操作系统原理与设计

第9章 VM（虚存2）

陈香兰

中国科学技术大学计算机学院

2009年09月01日

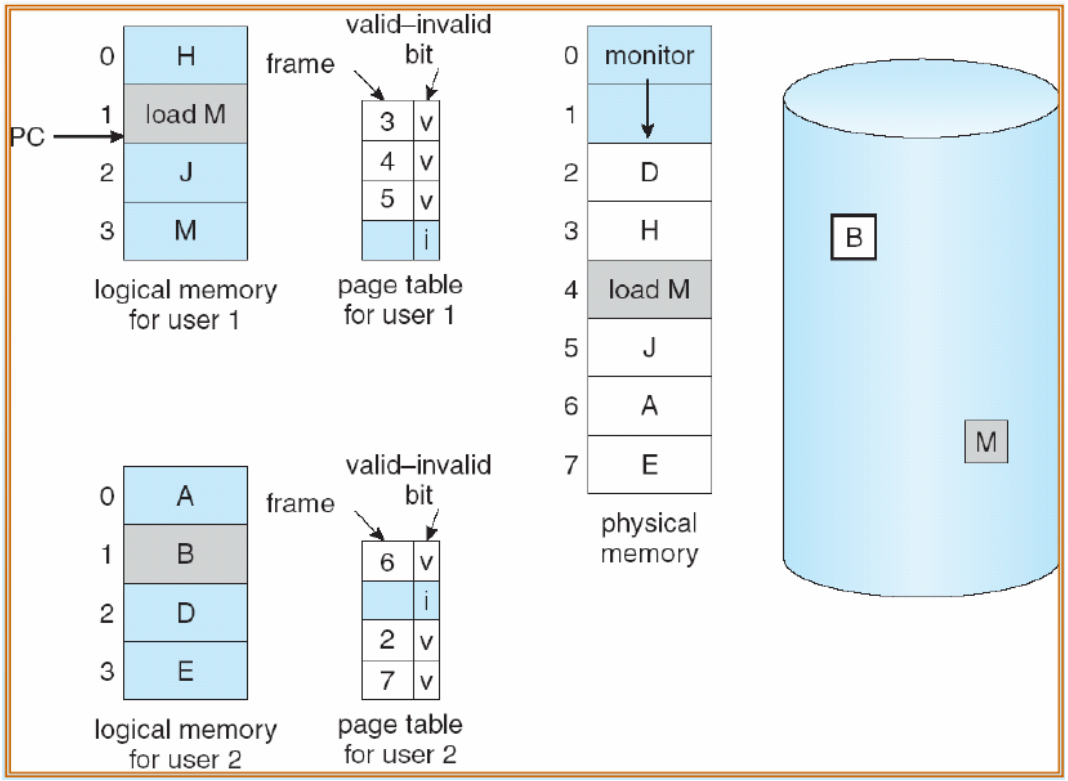
提纲

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

Page Replacement I

- Free page frame is managed by OS using free-frame-list
- What happens if there is no free frame?
  - ⇒Page replacement
- **over-allocation**
  - no free frames;
  - all memory is in use
- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement
- example:

Page Replacement II



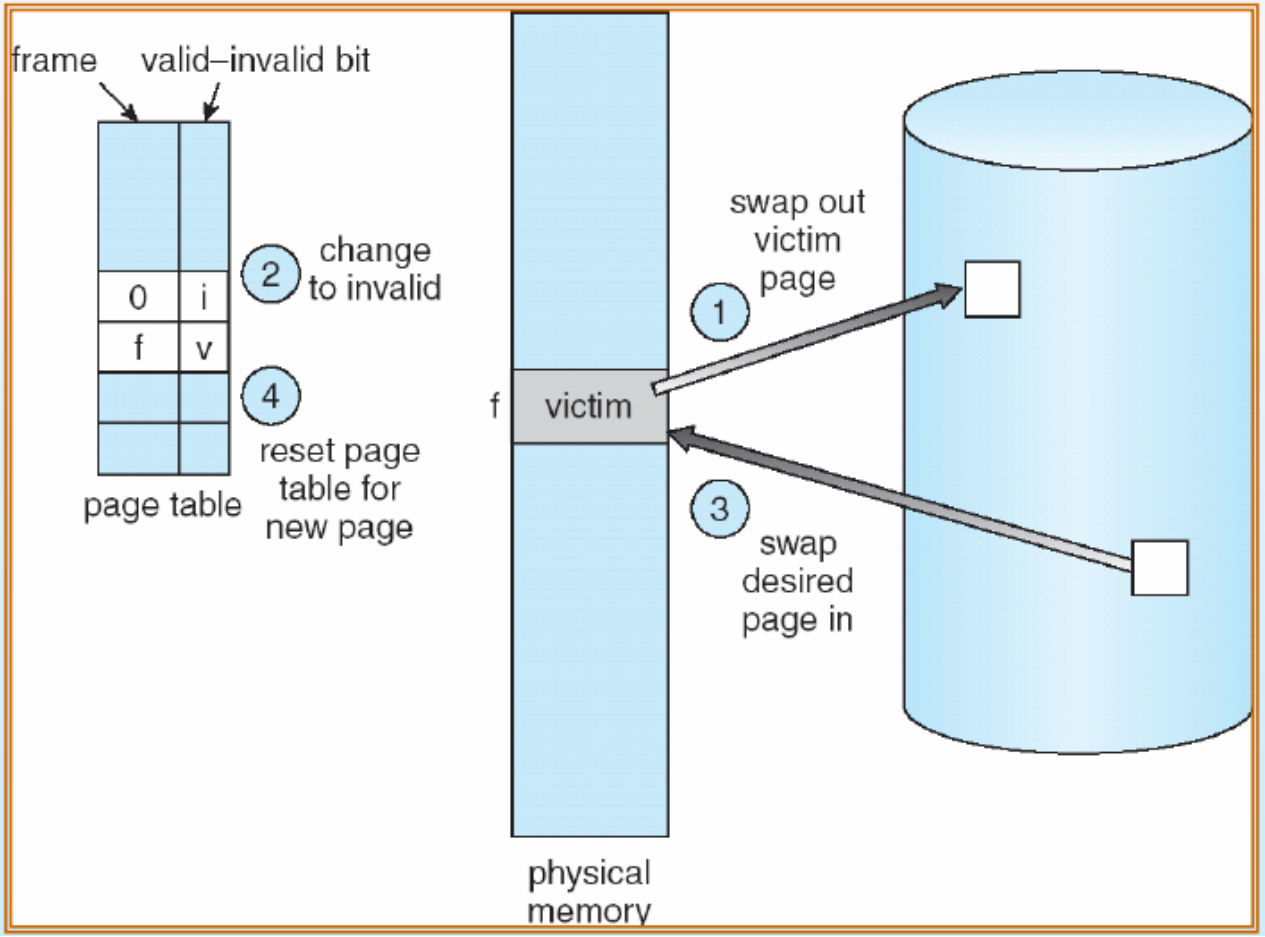
Outline

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

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



- NO MODIFY, NO WRITTEN (to disk/swap space)
  - Use **modify (dirty) bit** to reduce overhead of page transfers
    - only modified pages are written to disk
  - This technique also applies to read-only pages
    - for example, pages of binary code
- Page replacement **completes separation between logical memory and physical memory**
  - large virtual memory can be provided on a smaller physical memory
- demand paging, to lowest page-fault rate, two major problems
  - frame-allocation algorithms
  - page-replacement algorithms

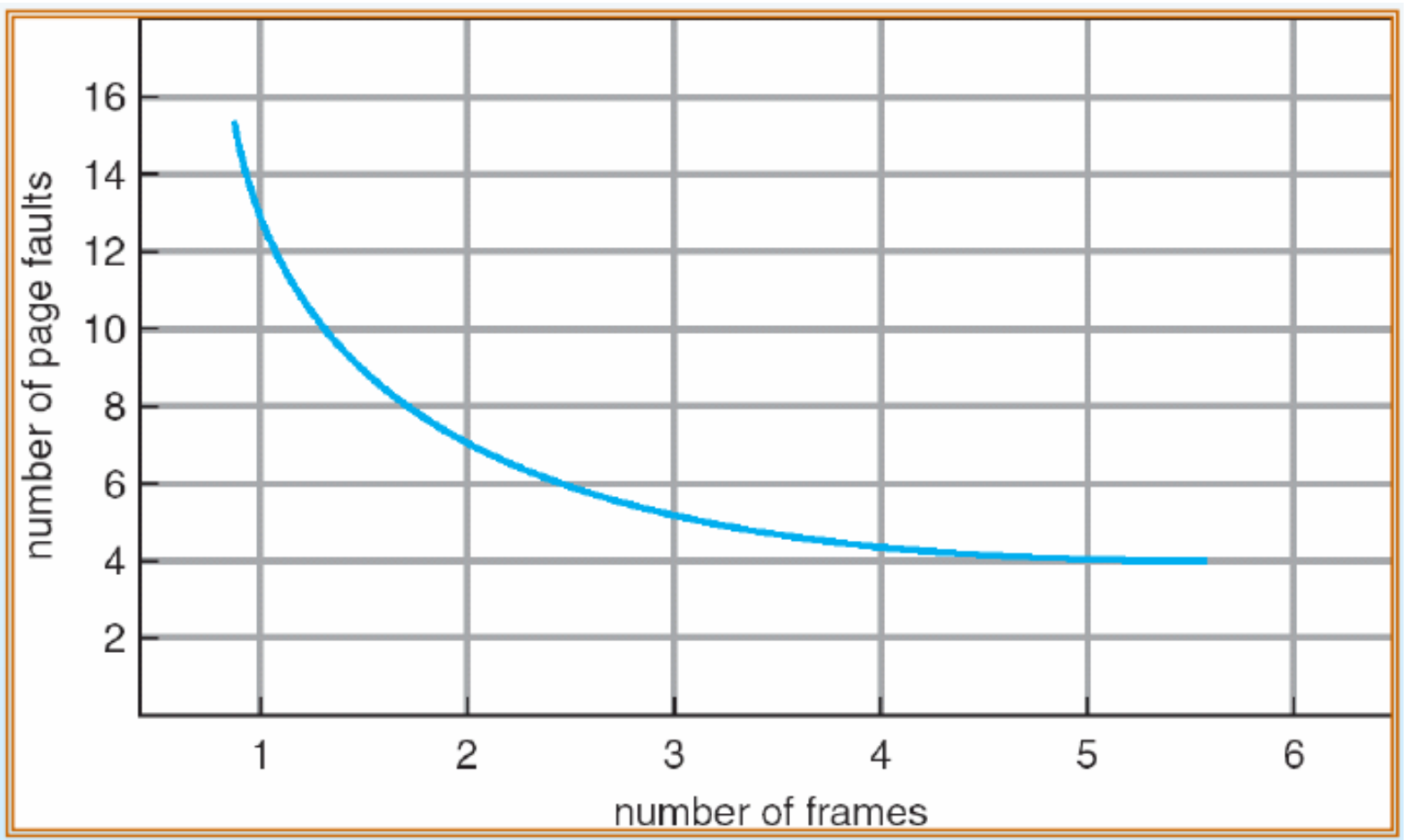
Page Replacement Algorithms I

- **GOAL**: to lowest page-fault rate
- Different algorithms are evaluated by running it on a particular string of memory references (**reference string**) and computing the number of page faults on that string
- A **reference string** is a sequence of addresses referenced by a program
- Example:
  - An address reference string:
    0100 **04**32 **01**01 **06**12 **01**02 0103 0104 0101 **06**11 **01**03 0104 0101 **06**10 **01**02 0103 0104 0101 **06**09 **01**02 0105

Page Replacement Algorithms II

- Assuming page size = 100 B, then its corresponding page reference string is:
  1 4 1 6 1 6 1 6 1 6 1
- how many page faults?
  - **determined by the number of page frames assigned to the process**
- if ≥ 3, then only 3 page faults
- if $= 1$, 11 pages faults

Graph of Page Faults Versus The Number of Frames



- In all our examples, the reference string is

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

- another one is

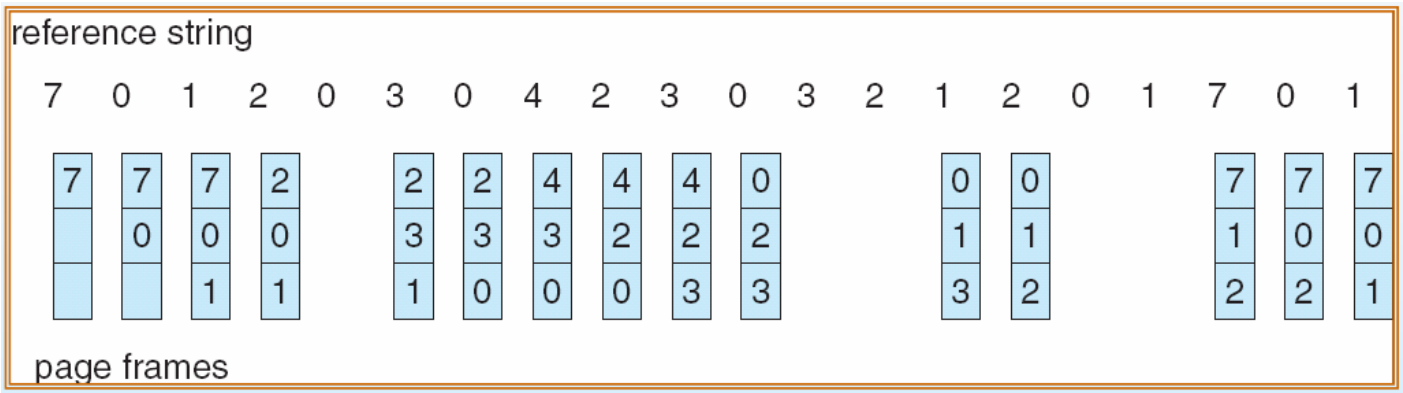
  7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

Outline

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

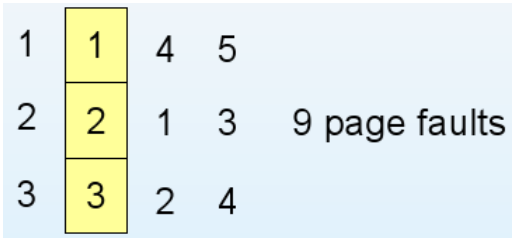
First-In-First-Out (FIFO) Algorithm I

1. example 1

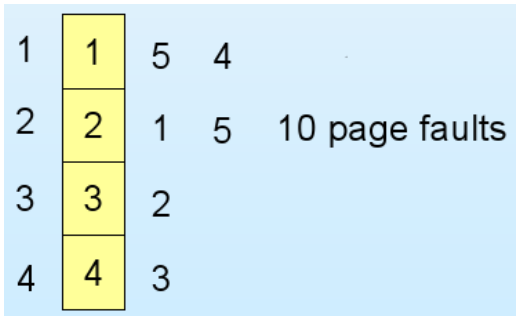


2. Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
   - if 3 frames

First-In-First-Out (FIFO) Algorithm II

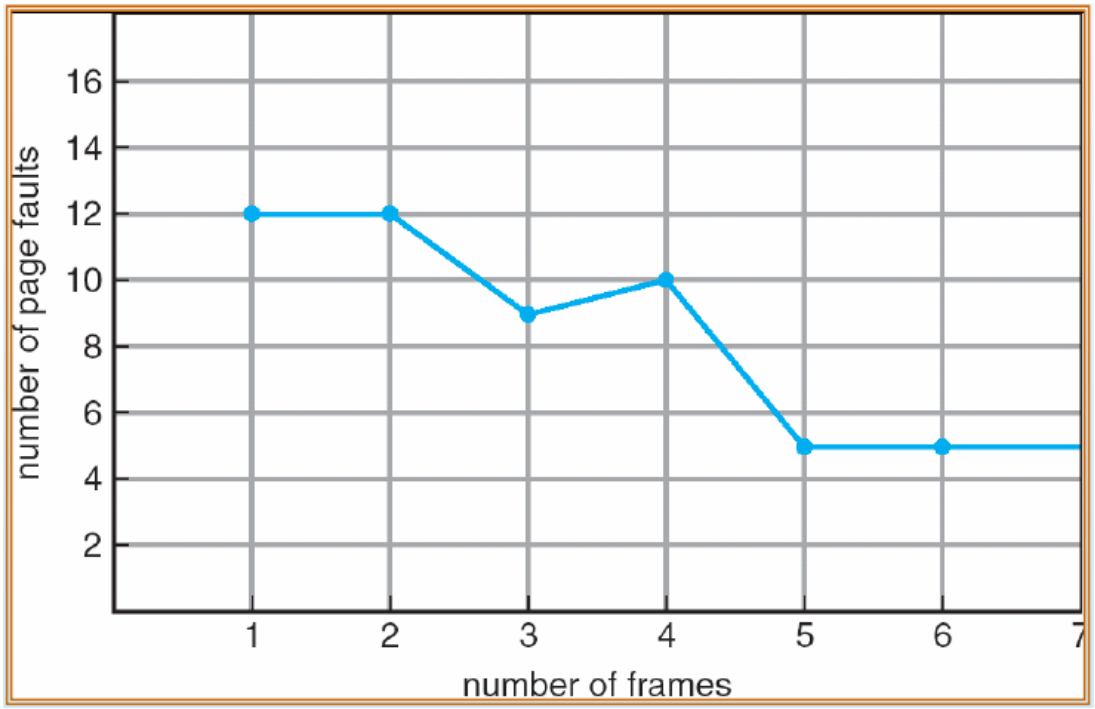


- if 4 frames



- Belady's (贝莱迪) Anomaly: more frames $\Rightarrow$ more page faults

FIFO Illustrating Belady's Anomaly



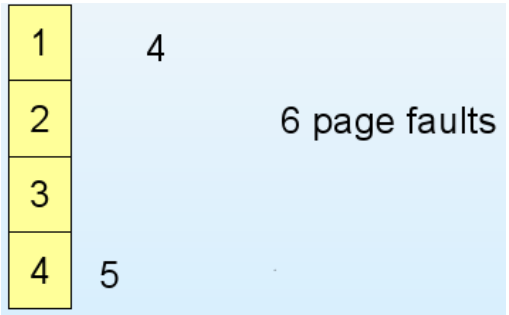
- **more memory, better performance?** MAY BE NOT!!
  - **Belady's anomaly**

Outline

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

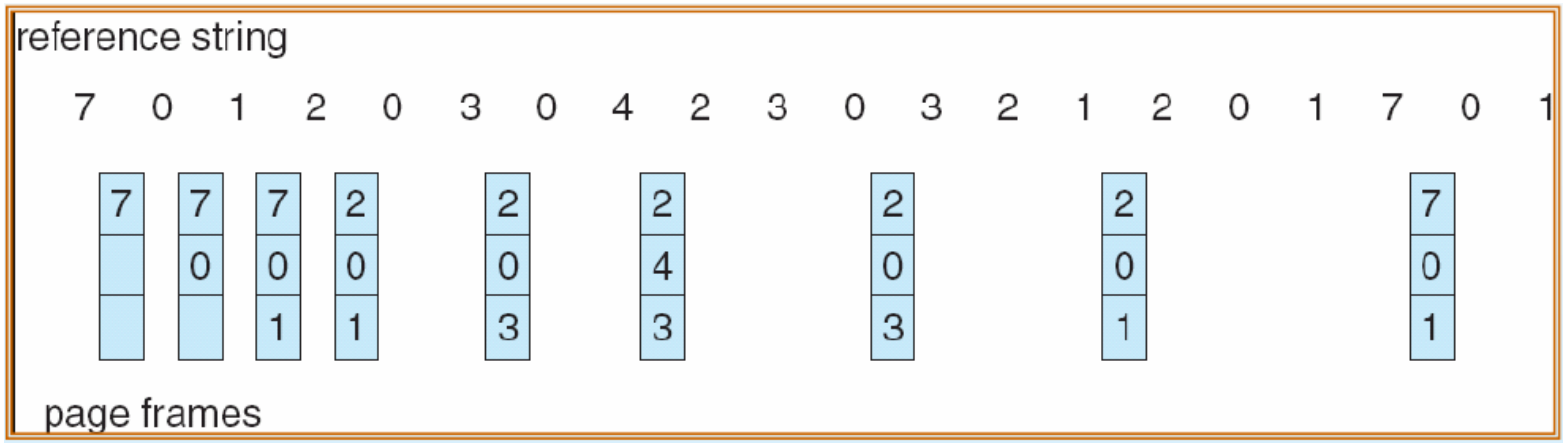
Optimal Algorithm

- Replace page that will not be used for longest period of time
- 4 frames example
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



- How do you know this?
- Used for measuring how well your algorithm performs

Optimal Page Replacement



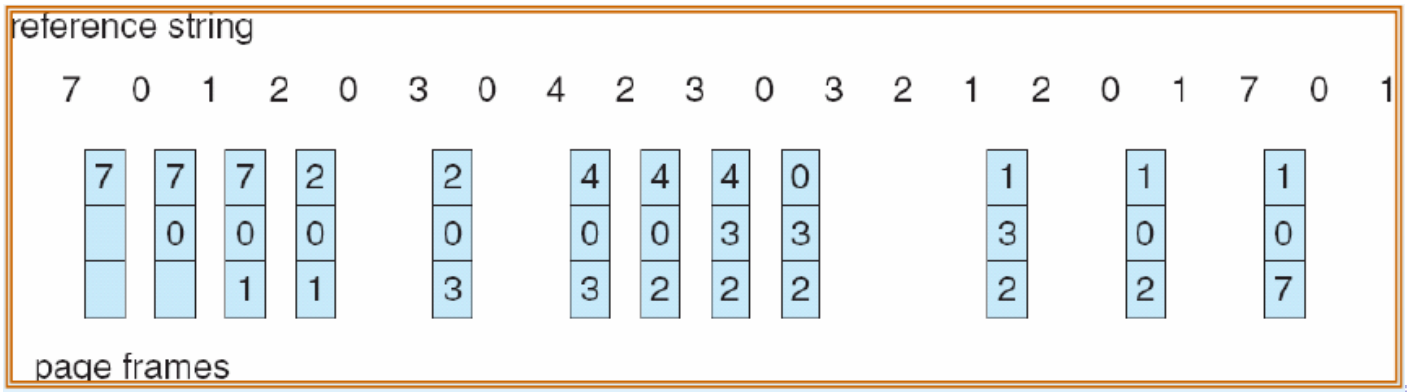
Outline

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

Least Recently Used (LRU) Algorithm I

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

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



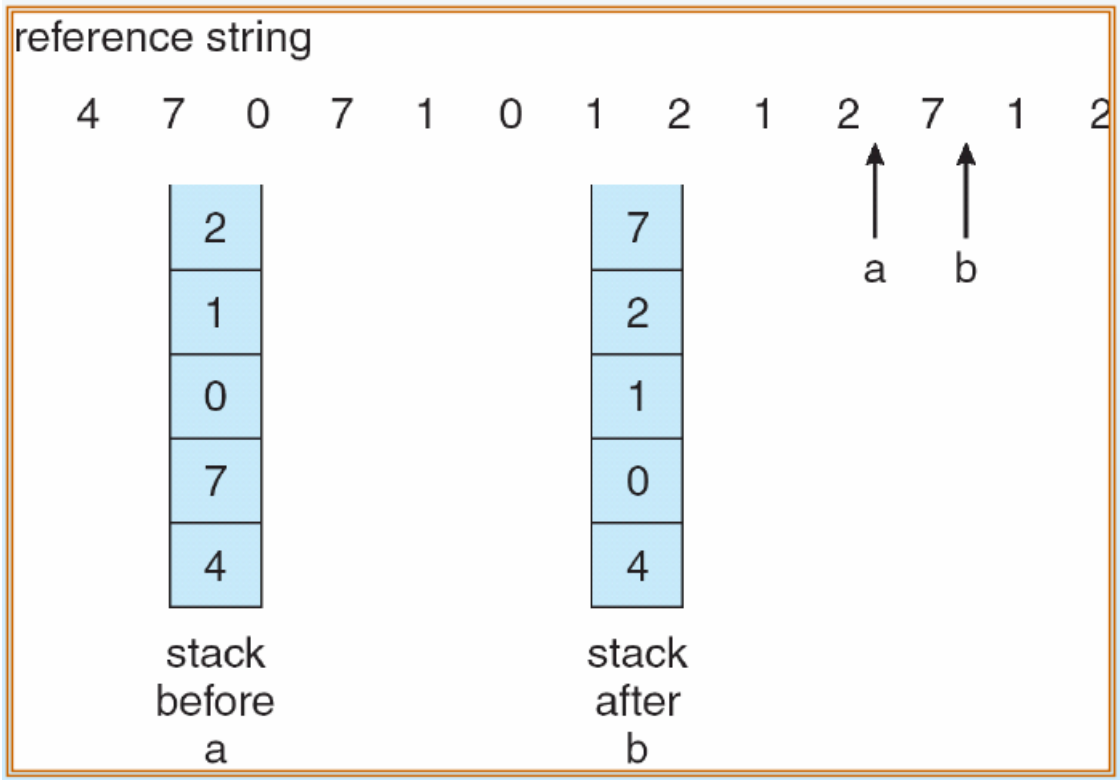
Least Recently Used (LRU) Algorithm II

HOW to implement LRU replacement?

1. **Counter** implementation
   - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter
   - When a page needs to be changed, look at the counters to determine which are to change
2. **Stack** implementation – keep a stack of page numbers in a double link form:
   - Page referenced:
     - move it to the top
     - requires 6 pointers to be changed
   - No search for replacement

Least Recently Used (LRU) Algorithm III

- stack example:



Outline

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

LRU Approximation Algorithms I

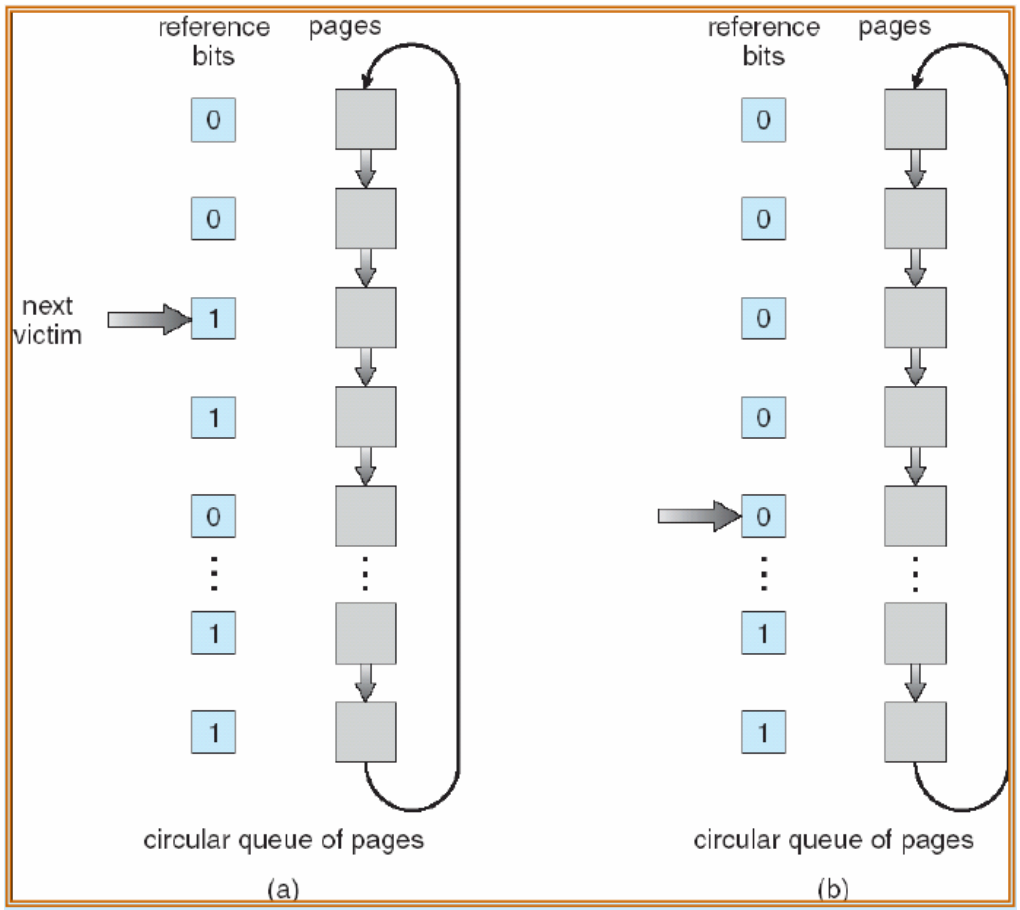
- Reference bit
  - With each page associate a bit, initially = 0
  - When page is referenced bit set to 1
  - Replace the one which is 0 (if one exists)
    - We do not know the order, however

1. Additinal-Reference-Bits Algorithm
   - reference bits + time ordering, for example
     - 8 bits
     - HW modifies the highest bit, only
     - periodically, right shift the 8 bits for each page
     - 00000000, ..., 01110111, ..., 11000100, ..., 11111111

LRU Approximation Algorithms II

2. Second chance Algorithm
   - Need 1 reference bit, modified FIFO algorithm
     - First, a page is selected by FIFO
     - then, the reference bit of the page is checked:
       0⇒replace it
       1⇒not replace it, get a second chance with reference bit:
       1→0, and time→current
   - Implementation: **Clock replacement**
     - clock order

LRU Approximation Algorithms III



LRU Approximation Algorithms IV

3. Enhanced Second-Chance Algothm
   - reference bit + modify bit
   - Four page classes （访问位，修改位）
     - (0, 0) – best page to replace
     - (0, 1) – not quite as good
     - (1, 0) – probably be used again soon
     - (1, 1) – probably be used again soon, and be dirty
   - Replace the first page encountered in the lowest nonempty class.
     1. Scan for (0, 0)
     2. Scan for (0, 1), & set reference bits to 0
     3. Loop back to step (a)

Outline

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

Counting Algorithms

- Keep a counter of the number of references that have been made to each page
- **LFU Algorithm**: replaces page with smallest count
- **MFU Algorithm**: based on the argument that the page with the smallest count was probably just brought in and has yet to be used

Outline

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

Page-Buffeing Algorithms I

- system commonly keep a pool of free frames
- when replacement occurs, **two frames** are involved
  - **a free frame** from the pool is allocated to the process
    - the desired page is read into the frame
  - **a viction frame** is chosen
    - written out later and the frame is added to the free pool
  - **NO NEED to write out before read in**
- An expansion
  - maintain a list of modified pages

Page-Buffeing Algorithms II

- when a paging device is idle, select a modified page, write it out, modify bit$\rightarrow$0

- Another modification
  - free frame with old page
  - the old page can be reused
    - less write out and less read in
  - VAX/VMS
  - some UNIX: + second chance
  - ...

Allocation of Frames

1. **minimum number of pages**
   - Each process needs minimum number of pages
   - Determined by ISA (Instruction-Set Architecture)
     - We must have enough frames to hold all the different pages that any single instruction can reference
   - Example: IBM 370
     6 pages to handle SS MOVE instruction:
     - instruction is 6 bytes, might span 2 pages
     - 2 pages to handle from
     - 2 pages to handle to
2. Two major **allocation schemes**
   - fixed allocation ; priority allocation
3. Two **replacement policy**
   - global vs. local

Fixed Allocation I

1. **Equal allocation** – For example, if there are 100 frames and 5 processes, give each process 20 frames.

$$
\begin{aligned}
\textit{frame number for any process} &= \frac{m}{n} \\
m &= \textit{total memory frames} \\
n &= \textit{number of processes}
\end{aligned}
$$

Fixed Allocation II

2. **Proportional allocation** – Allocate according to the size of process

$$
\begin{aligned}
s_i &= \text{size of process } p_i \\
S &= \Sigma s_i \\
m &= \text{total number of frames} \\
a_i &= \text{allocation for } p_i = \frac{s_i}{S} \times m
\end{aligned}
$$

- example:

$$
\begin{aligned}
m &= 64 \\
S_1 &= 10 \\
S_2 &= 127 \\
a_1 &= \frac{10}{137} \times 64 \approx 5 \\
a_2 &= \frac{127}{137} \times 64 \approx 59
\end{aligned}
$$

Priority Allocation

- Use a proportional allocation scheme **using priorities** rather than size
- If process P_i generates a page fault,
  - select for replacement one of its frames
  - select for replacement a frame from a process with lower priority number

Global vs. Local Allocation

- **Global replacement** – process selects a replacement frame from the set of all frames; one process can take a frame from another
  - **problem**: a process cannot control its own page-fault rate
- **Local replacement** – each process selects from only its own set of allocated frames
  - problem?

小结

1. Page Replacement
   - Basic Page Replacement
   - First-In-First-Out (FIFO) Algorithm
   - Optimal Algorithm
   - Least Recently Used (LRU) Algorithm
   - LRU Approximation Algorithms
   - Counting Algorithms
   - Page-Buffeing Algorithms
2. Allocation of Frames
3. 小结和作业

谢谢！