Demand Paged Virtual Memory

Sarah Diesburg

Operating Systems

CS 3430

Up to this point…

- We assume that a process needs to load all of its address space before running
  - e.g., 0x0 to 0xFFFFFFFF
- Observation: 90% of time is spent on 10% of code

Demand Paging

- ***Demand paging***: allows pages that are referenced actively to be loaded into memory
  - Remaining pages stay on disk
  - Provides the illusion of infinite physical memory

Demand Paging Mechanism

- Page tables sometimes need to point to disk locations (as opposed to memory locations)
- A table entry needs a *present* (*valid*) bit
    - Present means a page is in memory
    - Not present means that there is a ***page fault***

Page Fault

- Hardware trap
- OS performs the following steps while running other processes (analogy: firing and hiring someone)
  - Choose a page
  - If the page has been modified, write its contents to disk
  - Change the corresponding page table entry and TLB entry
  - Load new page into memory from disk
  - Update page table entry
  - Continue the thread

Transparent Page Faults

- ***Transparent*** (invisible) mechanisms
  - A process does not know how it happened
  - It needs to save the processor states and the faulting instruction

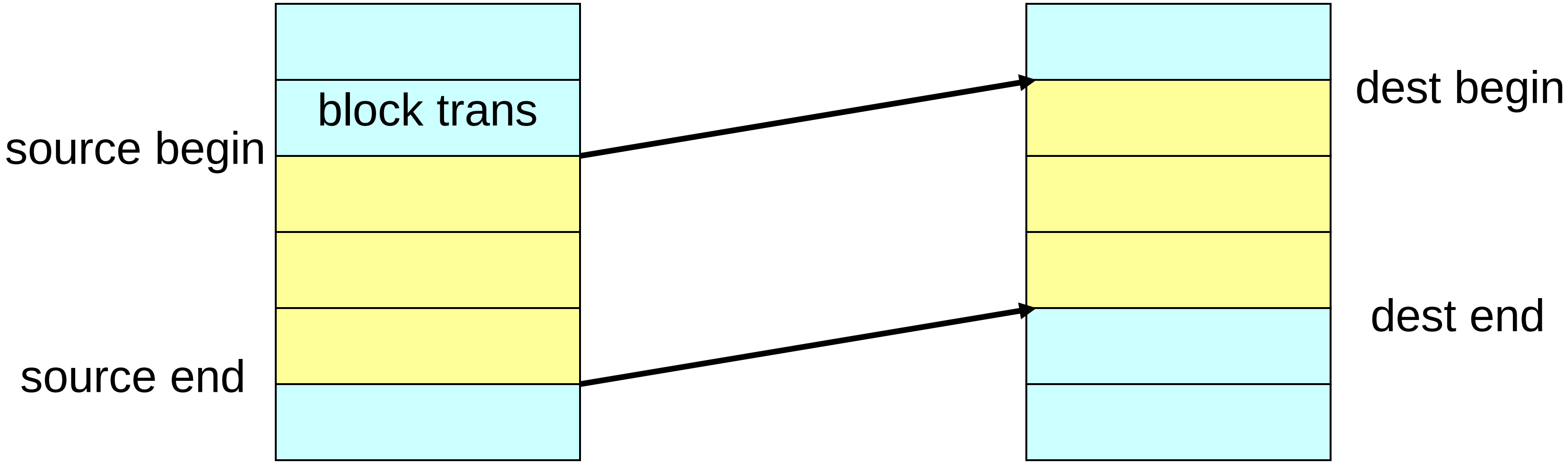
More on Transparent Page Faults

- An instruction may have side effects
  - Hardware needs to either unwind or finish off those side effects

```
ld r1, x
// page fault
```

More on Transparent Page Faults

- Hardware designers need to understand virtual memory
  - Unwinding instructions not always possible
  - Example: block transfer instruction



Page Replacement Policies

- ***Random replacement:*** replace a random page

  + Easy to implement in hardware (e.g., TLB)

  - May toss out useful pages

- ***First in, first out (FIFO):*** toss out the oldest page

  + Fair for all pages

  - May toss out pages that are heavily used

More Page Replacement Policies

- ***Optimal (MIN):*** replaces the page that will not be used for the longest time

  + Optimal

  - Does not know the future

- ***Least-recently used (LRU):*** replaces the page that has not been used for the longest time

  + Good if past use predicts future use

  - Tricky to implement efficiently

More Page Replacement Policies

- ***Least frequently used (LFU):*** replaces the page that is used least often
  - Tracks usage count of pages

  + Good if past use predicts future use

  - Difficult to replace pages with high counts

Example

- A process makes references to 4 pages: A, B, E, and R
  - Reference stream: BEERBAREBEAR
- Physical memory size: 3 pages

Beer?

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2												
3												

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E										
3												

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	*									
3												

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	*									
3				R								

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*									
3				R								

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*									
3				R			*					

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*				
3				R			*					

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*				
3				R			*					

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*	B			
3				R			*					

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*	B			
3				R			*					

FIFO

										↓		
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*	B			
3				R			*			E		

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	
2		E	*					*	B			
3				R			*			E		

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	
2		E	*					*	B			
3				R			*			E		

FIFO

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	R
2		E	*					*	B			
3				R			*			E		

FIFO

- 7 page faults

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	R
2		E	*					*	B			
3				R			*			E		

FIFO

- 4 compulsory cache misses

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	***B***				*	***A***					*	R
2		***E***	*					*	B			
3				***R***			*			E		

MIN

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	*									
3				R								

MIN

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

MIN

						↓						
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*		🟨	🟨	🟨			
2		E	*				🟨	🟨				
3				R			🟨					

MIN

						↓						
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A	🟨	🟨	🟨			
2		E	*				🟨	🟨				
3				R			🟨					

MIN

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*									
3				R			*					

MIN

								↓				
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*				
3				R			*					

MIN

									↓			
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A				🟨	🟨	
2		E	*					*		🟨		
3				R			*			🟨	🟨	🟨

MIN

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A				🟨	🟨	
2		E	*					*		🟨		
3				R			*		B	🟨	🟨	🟨

MIN

										↓		
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*		*		
3				R			*		B			

MIN

											↓	
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	
2		E	*					*		*		
3				R			*		B			

MIN

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	R
2		E	*					*		*		
3				R			*		B			

MIN

- 6 page faults

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	R
2		E	*					*		*		
3				R			*		B			

LRU

				↓								
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	*									
3				R								

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*			A						
3				R								

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*			A						
3				R			*					

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*			A						
3				R			*					

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E				
2		E	*			A						
3				R			*					

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E				
2		E	*			A						
3				R			*					

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E				
2		E	*			A			B			
3				R			*					

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			
3				R			*					

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			
3				R			*					

LRU

											↓	
Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			
3				R			*				A	

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			
3				R			*				A	

LRU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			R
3				R			*				A	

LRU

- 8 page faults

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			R
3				R			*				A	

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2												
3												

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E										
3												

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	2									
3												

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	2									
3				R								

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2							
2		E	2									
3				R								

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2							
2		E	2									
3				R		A						

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2							
2		E	2									
3				R		A	R					

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2							
2		E	2					3				
3				R		A	R					

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3				
3				R		A	R					

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3		4		
3				R		A	R					

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3		4		
3				R		A	R				A	

LFU

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3		4		
3				R		A	R				A	R

LFU

- 7 page faults

Memory page	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3		4		
3				R		A	R				A	R

Does adding RAM always reduce misses?

- Yes for LRU and MIN
  - Memory content of X pages $\subseteq$ X + 1 pages
- No for FIFO
  - Due to modulo math
  - ***Belady's anomaly:*** getting more page faults by increasing the memory size

Belady's Anomaly

- 9 page faults

Memory page	A	B	C	D	A	B	E	A	B	C	D	E
1	A			D			E					*
2		B			A			*		C		
3			C			B			*		D	

Belady's Anomaly

- 10 page faults

Memory page	A	B	C	D	A	B	E	A	B	C	D	E
1	A				*		E				D	
2		B				*		A				E
3			C						B			
4				D						C		

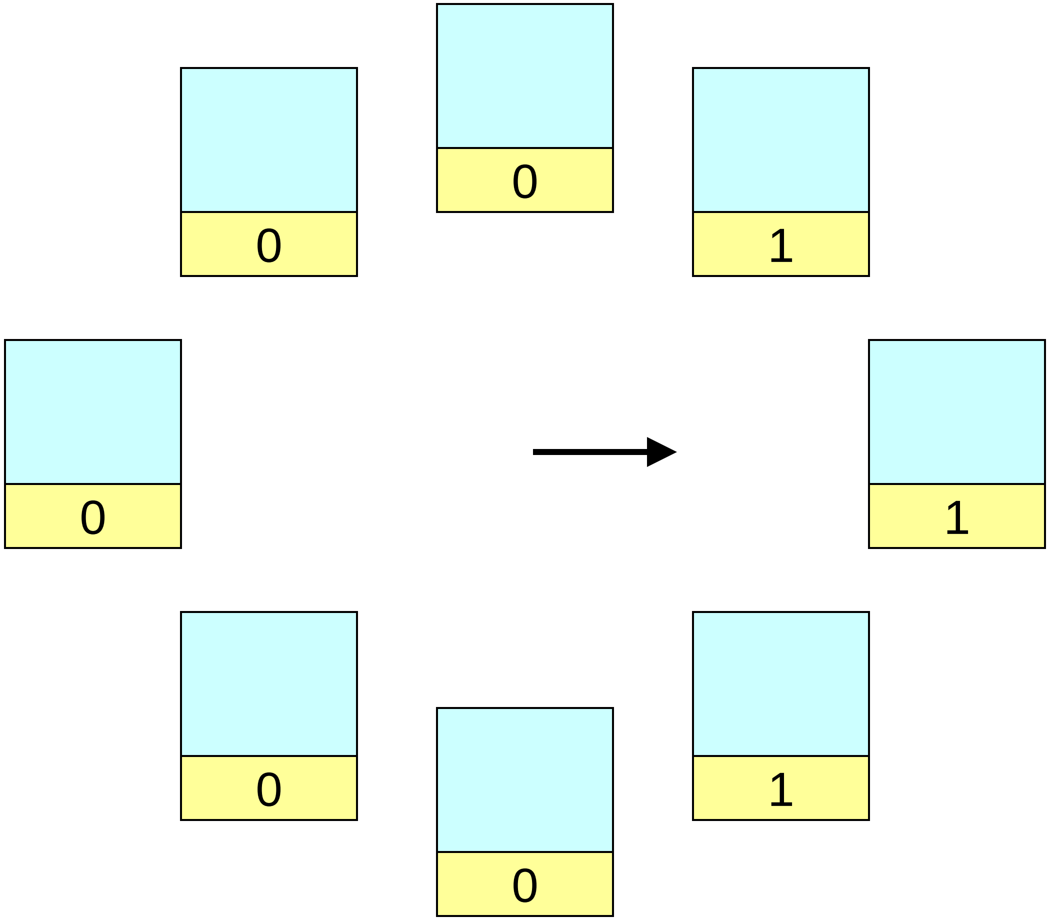
Implementing LRU

- Perfect LRU requires a timestamp on each reference to a cache page
  - Too expensive
- Common practice
  - Approximate the LRU behavior

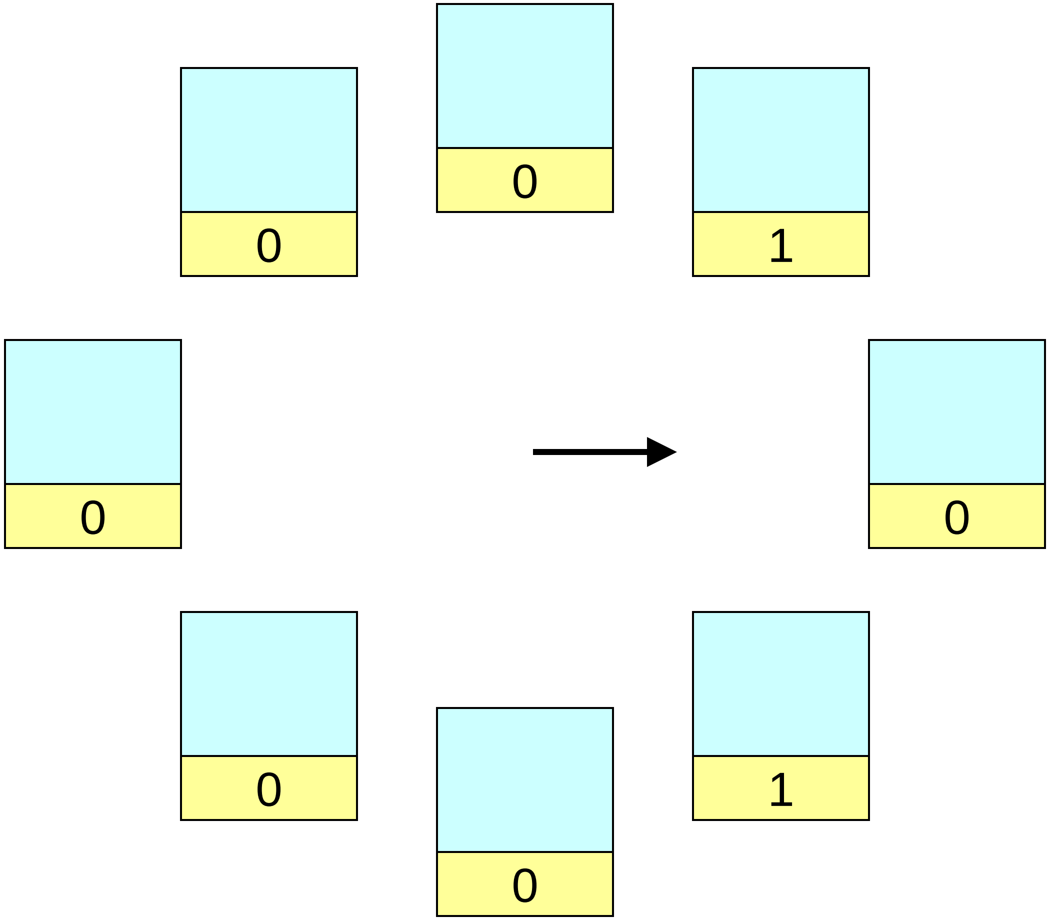
Clock Algorithm

- Replaces an old page, but not the oldest page
- Arranges physical pages in a circle
    - With a clock hand
- Each page has a ***used bit***
    - Set to 1 on reference
    - On page fault, sweep the clock hand
        - If the used bit == 1, set it to 0
        - If the used bit == 0, pick the page for replacement

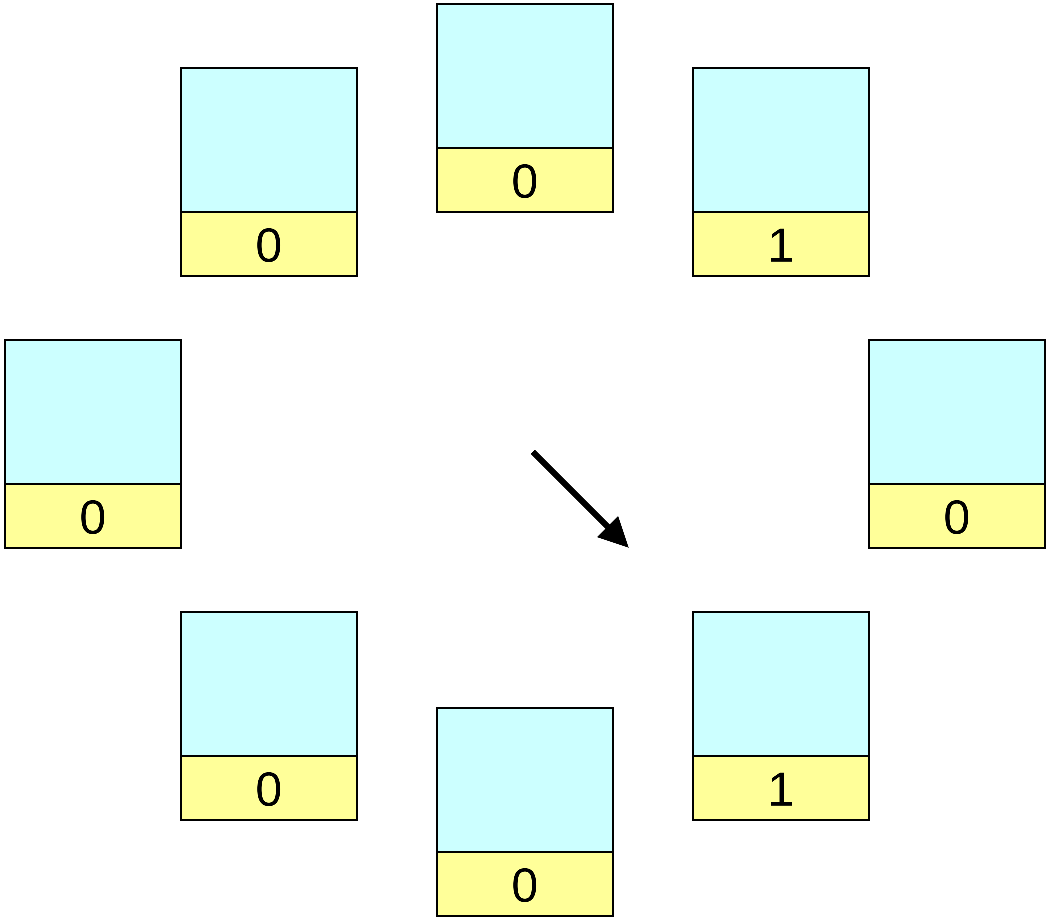
Clock Algorithm



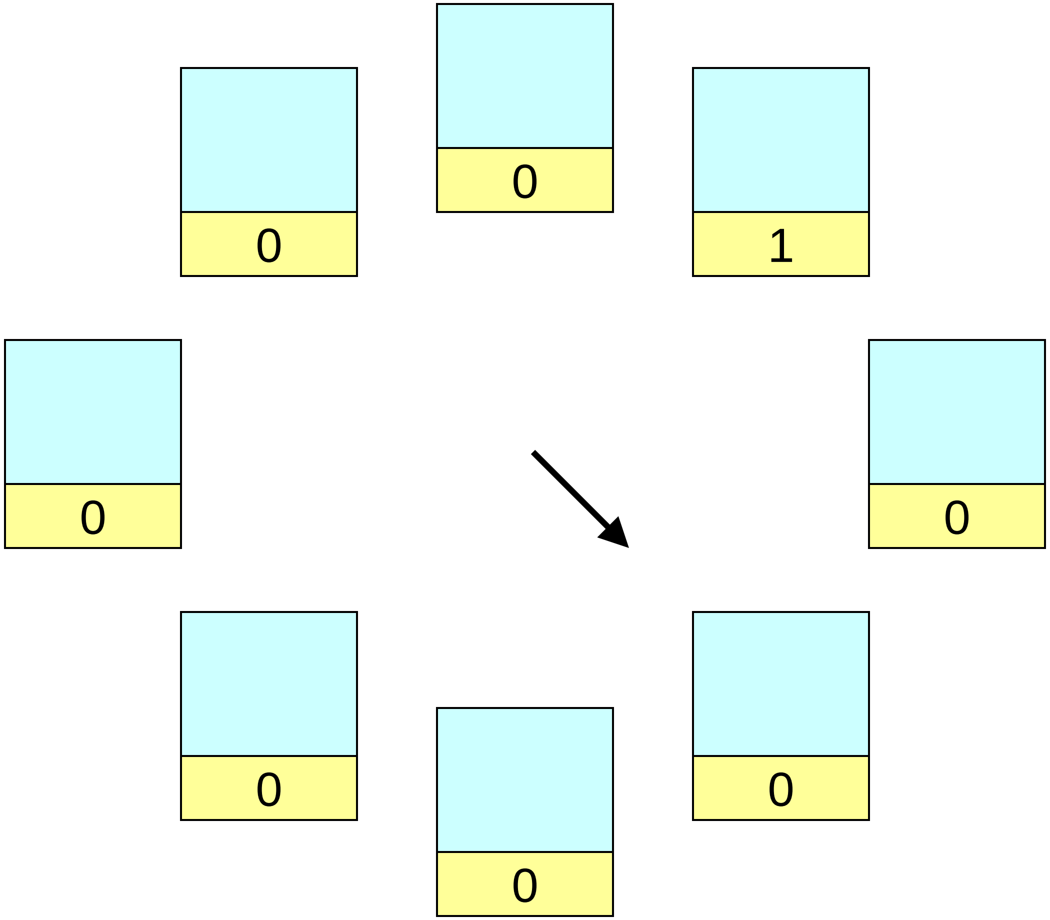
Clock Algorithm



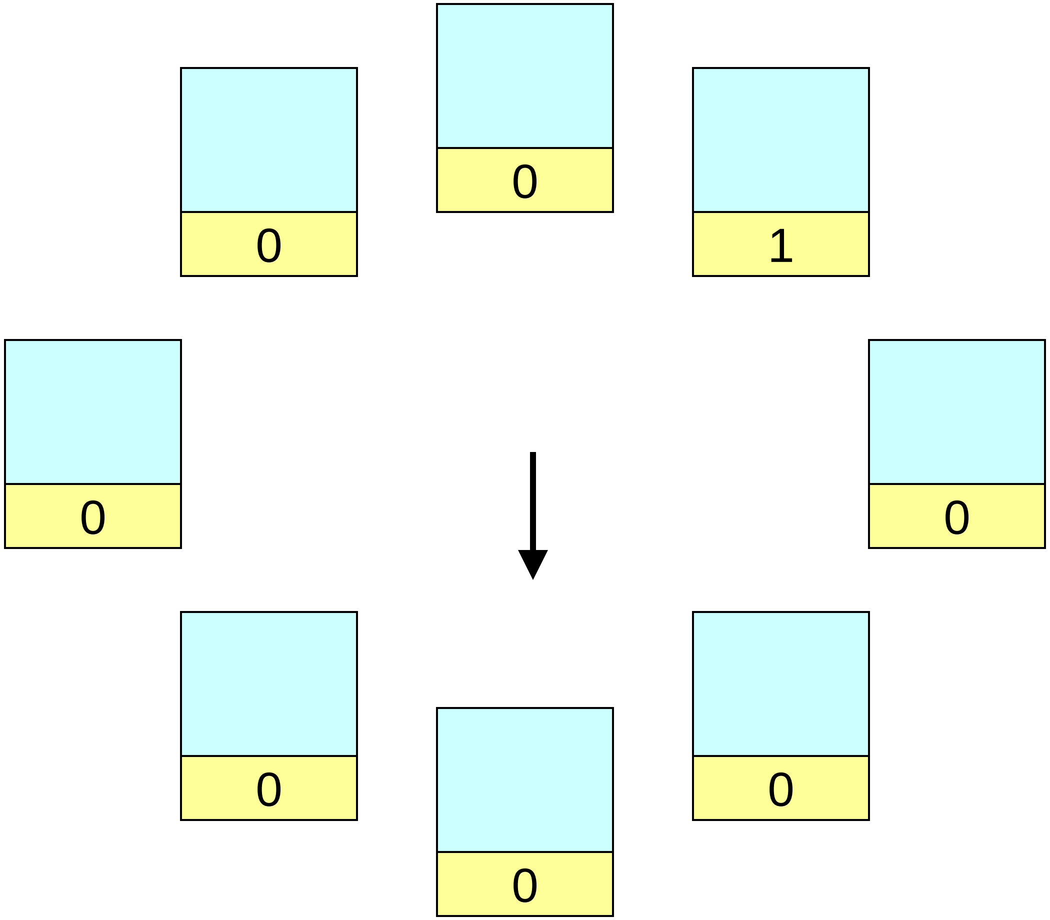
Clock Algorithm



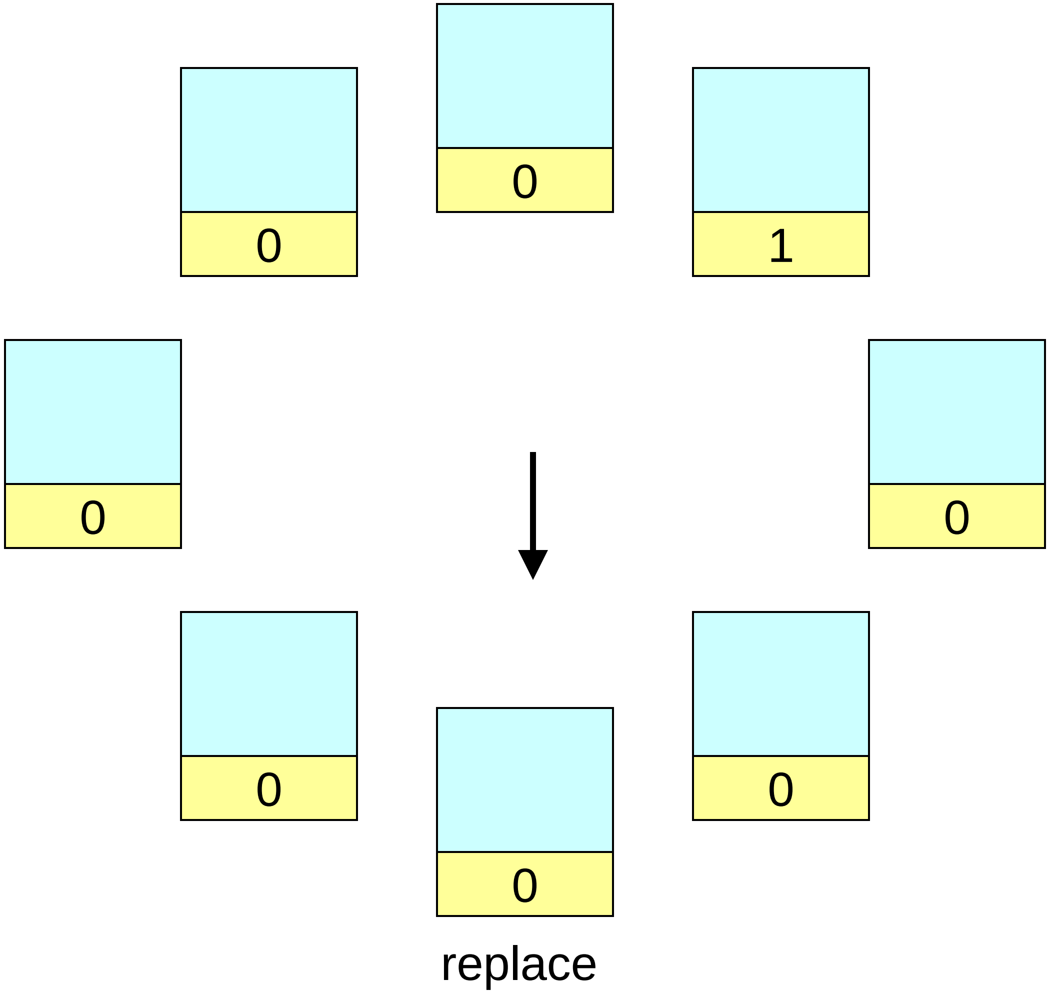
Clock Algorithm



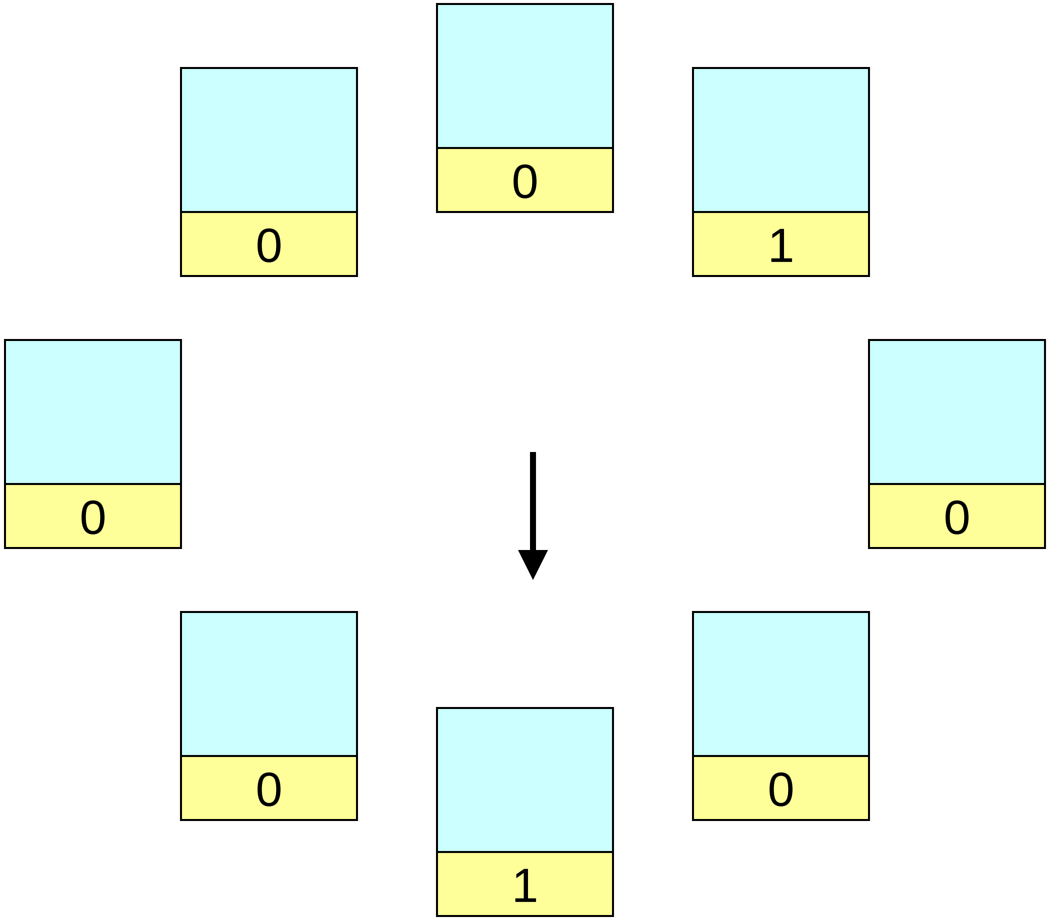
Clock Algorithm



Clock Algorithm



Clock Algorithm



Clock Algorithm

- The clock hand cannot sweep indefinitely
  - Each bit is eventually cleared
- Slow moving hand
  - Few page faults
- Quick moving hand
  - Many page faults

Nth Chance Algorithm

- A variant of clocking algorithm
  - A page has to be swept N times before being replaced
  - N → ∞, Nth Chance Algorithm → LRU
  - Common implementation
    - N = 2 for modified pages
    - N = 1 for unmodified pages

States for a Page Table Entry

- ***Used bit:*** set when a page is referenced; cleared by the clock algorithm
- ***Modified bit:*** set when a page is modified; cleared when a page is written to disk
- ***Valid bit:*** set when a program can legitimately use this entry
- ***Read-only:*** set for a program to read the page, but not to modify it (e.g., code pages)

Thrashing

- Occurs when the memory is overcommitted
  - Pages are still needed are tossed out
- Example
  - A process needs 50 memory pages
  - A machine has only 40 memory pages
  - Need to constantly move pages between memory and disk

Thrashing Avoidance

- Programs should minimize the maximum memory requirement at a given time
  - e.g., matrix multiplications can be broken into sub-matrix multiplications
- OS figures out the memory needed for each process
  - Runs only the computations that can fit in RAM

Working Set

- A set of pages that was referenced in the previous T seconds
  - T → ∞, working set → size of the entire process
- Observation
  - Beyond a certain threshold, more memory only slightly reduces the number of page faults

Working Set

- LRU, 3 memory pages, 12 page faults

Memory page	A	B	C	D	A	B	C	D	E	F	G	H
1	A			D			C			F		
2		B			A			D			G	
3			C			B			E			H

Working Set

- LRU, 4 memory pages, 8 page faults

Memory page	A	B	C	D	A	B	C	D	E	F	G	H
1	A				*				E			
2		B				*				F		
3			C				*				G	
4				D				*				H

Working Set

- LRU, 5 memory pages, 8 page faults

Memory page	A	B	C	D	A	B	C	D	E	F	G	H
1	A				*					F		
2		B				*					G	
3			C				*					H
4				D				*				
5									E			

Global and Local Replacement Policies

- ***Global replacement policy:*** all pages are in a single pool (e.g., UNIX)
  - One process needs more memory
    - Grabs memory from another process that needs less

  + Flexible

  - One process can drag down the entire system
- ***Per-process replacement policy:*** each process has its own pool of pages