CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Implementing LRU

LRU replacement is effective and avoids Belady's anomaly

Must be able to be implemented efficiently to be useful

Two primary ways of implementing LRU:

- counter
- stack

LRU with counter

Maintain global count of memory accesses that acts as "time"

Each time page accessed (read or write), count written to corresponding row of page table

Can easily compare which of two processes was least-recently used by comparing corresponding times (counts) in page table

LRU with counter

However, finding least-recently-used process out of entire set requires searching through page table

Counter needs to be written to page table on *every* memory access

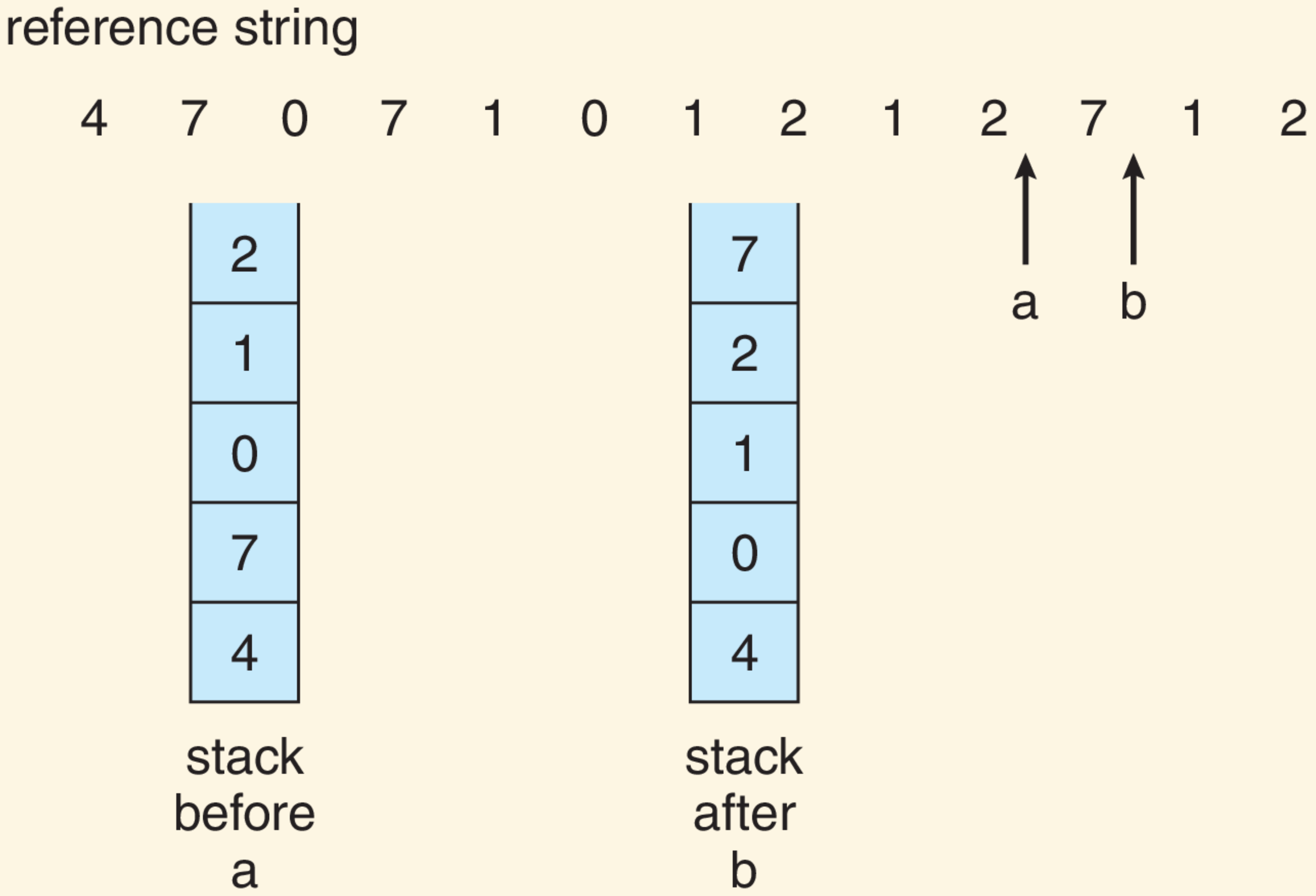
Clearly would require hardware support because we cannot affort to trap to OS on every memory access

LRU with stack

Keep all pages in a stack (not *the* stack)

Whenever page is accessed, move it out from middle and put on top of stack

Least-recently-used page is always on bottom of stack


reference string
4 7 0 7 1 0 1 2 1 2 7 1 2
a
b
2
1
0
7
4
7
2
1
0
4
stack before a
stack after b

LRU with stack

Since we are removing from middle, not really a stack from data structure point of view

Stored as doubly-linked list to facilitate moving from middle to top

Does not require search to find LRU because we keep pointers to head and tail of list

LRU with stack

Each update more expensive than updating in counter setup because pointers must be rearranged in linked list

Still requires hardware support for same reason as counter-based LRU -- cannot afford to trap to OS on every memory access

Hardware support for true LRU is uncommon

If hardware offers no support, typically something simpler such as FIFO must be used

Will discuss later how some systems offer partial support