

CS 370: OPERATING SYSTEMS [VIRTUAL MEMORY]

Demand Paging

When a process runs
 avoid the weight of all its pages
Guess what it needs
 bring in just those

The price you pay?
 page faults expensive they are
Too many of those?
 Trouble! for that's the process' cry for help

Shrudeep Pallickara
Computer Science
Colorado State University

COMPUTER SCIENCE DEPARTMENT



1

Frequently asked questions from the previous class survey

- Segments are paged?
- Hierarchical paging
 - Is the identification of the appropriate inner page table(s) deterministic?
 - How do TLBs work with it? Is it still only maintaining a mapping of virtual-page-to-physical-frame?
 - If you page-the-page-table, do all page tables need to be the same size?
 - Is there any scenario where hashed page tables would be better?
- If physical memory was infinite, would you need paging (or segmentation)?
- What are the consequences of having a bad hashing function?



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.2

2

Topics covered in this lecture

- Demand Paging
- Performance of Demand Paging
- Page Replacement
- Belady's anomaly
- Stack algorithms



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.3

3

DEMAND PAGING

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

4

Demand Paging: Basic concepts

- When a process is to be swapped in, **guess** which pages will be utilized by process
 - Before the process will be swapped out again
- **Avoid** reading unused pages
 - Better physical memory utilization
 - Reduced I/O
 - Lower swap times



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.5

5

Distinguishing between pages in memory and those on disk

- Valid-Invalid bits
 - Associated with entries in the page table
- **Valid**
 - Page is both legal and in memory
- **Invalid**
 - ① Page is *not in logical address space* of process
 - OR
 - ② Valid BUT currently *on disk*



COLORADO STATE UNIVERSITY

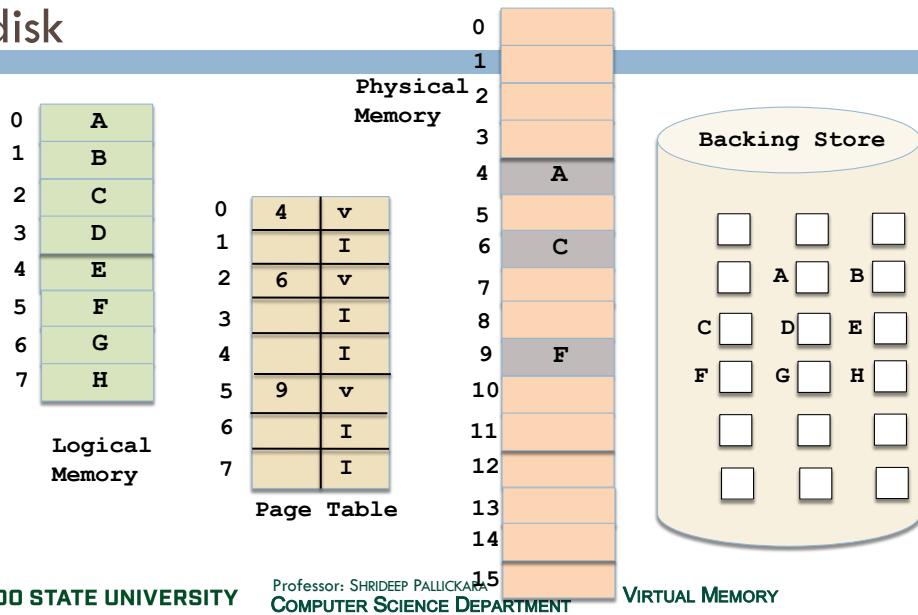
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.6

6

Distinguishing between pages in memory and those on disk



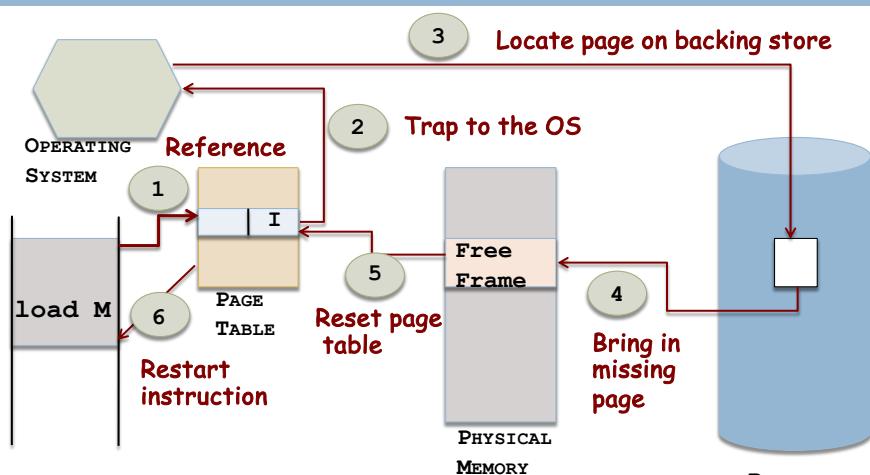
Handling Valid-invalid entries in the page table

- If process never attempts to access an invalid page?
 - No problems

- If process accesses page that is not memory resident?
 - **Page fault**



Handling page faults



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.9

9

Pure demand paging

- ❑ Never bring a page into memory unless it is required
- ❑ Execute process with no pages in memory
 - ❑ First instruction of process will fault for the page
- ❑ Page fault to load page into memory and execute



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.10

10

Potential problems with pure demand paging

- Multiple page faults per instruction execution
 - One fault for instruction
 - Many faults for data
- Multiple page faults per instruction are **rare**
 - **Locality of reference**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.11

11

Hardware requirements to support demand paging

- Page Table
- Secondary space
 - Section of disk known as **swap space** is used



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.12

12

Restarting instructions after a page fault

- Page faults occur at **memory reference**
- Use PCB to save state of the interrupted process
- Restart process in **exactly** the same place
 - Desired page is now in memory and accessible



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.13

13

Restarting processes after a page fault has been serviced

- If fault occurred during an instruction fetch
 - During restart, refetch the instruction
- If fault occurred while fetching operands
 - ① Fetch and decode instruction
 - ② Fetch the operand



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.14

14

Worst case example

- Add operands **A** and **B**
 - Place sum in **C**
- If we fault while storing **C**
 - Service page fault
 - Update page table
 - Restart instruction
 - Decode, fetch operands, and perform addition



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.15

15

Problems when operations modify several different memory locations

- E.g., Move a block from one memory location to another
- {C1} Either block straddles page-boundary
- {C2} Page fault occurs
- Move might be **partially** done
 - Uh-oh ...



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.16

16

Approaches to fault-proofing block transfers

- ① Compute and access both **ends of the block**
 - If a page fault were to happen it will at this point
 - Nothing has been partially modified
 - After fault servicing, block transfer completes
- ② Use temporary registers
 - Track overwritten values



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.17

17

Can on-demand paging be applied anywhere without modifications?

- Paging is between CPU and physical memory
 - **Transparent** to user process
- Non-demand paging can be applied to **any system**
- **Not so** for demand paging
 - Fault processing of special instructions non-trivial



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.18

18



PERFORMANCE OF DEMAND PAGING

19

Effective access times

- **Without** page faults, effective access times are equal to memory access times
 - 200 nanoseconds approximately
- **With** page faults
 - Account for fault servicing with disk I/O



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.20

20

Calculating the effective access times with demand paging

p : probability of a page fault

ma : memory access time

Effective access time =

$$(1-p) \times ma + p \times \text{page-fault-time}$$



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.21

21

Components of page-fault servicing

Service
interrupt

1~100 μ S

Read in
the page

Latency : 3 mS
Seek : 5 mS

Restart
process

1~100 μ S



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.22

22

Effective access times

- Effective access time =

$$(1-p) \times ma + p \times \text{page-fault-time}$$

$$= (1-p) \times 200\text{ns} + p \times (8\text{mS})$$

$$= (1-p) \times 200 + p \times (8,000,000)$$

$$= 200 + 7,999,800 \times p$$

Effective access time directly
proportional to page-fault rate



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.23

23

If performance degradation is to be less than 10%

$$220 > 200 + 7,999,800 \times p$$

$$20 > 7,999,800 \times p$$

$$p < 0.0000025$$

Fewer than 1 memory access out
of 399,990 can page-fault



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.24

24



OTHER ISSUES IN DEMAND PAGING

COMPUTER SCIENCE DEPARTMENT



25

Allocation of physical memory to I/O and programs
is a challenge

- Memory used for holding **program** pages
- **I/O buffers** also consume a big chunk of memory
- Solutions:
 - Fixed percentage set aside for I/O buffers
 - Processes and the I/O subsystem compete



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.26

26

Demand paging and the limits of logical memory

- Without demand paging
 - All pages of process **must be** in physical memory
 - Logical memory **limited** to size of physical memory
- With demand paging
 - All pages of process **need not be** in physical memory
 - Size of logical address space is **no longer constrained** by physical memory



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.27

27

Demand paging is the OS' attempt to improve CPU utilization and system throughput

- Load pages into memory when they are **referenced**
 - Increases degree of **multiprogramming**
- Example
 - 40 pages of physical memory
 - 6 processes, each of which is 10 pages in size
 - Each process only needs 5 pages as of now
 - Run 6 processes with 10 pages to spare



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.28

28

Increasing the degree of multiprogramming can be tricky

- Essentially, we are **over-allocating** physical memory
- Example
 - Physical memory = 40 pages
 - 6 processes each of which is of size 10 pages
 - But are using 5 pages each as of now
 - What happens if each process needs all 10 pages?
 - 60 physical frames needed



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.29

29

Coping with over-allocation of memory

- **Terminate** a user process
 - But paging should be transparent to the user
- **Swap out** a process
 - Reduces the degree of multiprogramming
- **Page replacement**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.30

30

The two core problems in demand paging

- **Frame allocation**

- How many frames to allocate to a process

- **Page replacement**

- Select the frame(s) for replacement

- **Caveat:**

- Disk I/O is expensive, so inefficient solutions can weigh things down



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.31

31



32

Page replacement

- If no frame is free?
 - Find one that is not currently being used
 - Use it



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.33

33

Freeing a physical memory frame

- Write frame contents to swap space
- Change page table of process
 - To reflect that page is no longer in memory
- Freed frame can now hold some other page



COLORADO STATE UNIVERSITY

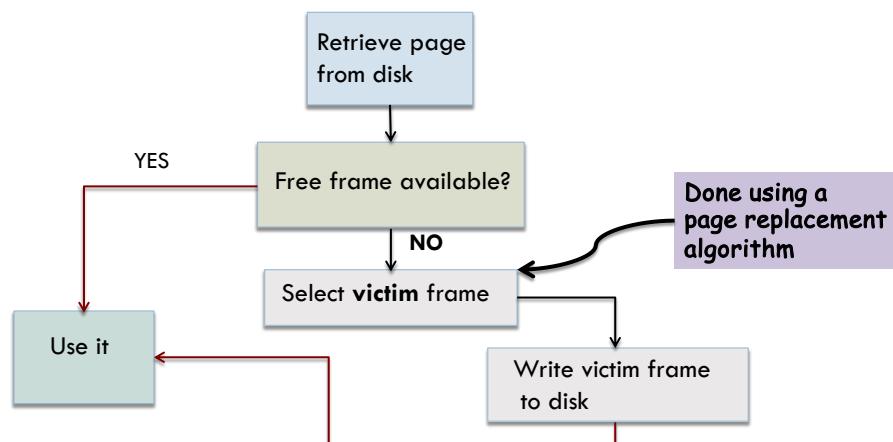
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.34

34

Servicing a page fault



COLORADO STATE UNIVERSITY

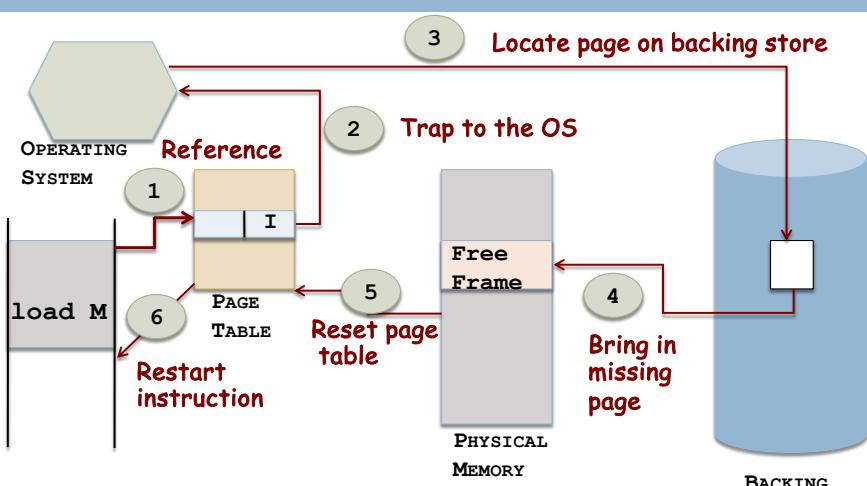
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.35

35

Page replacement is central to demand paging



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.36

36

Overheads for page replacement

- If no frames are free: **2** page transfers needed
 - Victim page out
 - New page in
- No free frames?
 - Doubles page-fault service time
 - Increases effective access time



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.37

37

Using the modify bit to reduce page replacement overheads

- Each page/frame has a **modify** bit
 - Set by hardware when page is written into
 - Indicates if page was modified
 - Since the last time it was read from disk
- During page replacement
 - If victim page not modified, no need to write it to disk
 - Reduces I/O time by **one-half**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.38

38

PAGE REPLACEMENT ALGORITHMS

COMPUTER SCIENCE DEPARTMENT



39

Page replacement algorithms:

- What are we looking for?
 - **Low page-fault rates**
- How do we evaluate them?
 - Run algorithm on a string of memory references
 - **Reference string**
 - Compute number of page faults



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.40

40

The reference string: Snapshot memory references

- We track page numbers
 - Not the entire address
- If we have a reference to a memory-resident page p
 - Any references to p that follow will not page fault
 - Page is already in memory



COLORADO STATE UNIVERSITY

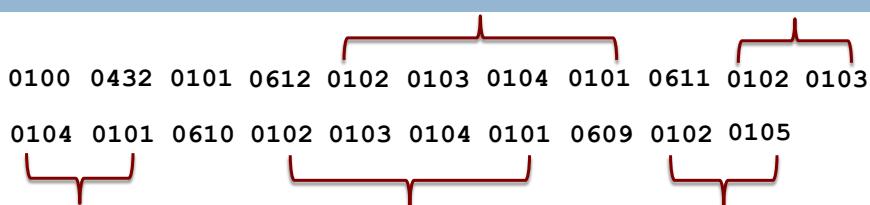
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.41

41

The reference string: Example Page size = 100 bytes



1, 4, 1, 6, 1, 6, 1, 6, 1, 6, 1



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.42

42

Factors involved in determining page faults

- Reference string of executing process
- Page replacement algorithm
- Number of physical memory frames available
- Intuitively:
 - Page faults reduce as the number of page frames increase



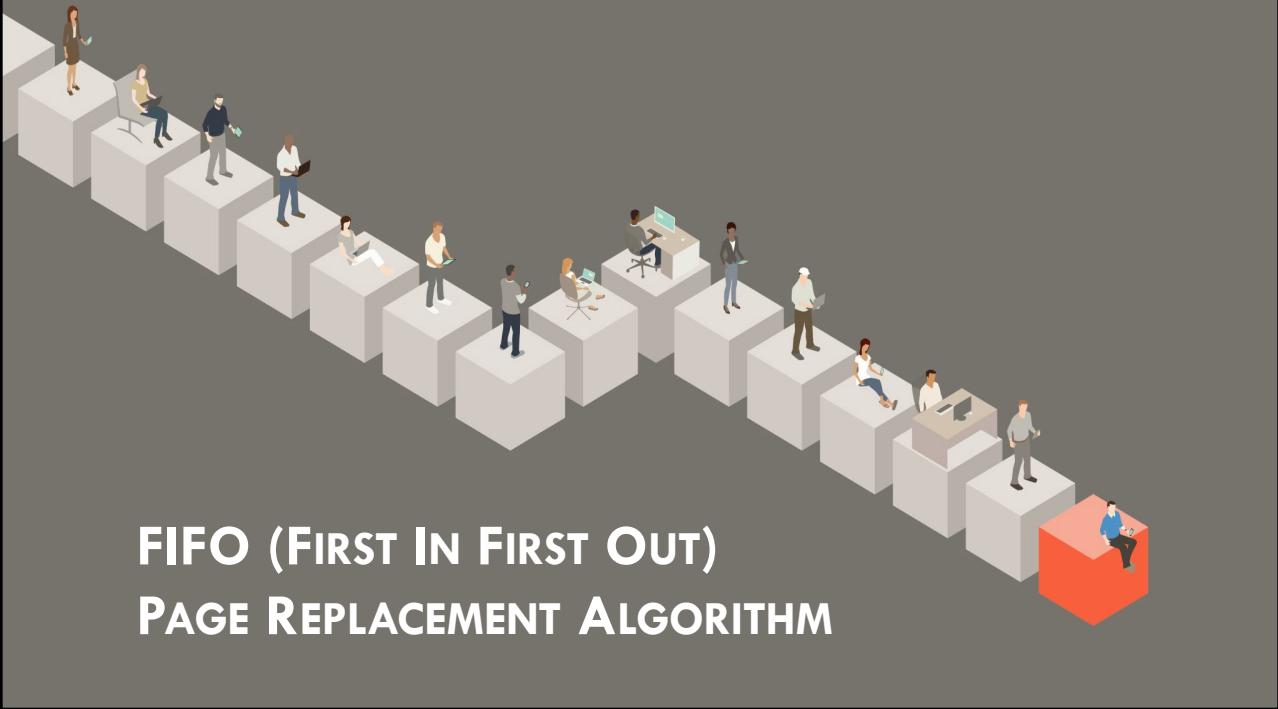
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.43

43



**FIFO (FIRST IN FIRST OUT)
PAGE REPLACEMENT ALGORITHM**

44

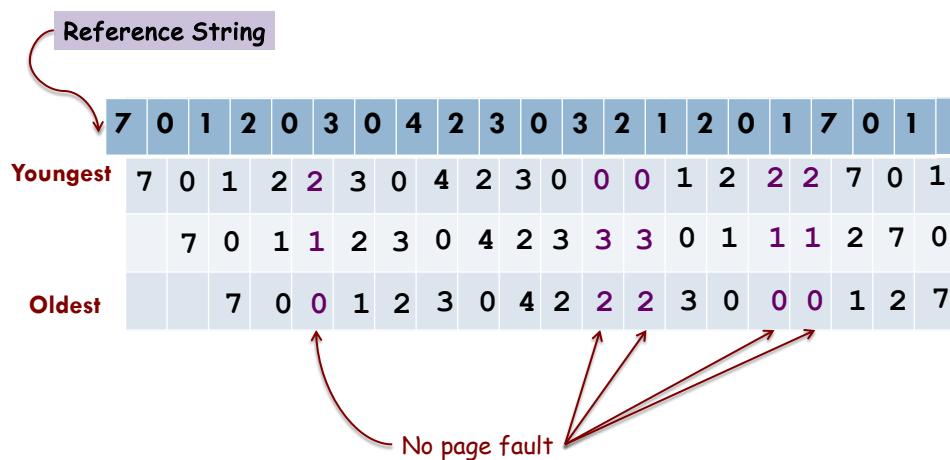
FIFO page replacement algorithm: Out with the old; in with the new

- When a page must be replaced
 - Replace the **oldest** one
- OS maintains list of all pages currently in memory
 - Page at head of the list: Oldest one
 - Page at the tail: Recent arrival
- During a page fault
 - Page at the head is removed
 - New page added to the tail



45

FIFO example: 3 memory frames



46

With your feet on the air and your head on the ground
Try this trick and spin it, yeah

Where is My Mind, Charles Thompson/Black Francis, Pixies

BELADY'S ANOMALY

COMPUTER SCIENCE DEPARTMENT



47

Intuitively the greater the number of memory frames,
the lower the faults

- Surprisingly, this is **not always** the case
- In 1969 Belady, Nelson and Shedler discovered counter example* in FIFO
 - FIFO caused more faults with 4 frames than 3
- This strange situation is now called **Belady's anomaly**

An anomaly in space-time characteristics of certain programs running in a paging machine.
Belady, Nelson and Shedler. Communications of the ACM. Vol. 12 (6) pp 349-353. 1969.



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.48

48

Belady's anomaly: FIFO

Same reference string, different frames

0 1 2 3 0 1 4 0 1 2 3 4													
Youngest													Numbers in this color: No page fault
Youngest													9 page faults with 3 frames
Oldest													9 page faults with 3 frames
Youngest													10 page faults with 4 frames
Oldest													10 page faults with 4 frames



COLORADO STATE UNIVERSITY

COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.49

49

Belady's anomaly

- Led to a whole theory on paging algorithms and properties
- **Stack algorithms**



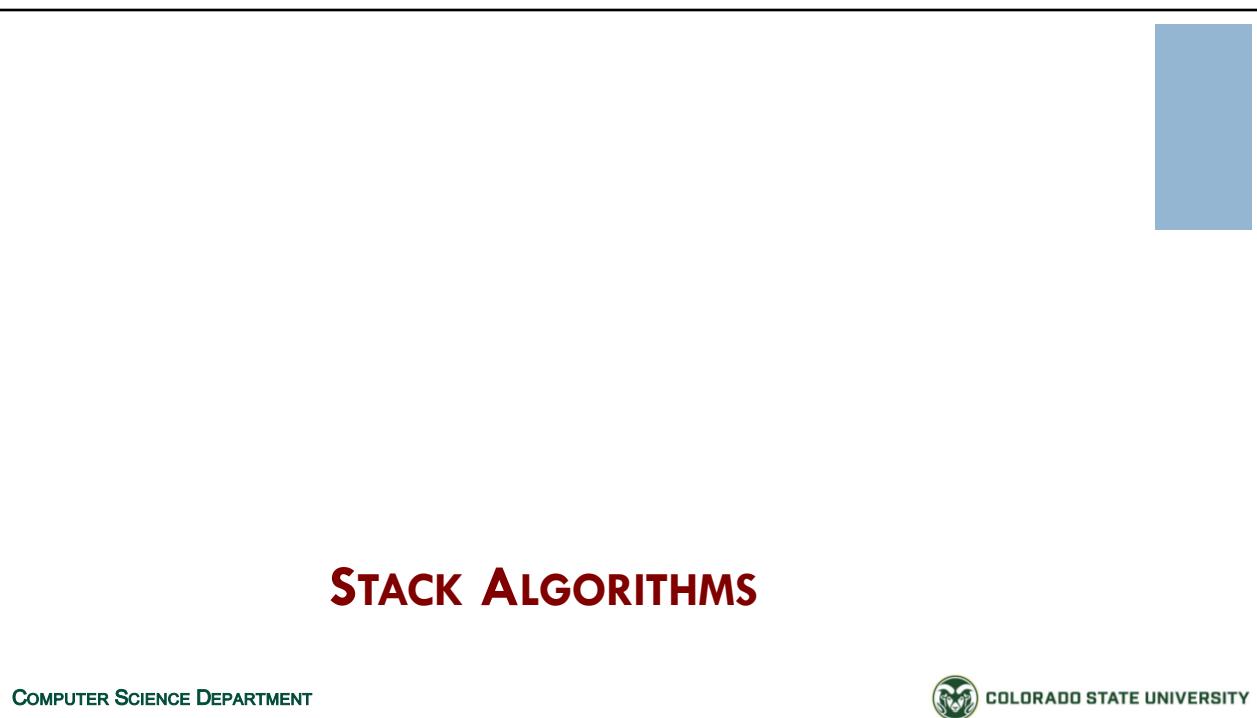
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.50

50



STACK ALGORITHMS

COMPUTER SCIENCE DEPARTMENT

 COLORADO STATE UNIVERSITY

51

The Model

- There is an array M
 - Keeps track of the state of memory
- M has as many elements as pages of virtual memory
- Divided into two parts
 - Top part: m entries {Pages currently in memory}
 - Bottom part: $n-m$ entries
 - Pages that were referenced BUT paged out

 COLORADO STATE UNIVERSITY

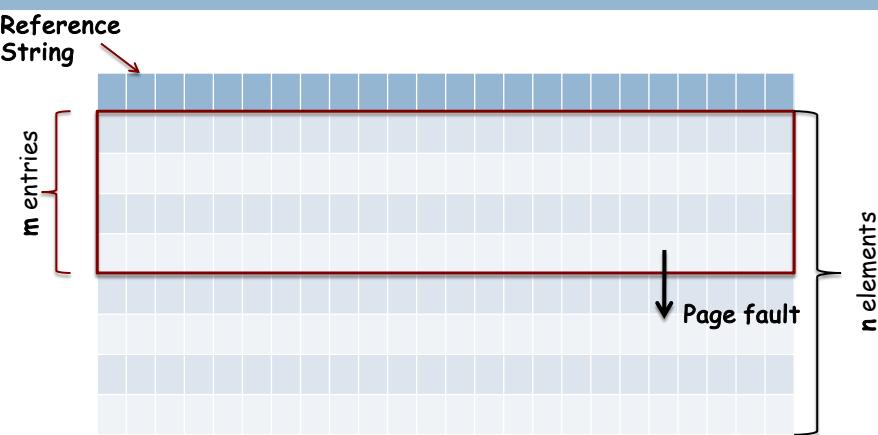
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.52

52

The model



Tracking the state of the array M over time



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.53

53

Properties of the model

- When a page is referenced?
 - Move to the **top** entry of M
- If the referenced page is already in M ?
 - All pages above it **moved down** one position
 - Pages below it are not moved
- **Transition** from within box to outside of it?
 - Page eviction from main memory



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.54

54

The model

0	2	1	3	5	4	6	3	7	4	7	3	3	5	5	3	1	1	1	7	2	3	4	1
0	2	1	3	5	4	6	3	7	4	7	7	3	3	5	3	3	3	1	7	2	3	4	
0	2	1	3	5	4	6	3	3	4	4	7	7	7	5	5	5	3	1	7	2	3		
0	2	1	3	5	4	6	6	6	6	4	4	4	7	7	7	5	3	1	7	2			
0	2	1	1	5	5	5	5	6	6	6	4	4	4	4	5	5	1	7					
0	2	2	1	1	1	1	1	1	1	1	6	6	6	6	4	4	5	5					
0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	6	6	6	6	6		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Properties of the model

- $M(m, r)$
 - The set of pages in the top part of \mathbf{M}
 - m page frames
 - r memory references



A property that has some interesting implications

- $M(m, r)$ subset of $M(m+1, r)$
- Set of pages in the top part of M with m frames
 - Also included in M with $(m+1)$ frames



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.57

57

What the subset relationship means

- Execute a process with a set of memory frames
- If we increase memory size by one frame and re-execute
 - **At every point of execution** all pages in the first execution are present in the second run
- Does not suffer from Belady's anomaly
 - **Stack algorithms**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.58

58

The contents of this slide-set are based on the following references

- Avi Silberschatz, Peter Galvin, Greg Gagne. *Operating Systems Concepts*, 9th edition. John Wiley & Sons, Inc. ISBN-13: 978-1118063330. [Chapter 9]
- Andrew S Tanenbaum. *Modern Operating Systems*. 4th Edition, 2014. Prentice Hall. ISBN: 013359162X / 978-0133591620. [Chapter 3]



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

VIRTUAL MEMORY

L23.59