

CS 1550

Week 10

Project 3

Teaching Assistant Henrique Potter

Simulate memory page allocation and page eviction algorithm

- Simulate memory page allocation and page eviction algorithm
 - Your program will read from a memory trace

- Simulate memory page allocation and page eviction algorithm
 - Your program will read from a memory trace

- 1 190a7c20
- s 3856bbe0
- 1 190afc20
- 1 15216f00
- 1 190a7c20
- 1 190a7c28
- 1 190a7c28
- 1 190aff38

- Simulate memory page allocation and page eviction algorithm
 - Your program will read from a memory trace
 - Access Type: load(l); store(s)
 - You will implement how loaded pages are evicted
 - 1 190a7c20
 - s 3856bbe0
 - 1 190afc20
 - 1 15216f00
 - 1 190a7c20
 - 1 190a7c28
 - 1 190a7c28
 - 1 190aff38

- Since it is a 32-bit address space.
 - Each page in size of 4KB (2^12 bytes per page)

- 1 190a7c20
- s 3856bbe0
- 1 190afc20
- 1 15216f00
- 1 190a7c20
- 1 190a7c28
- 1 190a7c28
- 1 190aff38

- Since it is a 32-bit address space.
 - First 20 bits is used for the address

Page Address



1 190a7c20

s 3856bbe0

1 190afc20

1 15216f00

1 190a7c20

1 190a7c28

1 190a7c28

- Since it is a 32-bit address space.
 - First 20 bits is used for the address
 - The rest is used for offset

Page Address Page Offset



1 190a7c20

s 3856bbe0

1 190afc20

1 15216f00

1 190a7c20

1 190a7c28

1 190a7c28

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

0	
1	
2	

1 190a7c20

s 3856bbe0

1 190afc20

1 15216f00

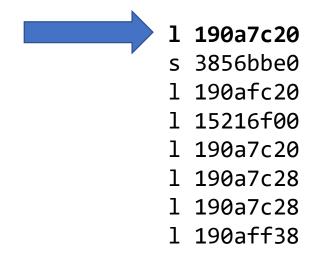
1 190a7c20

1 190a7c28

1 190a7c28

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

0	
1	
2	



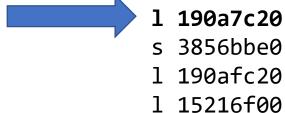
- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

 0

 1

 2

Pagefault since it is not in the page table



1 190a7c20

1 190a7c28

1 190a7c28

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

0	190a7
1	
2	

Pagefault since it is not in the page table



1 190afc20

1 15216f00

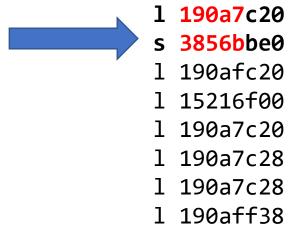
l 190a7c20

1 190a7c28

1 190a7c28

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

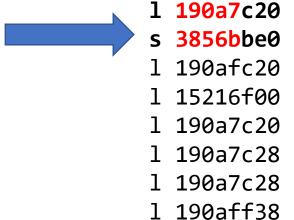
0	190a7
1	
2	



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

0	190a7
1	
2	

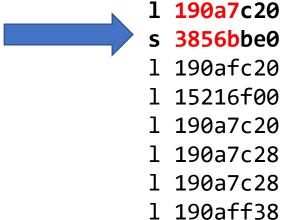
Pagefault since it is not in the page table



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

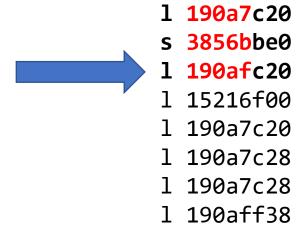
0	190a7
1	3856b
2	

Pagefault since it is not in the page table



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

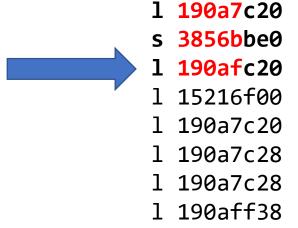
0	190a7
1	3856b
2	



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

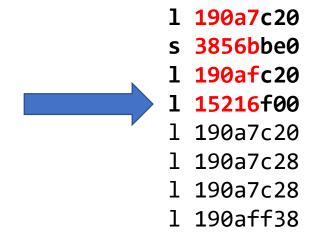
0	190a7
1	3856b
2	190af

Pagefault since it is not in the page table



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

0	190a7
1	3856b
2	190af

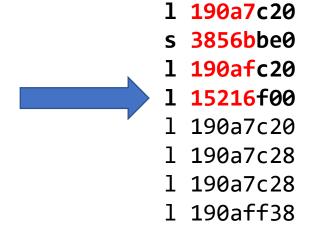


- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

Pagefault again

0	190a7
1	3856b
2	190af

We need to evict someone!!

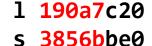


- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

0	190a7
1	3856b
2	190af

We need to evict someone!!

Pagefault again





1 15216f00

l 190a7c20

1 190a7c28

1 190a7c28

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

Pagefault again



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

Pagefault again



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Least Recently Used(LRU)

Pagefault again



- Let's suppose you have 12KB of physical memory
 - Page has 4KB

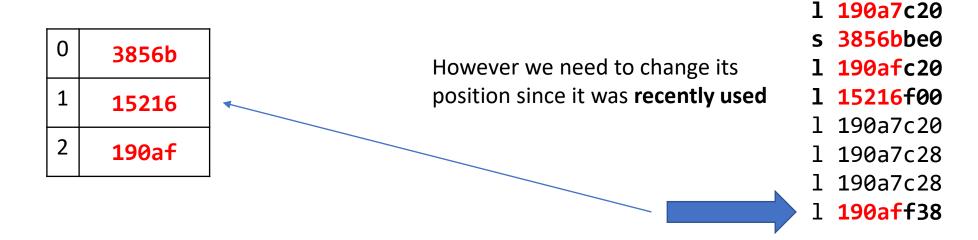
0	3856b	Assume we skip to page 190af no	3856bbe0 190afc20
1	190af	page fault would occur since it is	15216f00
2	15216	already in the page table	190a7c20 190a7c28
			190a7c28 190aff38

1 190a7c20

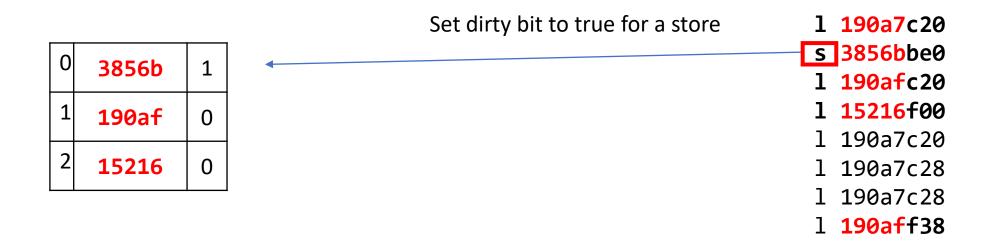
- Let's suppose you have 12KB of physical memory
 - Page has 4KB

			1	190a7c20
0	3856b		S	3856bbe0
	<u> </u>	However we need to change its	1	190 afc20
1	190af	position since it was recently used	1	15216 f00
			1	190a7c20
2	15216		1	190a7c28
			1	190a7c28
			1	190aff38

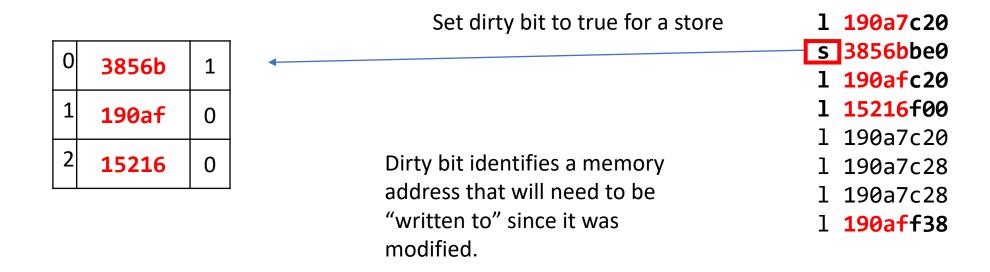
- Let's suppose you have 12KB of physical memory
 - Page has 4KB



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Set dirty bit to true if a store



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Set dirty bit to true if a store



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm

0	
1	
2	

1 190a7c20

s 3856bbe0

1 190a7c24

1 190afc20

1 15216f00

l 190a7c20

1 190a7c28

1 190a7c28

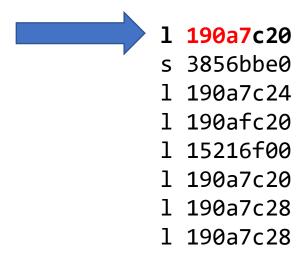
- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit): set to 1 If page accessed again after allocated in memory

0				
1				
2				
1				
	R bits			

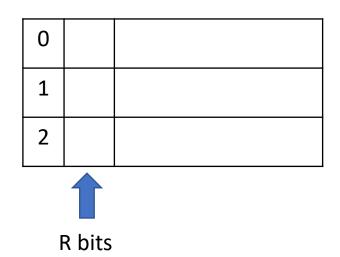
1 190a7c20
s 3856bbe0
l 190a7c24
l 190afc20
l 15216f00
l 190a7c20
l 190a7c28
l 190a7c28

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

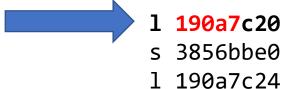
0				
1				
2				
1	R bits			



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)



Pagefault since it is not in the page table



1 190afc20

1 15216f00

l 190a7c20

l 190a7c28

1 190a7c28

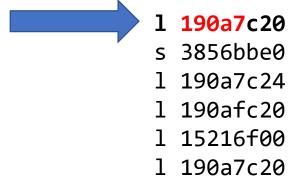
- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

0	0	190a7	
1			
2			
R bits			

Pagefault since it is not in the page table

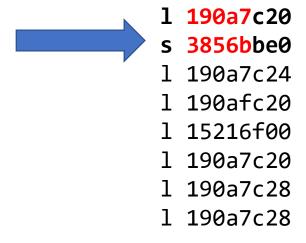
1 190a7c28

1 190a7c28



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

0	0	190a7	
1			
2			
R bits			



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

0	0	190a7	
1			
2			
1			
R bits			

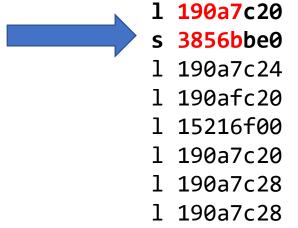
Pagefault since it is not in the page table



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

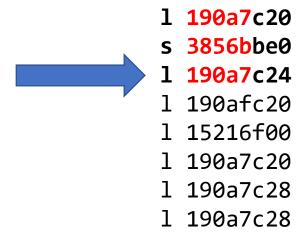
0	0	190a7		
1	0	3856b		
2				
ſ	R bits			

Pagefault since it is not in the page table



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

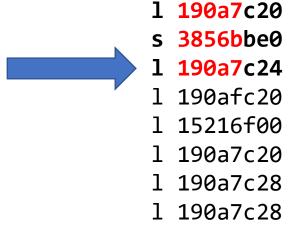
0	0	190a7
1	0	3856b
2		
	1	
	R bits	



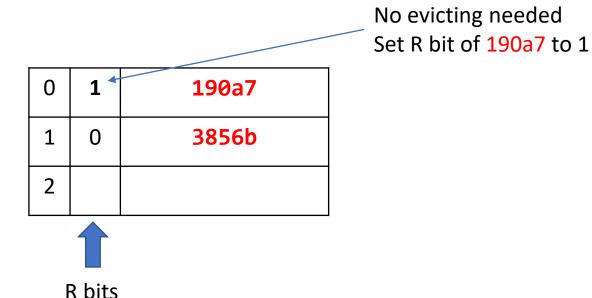
- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

0	0	190a7
1	0	3856b
2		
F	R bits	

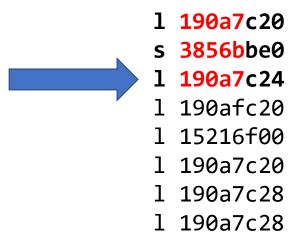
Page 190a7 accessed again



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

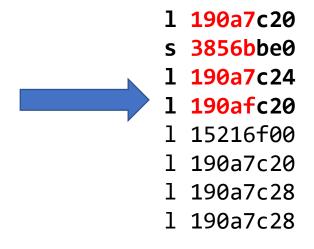


Page 190a7 accessed again



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

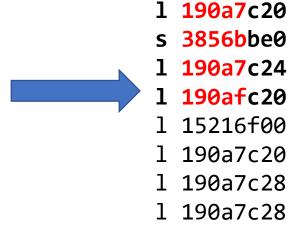
0	1	190a7
1	0	3856b
2		
	1	
R bits		



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

0	1	190a7
1	0	3856b
2		
	1	
R bits		

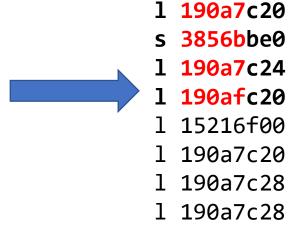
Pagefault since it is not in the page table



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

0	1	190a7
1	0	3856b
2	0	190af
1		
R bits		

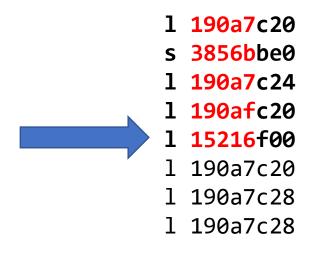
Pagefault since it is not in the page table



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

R bits

0	1	190a7
1	0	3856b
2	0	190af
•	1	

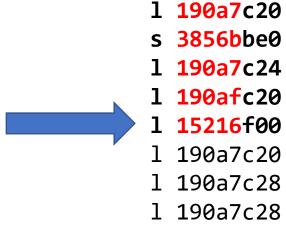


- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

R bits

0	1	190a7
1	0	3856b
2	0	190af

b f

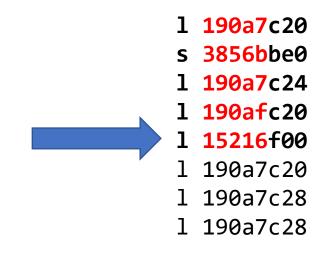


- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

R bits

0	1	190a7
1	0	3856b
2	0	190af
•		

We need to evict someone!!

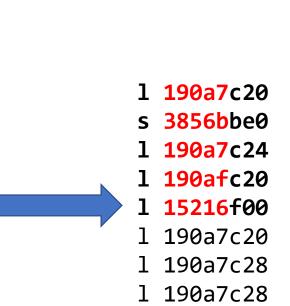


- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

R bits

0	1	190a7
1	0	3856b
2	0	190af
	1	

We need to evict someone!!



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

1 190a7c20 We need to evict 3856bbe0 190a7 0 1 someone!! 1 190a7c24 1 190afc20 3856b 0 1 15216f00 2 190af 0 1 190a7c20 1 190a7c28 1 190a7c28 R bits

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

1 190a7c20 We need to evict 3856bbe0 190a7 0 1 someone!! 1 190a7c24 Entry 0: R bit is 1 1 190afc20 3856b 0 1 15216f00 2 190af 0 1 190a7c20 1 190a7c28 1 190a7c28 R bits

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

R bits

We need to evict someone!!

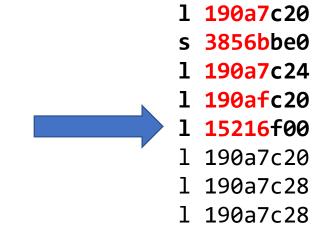
1 0 3856b

2 0 190af

We need to evict someone!!

Entry 0: R bit is 1

Set it to 0 and go to next entry



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

1 190a7c20 We need to evict 3856bbe0 190a7 0 0 someone!! 1 190a7c24 Entry 1: R bit is 0 1 190afc20 3856b 0 1 15216f00 2 190af 0 1 190a7c20 1 190a7c28 1 190a7c28 R bits

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

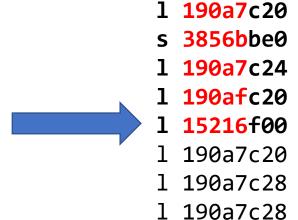
R bits

We need to evict someone!!

1 0 3856b

2 0 190af

We need to evict someone!!
Entry 1: R bit is 0
Evict entry 1



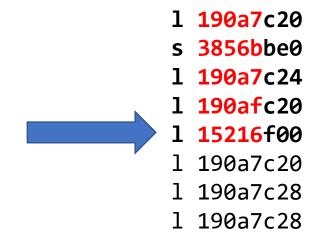
- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

R bits

We need to evict someone!!

Entry 1: R bit is 0

Evict entry 1

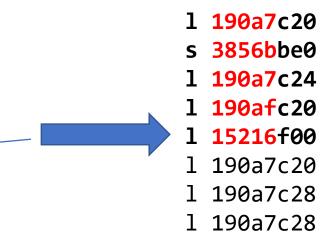


- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

R bits

0	0	190a7
1	0	190af
2	0	15216

We need to evict someone!! Entry 1: R bit is 0 Evict entry 1



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume Second Chance Algorithm
 - Referenced bit (R bit)

			We need to evict
0	0	190a7	someone!!
1	0	190af	Entry 1: R bit is 0 Evict entry 1
2	0	15216	•
	1		Similar as FIFO but pages accessed
ſ	R bits		again will get another chance

Pagefault again

l 190a7c20 s 3856bbe0 l 190a7c24 l 190afc20 l 15216f00 l 190a7c20 l 190a7c28 l 190a7c28

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT

0	
1	
2	

1 190a7c20

s 3856bbe0

1 190afc20

1 15216f00

1 190a7c20

1 190a7c28

1 190a7c28

1 190aff38

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning

0	
1	
2	

l 190a7c20

s 3856bbe0

1 190afc20

1 15216f00

1 190a7c20

1 190a7c28

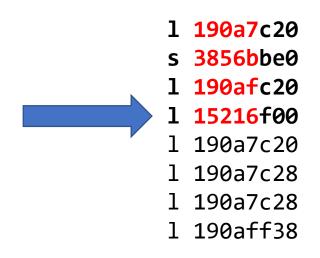
l 190a7c28

1 190aff38

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning
 - When evicting needed

0	190a7	
1	3856b	
2	190af	

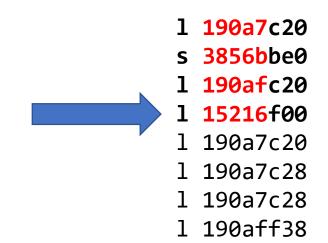
We need to evict someone!!



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning
 - When evicting needed

0	190a7	_
1	3856b	
2	190af	

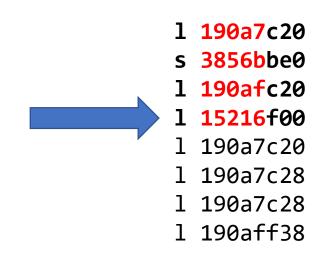
We need to evict someone!!



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning
 - When evicting needed

0	190a7 ←
1	3856b
2	190af

We need to evict someone!!
But page 190a7 will be used later!



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning
 - When evicting needed

0	190a7	
1	3856b	1
2	190af	

We need to evict someone!!
But page 190a7 will be used later!

Go next until find one that is no longer needed in the future

Pagefault again

s 3856bbe0
1 190afc20
1 15216f00
1 190a7c20

1 190a7c20

1 190a7c28

1 190a7c28

1 190aff38

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning
 - When evicting needed

0	190a7
1	190af
2	

We need to evict someone!!
But page 190a7 will be used later!

Go next until find one that is no longer needed in the future

Pagefault again

1 190a7c20 s 3856bbe0

1 190afc20

1 15216f00

l 190a7c20

1 190a7c28

1 190a7c28

1 190aff38

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning
 - When evicting needed

0	190a7
1	190af
2	15216

We need to evict 1 190a7c20 someone!! 3856bbe0 But page 190a7 will 1 190afc20 be used later! 1 15216f00 1 190a7c20 Go next until find 1 190a7c28 one that is no 1 190a7c28 longer needed in 1 190aff38 the future

- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning
 - When evicting needed

0	190a7
1	3856b
2	15216

What if there is a tie?
Multiple pages no longer needed in the future?



- Let's suppose you have 12KB of physical memory
 - Page has 4KB
 - Assume OPT
 - You already know memory access trace at the beginning
 - When evicting needed

0	190a7	
1	3856b	
2	15216	

What if there is a tie? Multiple pages no longer needed in the future?

Use LRU among those tie pages



- Few Tips about OPT:
 - If you use naïve way of looking up across all traces you will **TIMEOUT**

- Few Tips about OPT:
 - If you use naïve way of looking up across all traces you will **TIMEOUT**
 - How to reduce looking up time

- Few Tips about OPT:
 - If you use naïve way of looking up across all traces you will **TIMEOUT**
 - How to reduce looking up time: Save all page information in a hash table

- Few Tips about OPT:
 - If you use naïve way of looking up across all traces you will TIMEOUT
 - How to reduce looking up time: Save all page information in a hash table
 - <Page>:<recording number of lines>

- Few Tips about OPT:
 - If you use naïve way of looking up across all traces you will TIMEOUT
 - How to reduce looking up time: Save all page information in a hash table
 - <Page>:<recording number of lines>

- l 190a7c20
- s 3856bbe0
- 1 190afc21
- 1 15216f00
- 1 190a7c22
- 1 190aff38

- Few Tips about OPT:
 - If you use naïve way of looking up across all traces you will TIMEOUT
 - How to reduce looking up time: Save all page information in a hash table
 - <Page>:<recording number of lines>

190a7	0, 4
3856b	1
190af	2,5
15216	3

1 **190a7**c20

s **3856b**be0

1 190afc21

1 **15216**f00

1 **190a7**c22

1 **190af**f38

- No need to use qemu
- You will write the simulator from scratch with Java, C/C++,Perl, or Python
- Read from memory traces text files
- Count the number of events (pagefaults, page evictions etc.)
 - Compare eviction algorithms

CS 1550 – Project 3

- **Due**: Monday, April 6th, 2020 @11:59pm
- Late: Wednesday, Apr 8th, 2020 @11:59pm with 10% reduction per late day