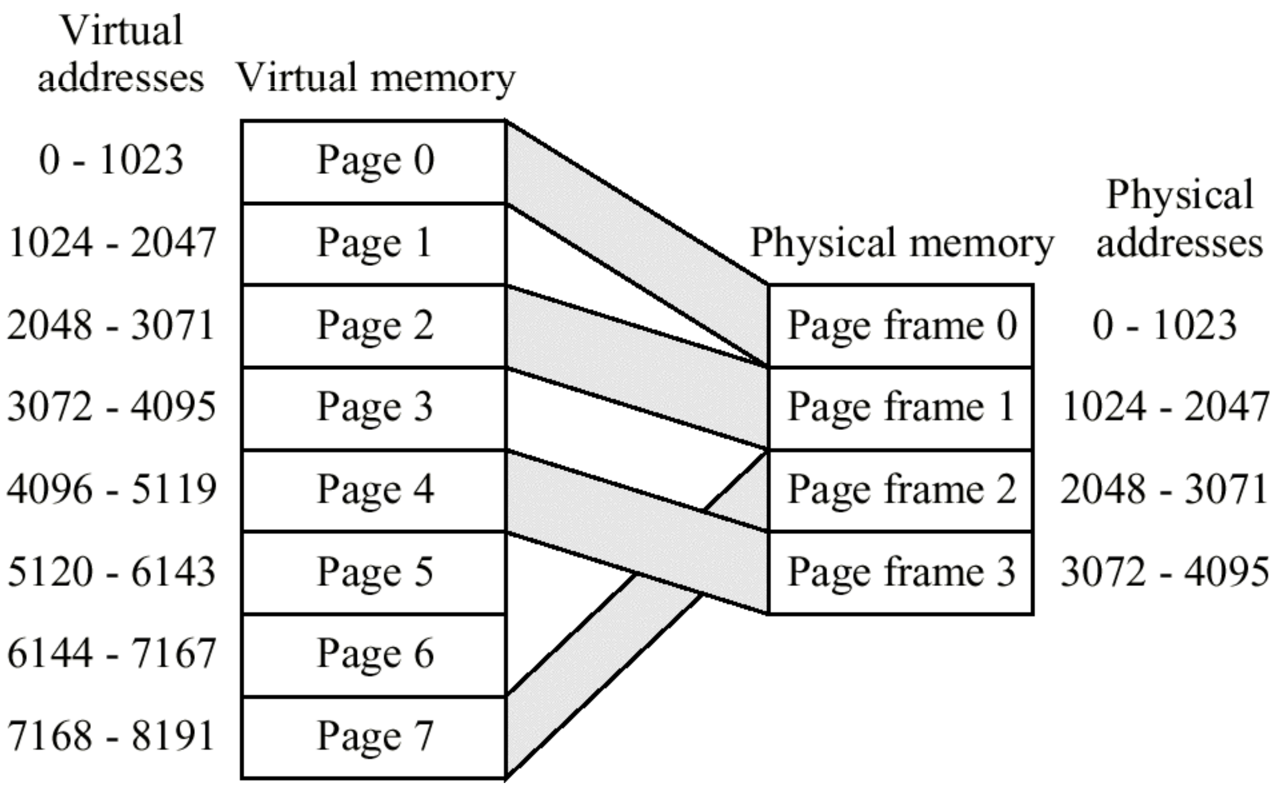


Virtual Memory (3)

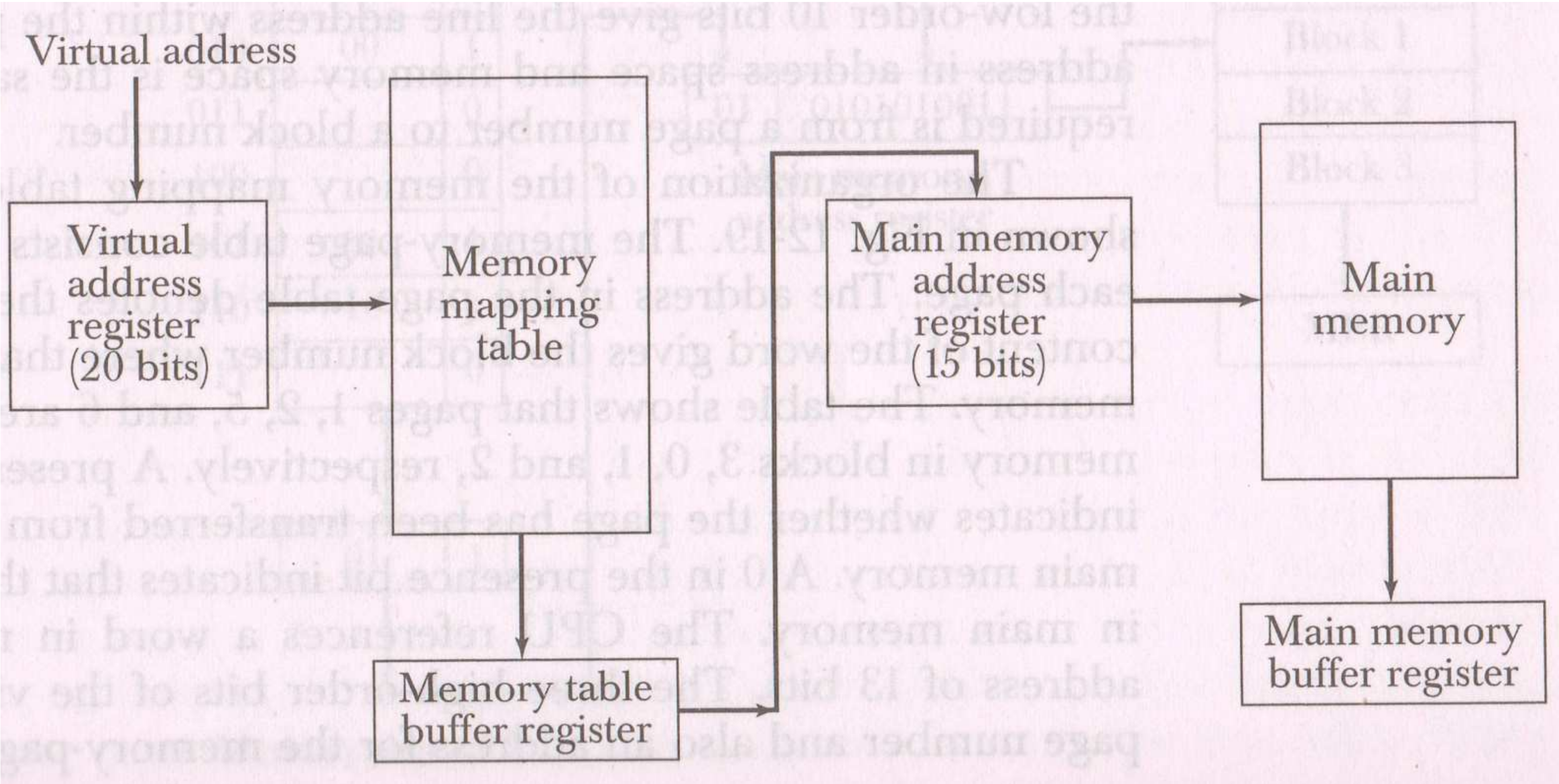
- Only part of the program needs to be in memory for execution
- Logical address space can therefore be much larger than physical address space
- Allows for more efficient process creation

Virtual Memory (4)

- Virtual memory is stored in a hard disk image
- The physical memory holds a small number of virtual pages in physical page frames



Memory Mapping Table



Memory Mapping Table (2)

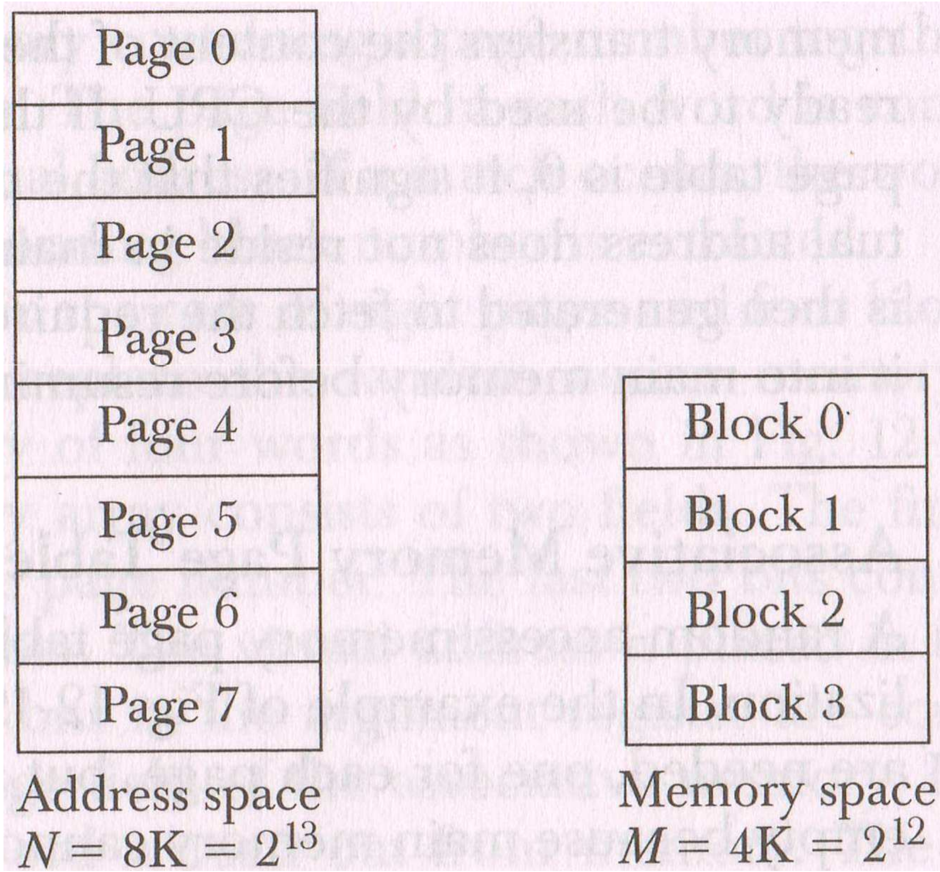
- Where to store the mem-mapping table?
  - In a separate memory
    - Add a new memory unit
    - Requires one extra memory access time
  - In main memory
    - Main memory space is decreased
    - Two accesses to the main memory are required – programs run at half speed
  - In an associative memory / CAM

Address Mapping Using Pages

- Address mapping is simplified if address space and memory space are divided into groups of fixed size
  - Physical memory is broken down into equal sized groups, called blocks (also called page frame)
  - Address space is broken down into same-size-groups, called pages

Pages and Blocks (example)

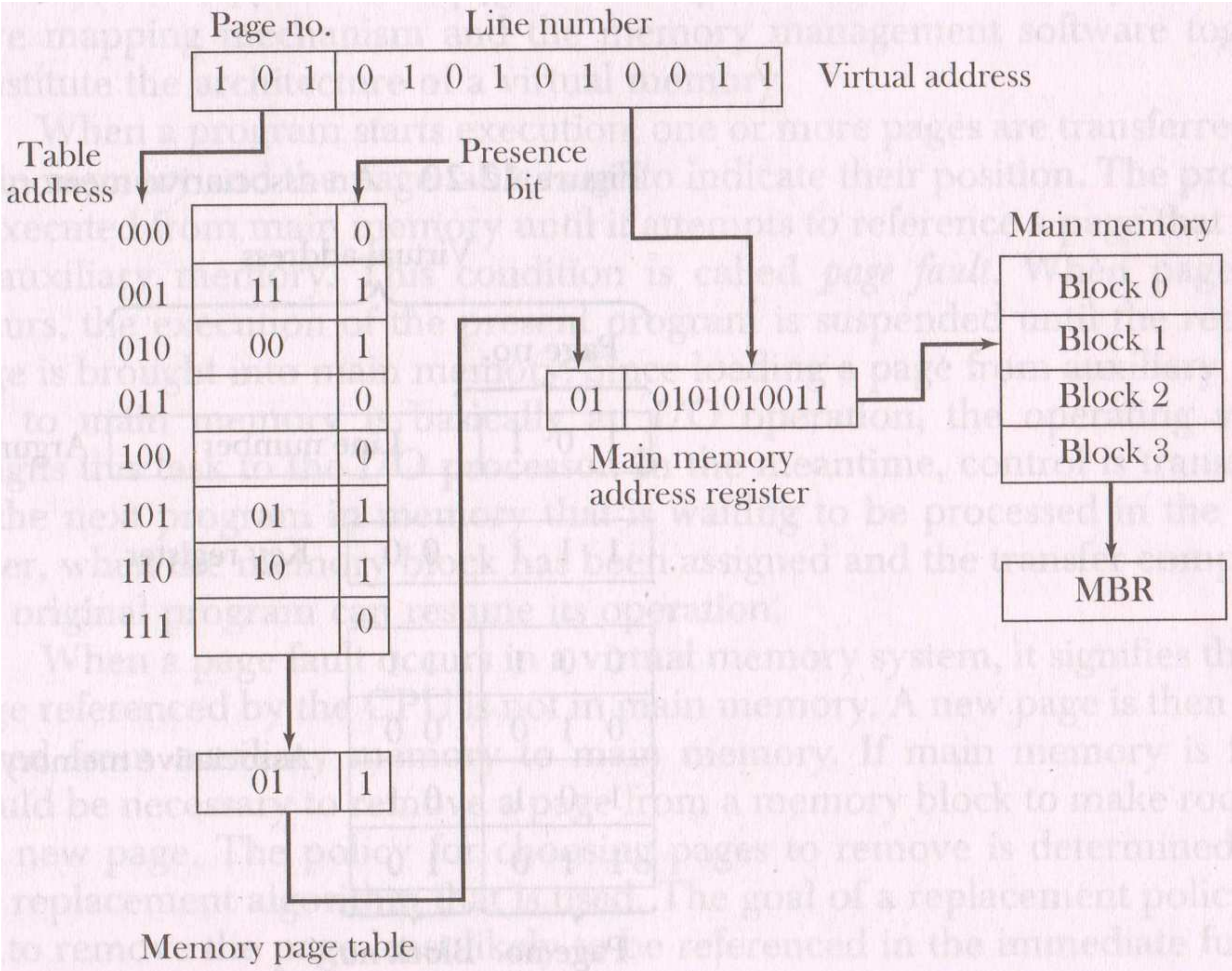
- E.g: if auxiliary memory contains 8K and main memory contains 4K and page size equals to 1K,
  - then auxiliary memory has 8 pages and main memory has 4 pages



Address Mapping Using Pages (2)

- A virtual address = a page number address + a line within the page
- If 2^P words per page, then
  - P lower order bits are used to specify a line address,
  - remaining higher order bits are used to specify the page number
- Previous example, 13 bit virtual address
  - 3 higher bits to identify a page and 10 bits to identify a line

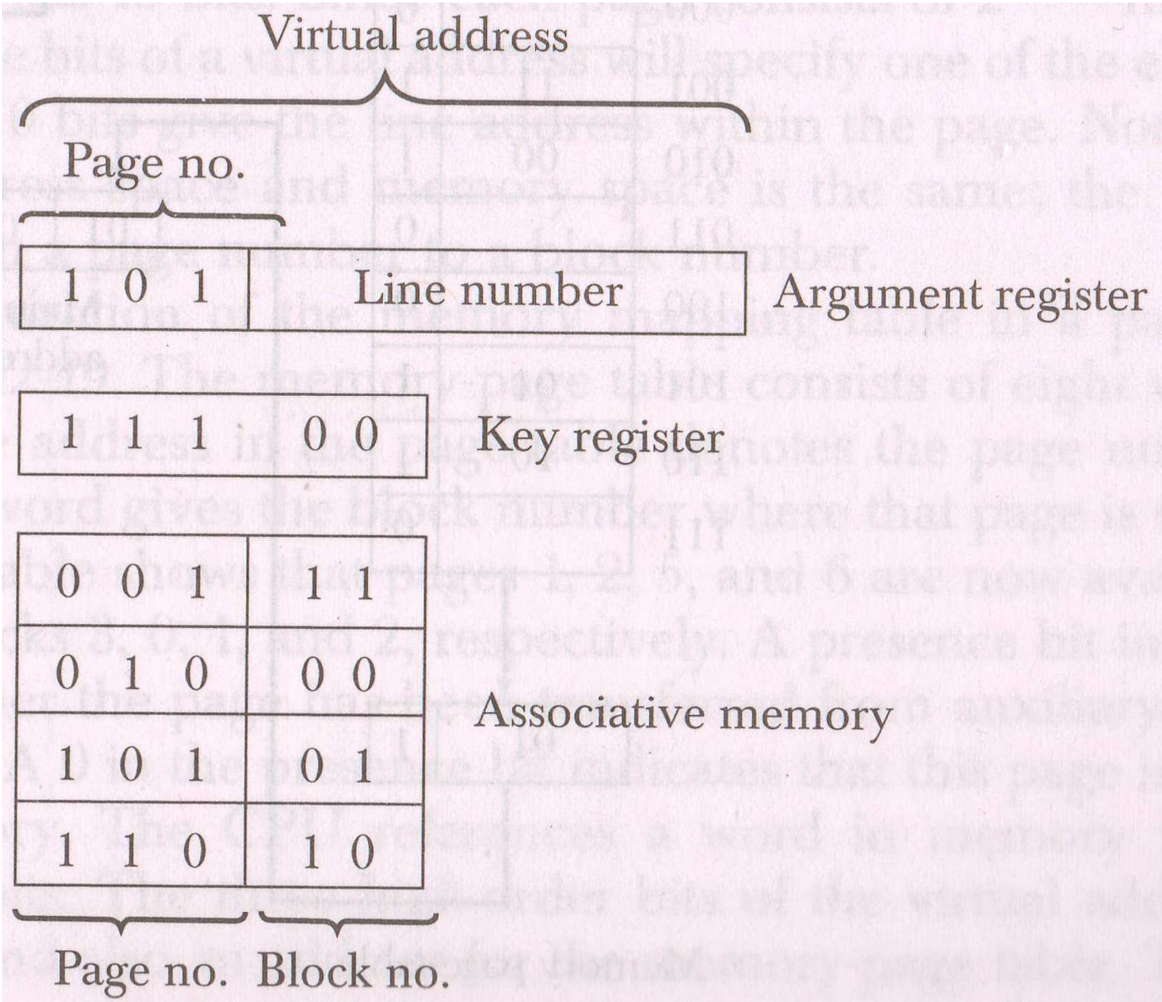
Memory Table in a Paged System



Associative Memory Page Table

- If address space 1024 K and memory space 32 K, with a 1K page/block size
  - Only 32 locations in the 1024 K wide memory page table will have a presence bit equal to 1
- Construct a memory page table with
  - equal number of words as number of blocks in main memory
  - Reduces memory size and increases usability
  - Use associative memory (CAM)

Associative Memory Page Table (2)



Page Replacement

- Virtual memory system
  - Software techniques
    - Which page to be removed from main memory?
    - When a new page should be brought in?
    - Where in main memory, a newly brought page is to be kept
  - Hardware techniques
    - Carry out the operations decided by the software system

Page Replacement (2)

- When a referenced page is not in main memory
  - Page fault occurs
  - New page needs to be transferred from auxiliary memory to main memory
- If main memory is full
  - Remove a suitable page to bring in the new one
  - Follow replacement algorithm
    - FIFO, LRU

Basic Page Replacement

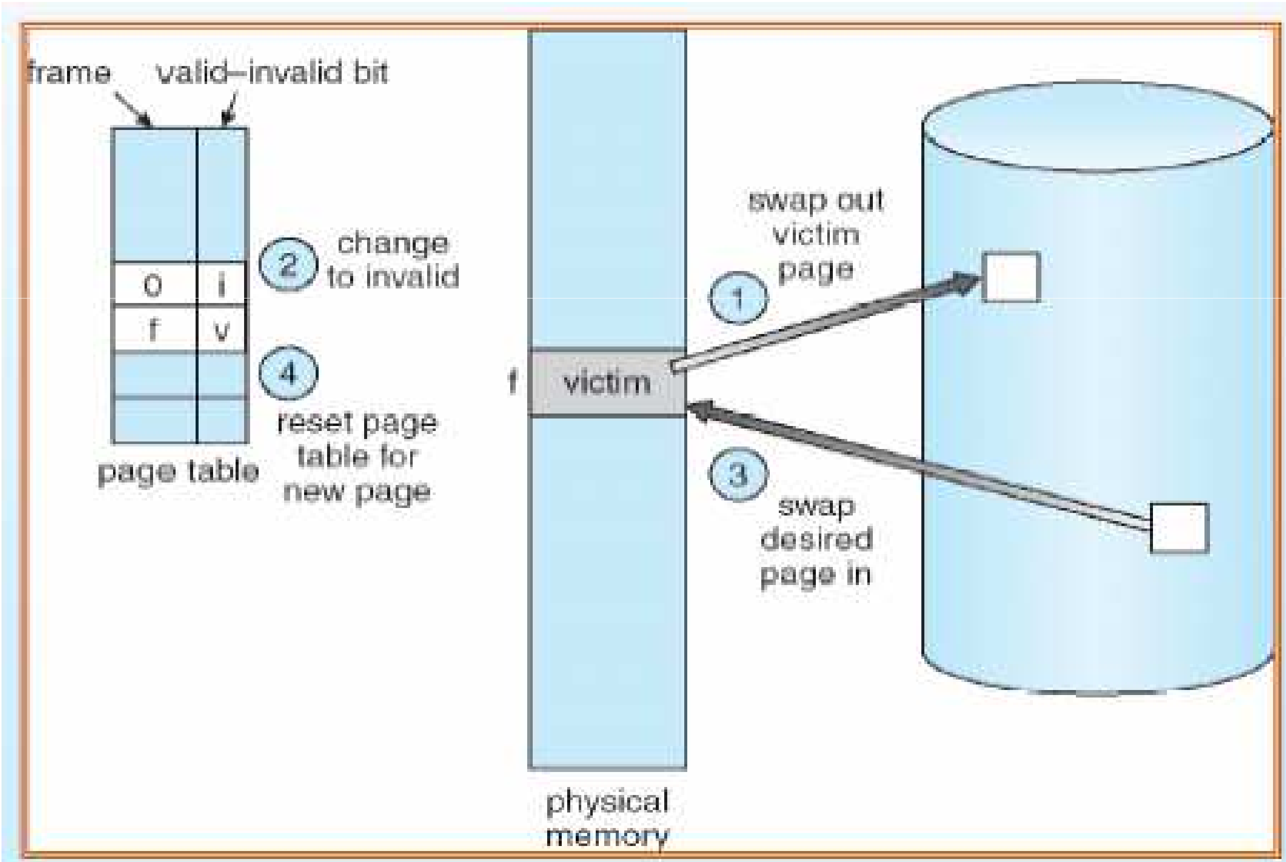
1. Find the location of the desired page on disk

2. Find a free frame:
   - If there is a free frame, use it
   - If there is no free frame, use a page replacement algorithm to select a **victim** frame

3. Bring the desired page into the (newly) free frame; update the page and frame tables

4. Restart the process

Page Replacement

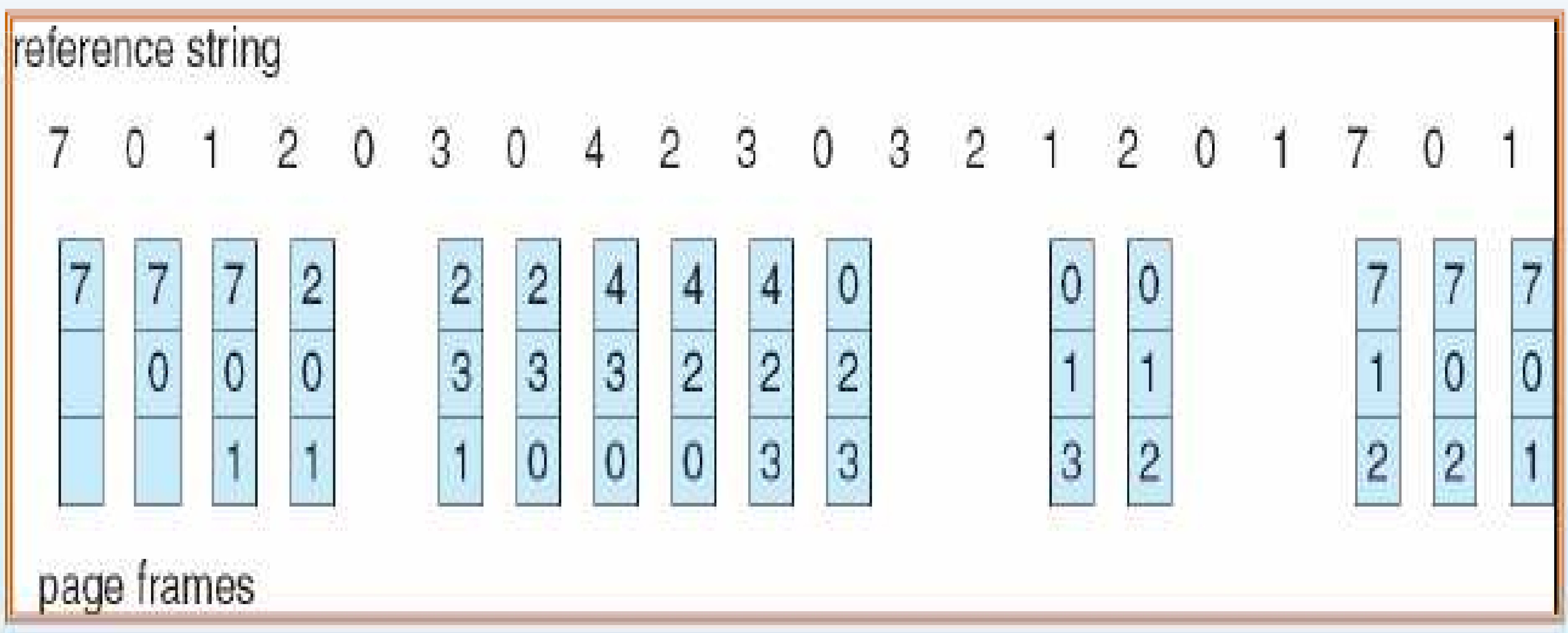


Page Replacement Algorithms

- Goal:
  - Want lowest page-fault rate
- Evaluate algorithm by
  - running it on a particular string of memory references (reference string), and
  - computing the number of page faults on that string

FIFO

- When a page must be replaced, the oldest page is chosen



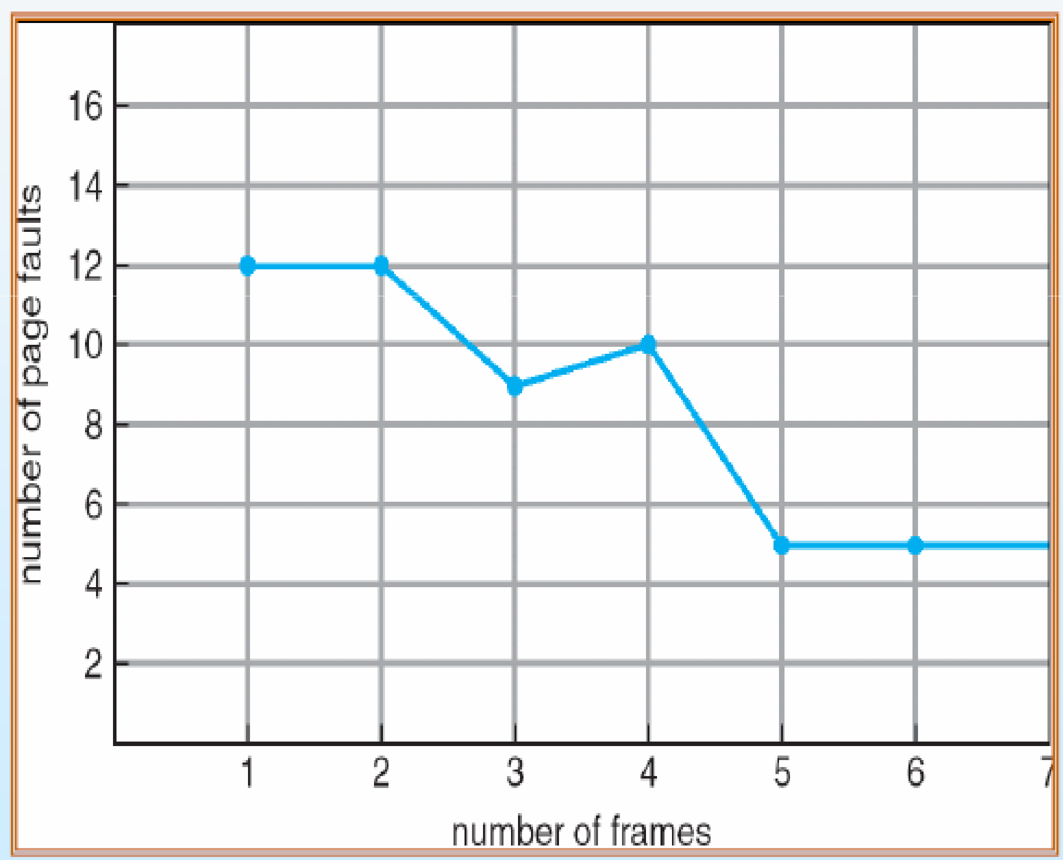
FIFO

- When a page must be replaced, the oldest page is chosen
- Consider the reference string is

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

  - 3 frame (9 page faults)
  - 4 frame (10 page faults)
- Notice that the number of faults for 4 frames is greater than the umber of faults for 3 frames!!
  - This unexpected result is known as **Belady's anomaly**

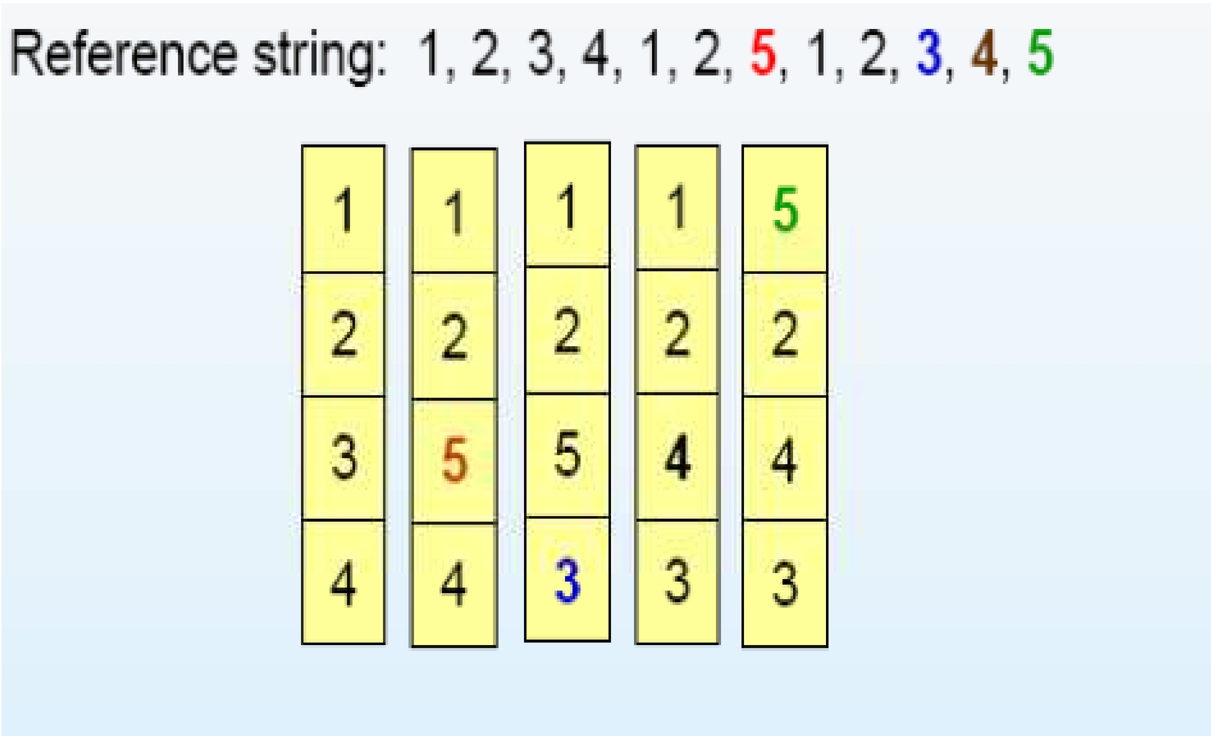
FIFO Illustrating Belady's Anomaly



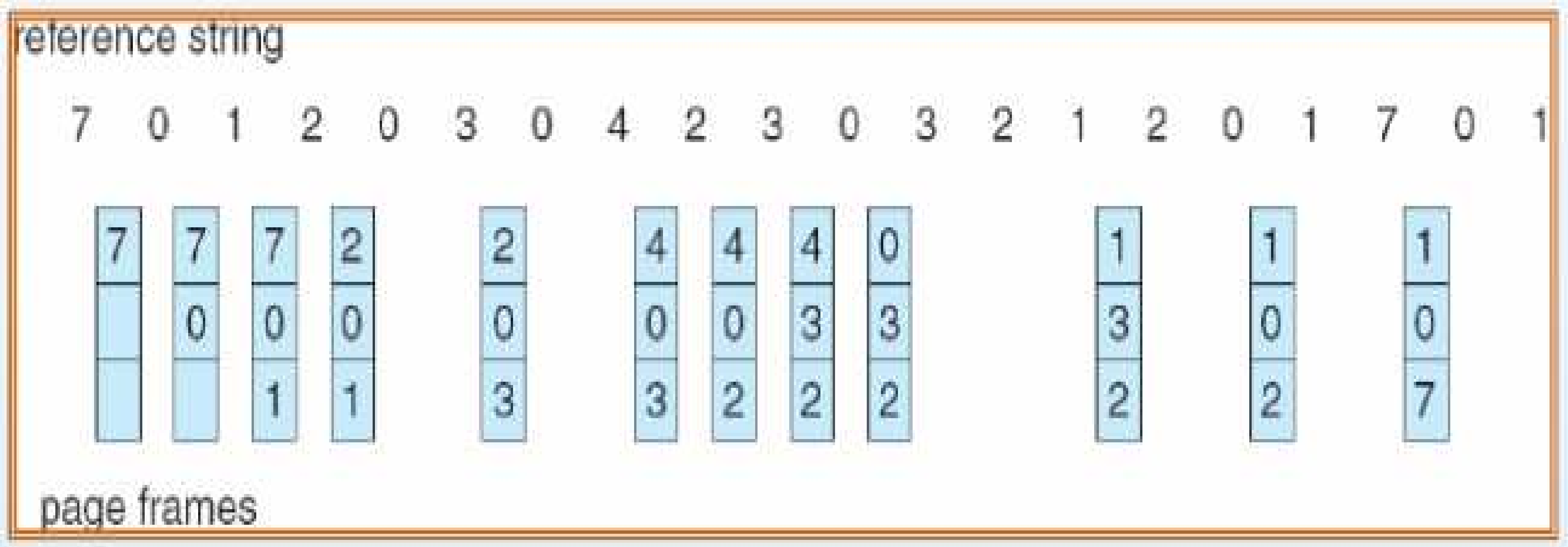
Least-recently-used (LRU) algorithm

- LRU replacement associates with each page the time of that page's last use
- When a page must be replaced, LRU chooses the page that has not been used for the longest period of time

Least-recently-used (LRU) algorithm



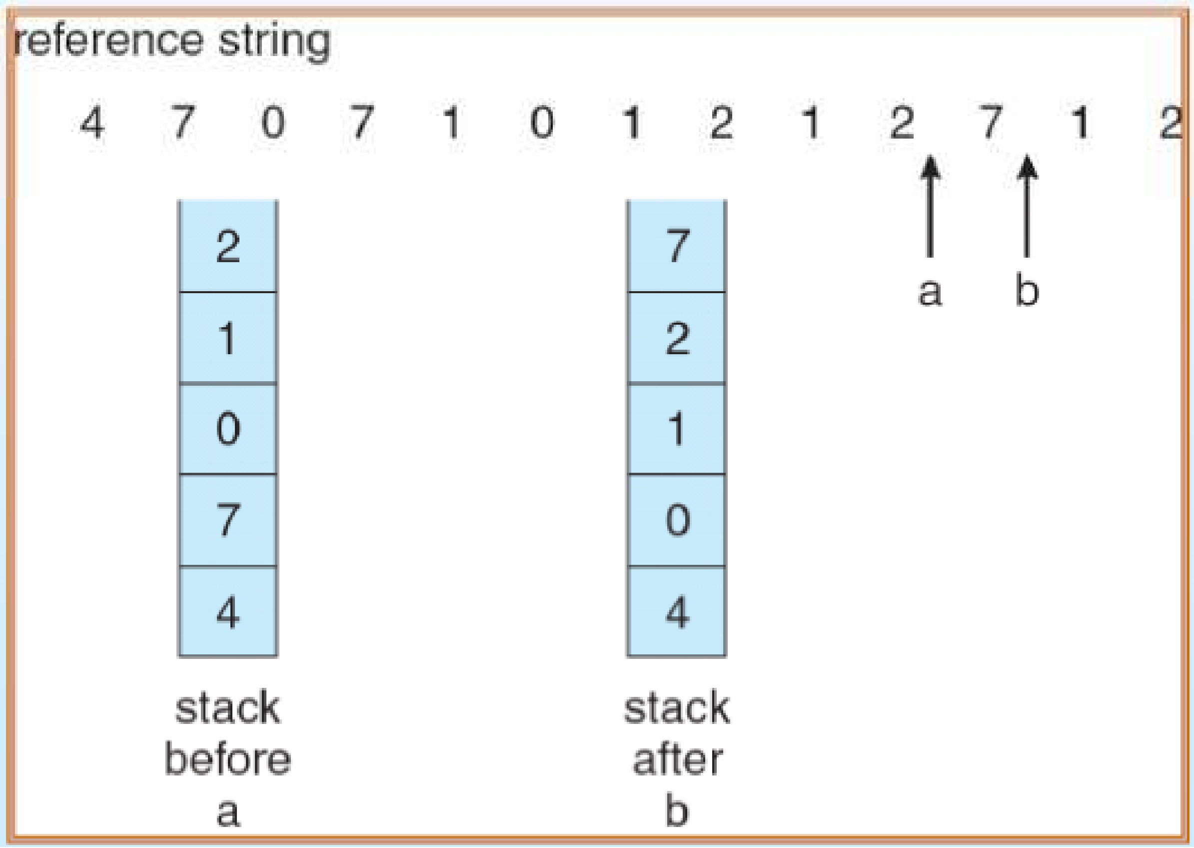
Least-recently-used (LRU) algorithm



Least-recently-used (LRU) algorithm

- The major problem is how to implement LRU replacement:
  - Counter: whenever a reference to a page is made, the content of the clock register are copied to the time-of-use field in the page table entry for the page.
    - We replace the page with the **smallest** time value
  - Stack: Whenever a page is referenced, it is removed from the stack and put on the top.
    - In this way, the most recently used page is always at the top of the stack

Stack Implementation



Optimal Page-Replacement Algorithm

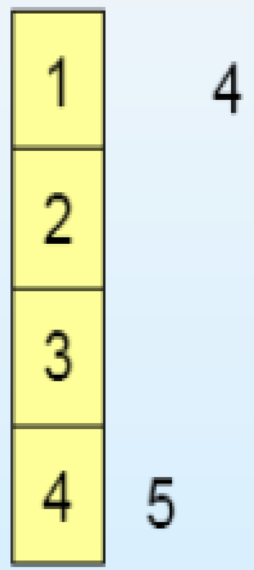
- Replace page that will not be used for longest period of time

- This is a design to guarantee the lowest page-fault rate for a fixed number of frames

Optimal Page-Replacement Algorithm

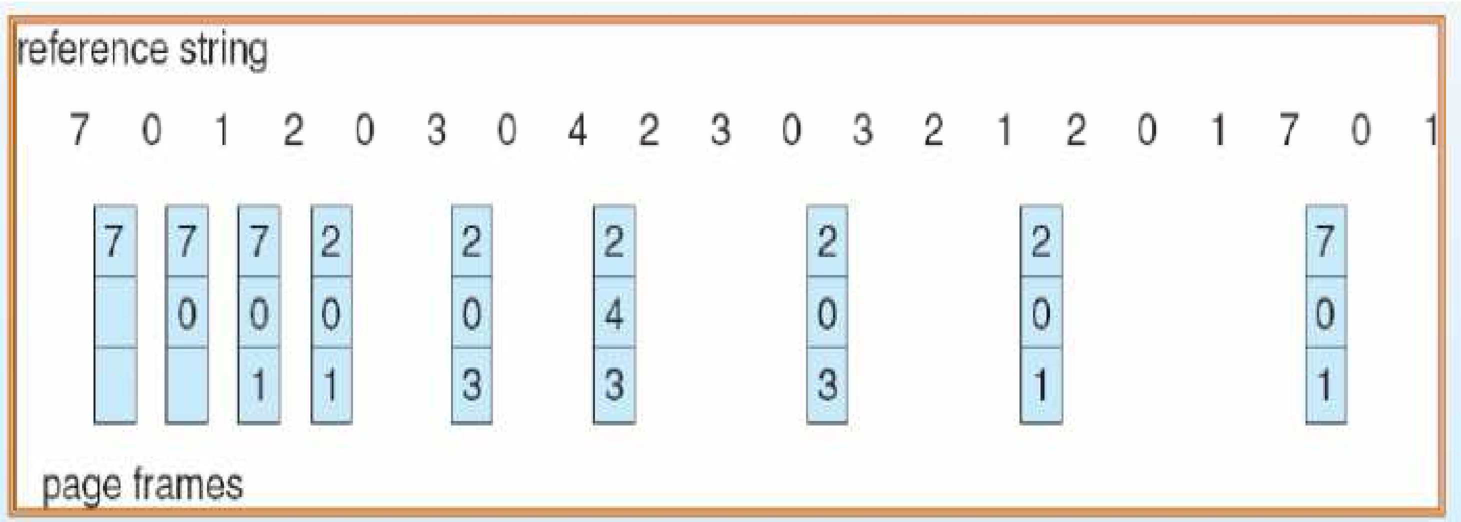
4 frames example

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



6 page faults

Optimal Page-Replacement Algorithm



Optimal Page-Replacement Algorithm

- Unfortunately, the optimal page-replacement is difficult to implement, because it requires future knowledge of the reference string

Second-Chance Algorithm

- Basically, it's a FIFO algorithm
  - If the page is referenced, we set the bit into 1
  - When a page has been selected, we inspect its reference bit
  - If the value is 0, we proceed to replace this page, otherwise, we give the page a second chance and move on to select the next FIFO page
  - When a page get a second chance, it's reference bit is cleared, and its arrival time is reset to the current time

Second-Chance Algorithm

- When a page get a second chance, it's reference bit is cleared, and its arrival time is reset to the current time
- If a page is used often enough to keep its reference bit set, it will never be replaced

Memory Management Hardware

- Basic components of a memory manager for a multi-programming environment
  - Facility for dynamic storage relocation
    - Logical to physical address mapping
  - Provision for sharing common programs
    - Several users will use same compiler program
  - Protection of information
    - Users should not be able to change or copy other user's information or to disrupt OS functions

Dynamic Storage Relocation

- Fixed page size in virtual memory is inconvenient
  - Affects program size and logical structure
- Divide programs and data into logical segments
  - Logically related instructions or data associated with a given name
  - May be generated by programmer or by OS
  - E.g., a subroutine, data array, symbol table, user's program, etc

Logical Address

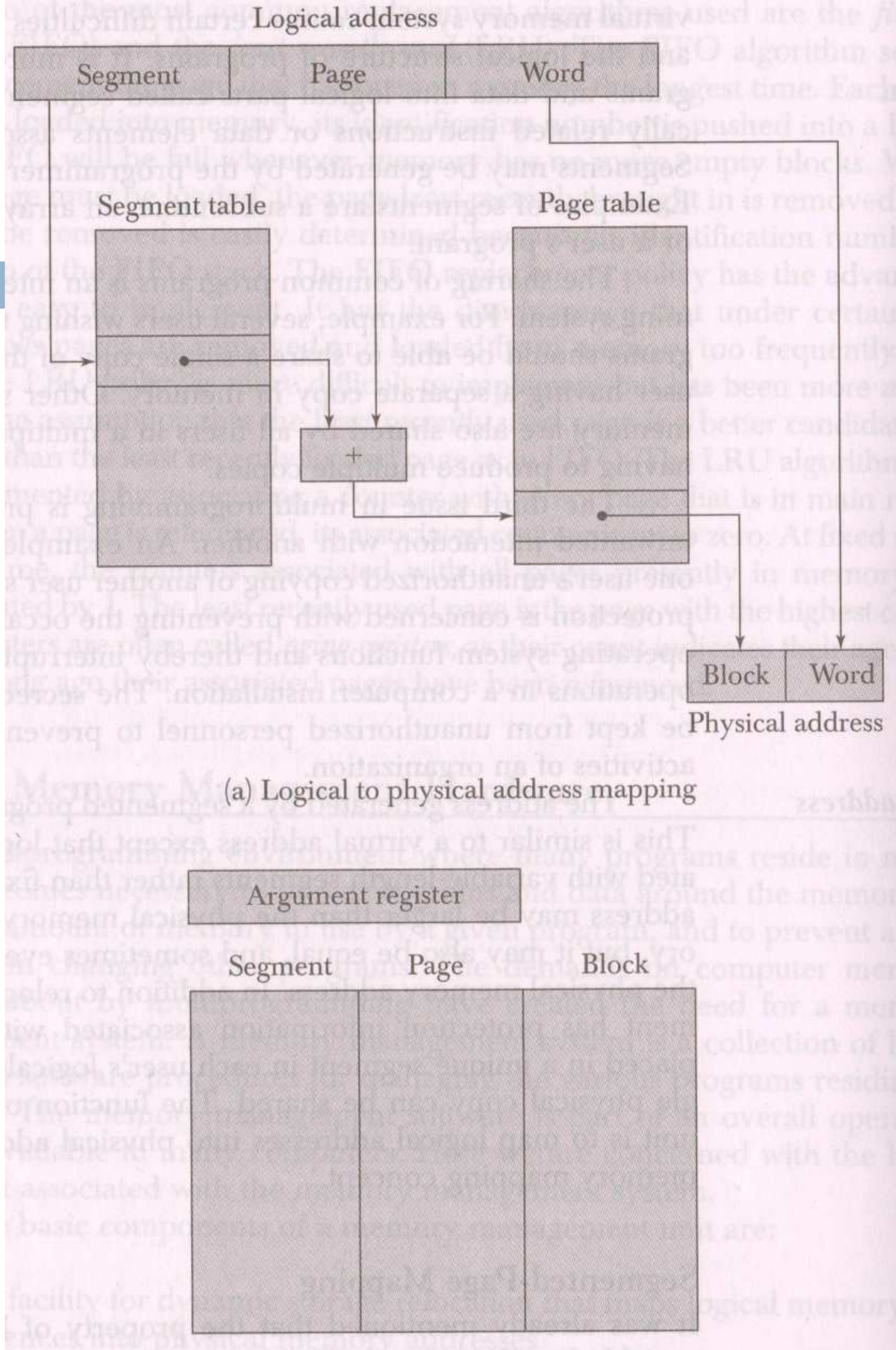
- Address generated by a segmented program is called a logical address
  - Logical address space is associated with variable length segments, not fixed-length pages
- Logical address may be larger, equal, or smaller than the physical memory address
- Each segment has (associated with it)
  - Relocation information
  - Protection information
- Shared programs are stored in a unique segment in each user's logical address space

Segmented-Page Mapping

- Each segment has variable length
  - Can grow and shrink
- How to manage segment length?
  - Associate equal-size pages to each segment
  - One segment number may be associated with anything from 1 to 2^k pages
    - So, the segment size varies based on the number of pages it actually has

Segmented-Page Mapping

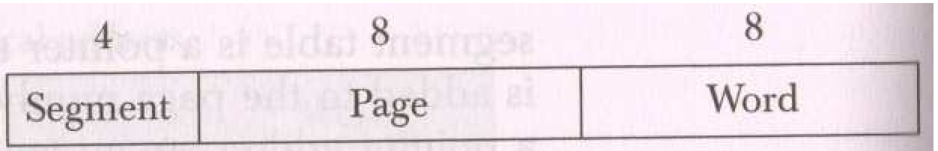
- Two mapping tables
  - Store in 2 separate small memories, or
  - In Main memory
  - Requires 3 memory accesses
- Use an associative memory
  - Called TLB



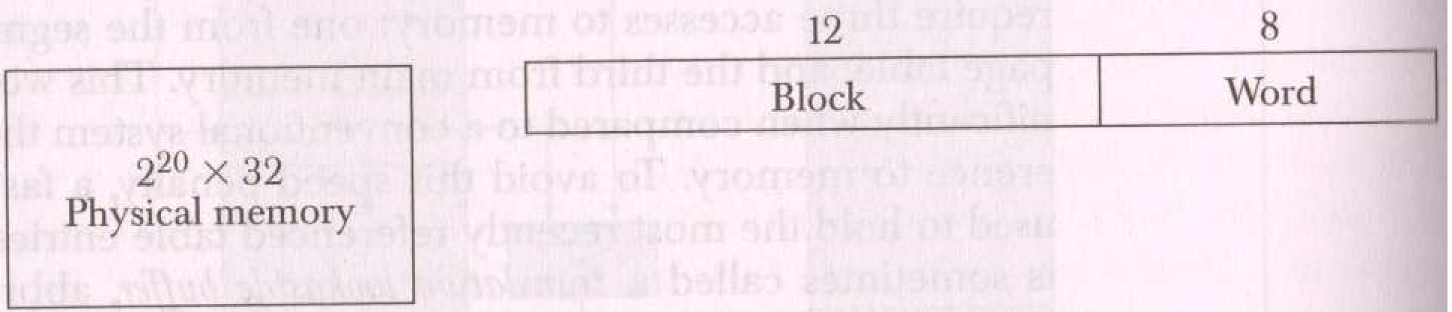
(a) Logical to physical address mapping

(b) Associative memory translation look-aside buffer (TLB)

Logical & Physical Address



(a) Logical address format: 16 segments of 256 pages each, each page has 256 words



(b) Physical address format: 4096 blocks of 256 words each, each word has 32 bits

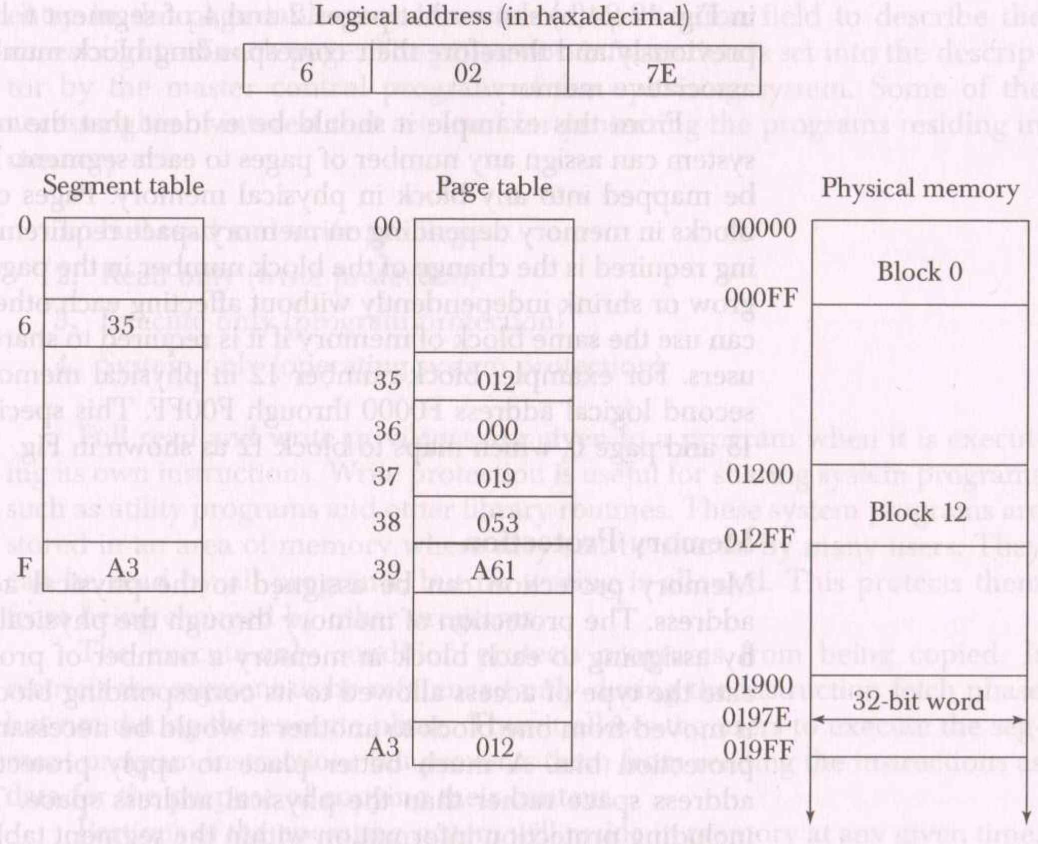
Logical & Physical Memory Address

Hexadecimal address	Page number
60000	Page 0
60100	Page 1
60200	Page 2
60300	Page 3
60400	Page 4
604FF	

(a) Logical address assignment

Segment	Page	Block
6	00	012
6	01	000
6	02	019
6	03	053
6	04	A61

(b) Segment-page versus memory block assignment



(a) Segment and page table mapping

Segment	Page	Block
6	02	019
6	04	A61

(b) Associative memory (TLB)

Memory Protection

- Full read and write privileges
- Read only (write protection)
- Execute only (program protection)
- System only (OS protection)