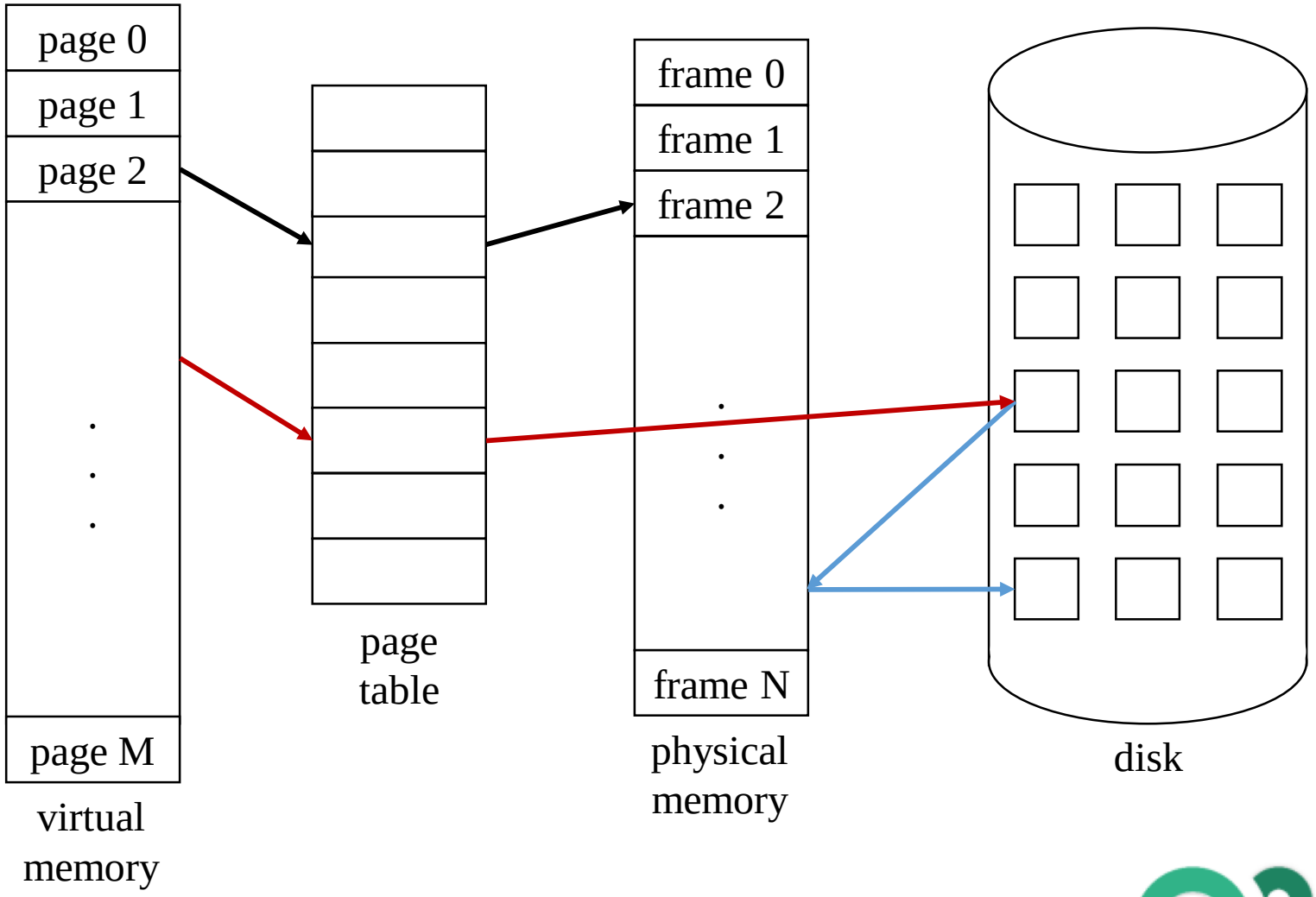
OS 2019 Homework4

Memory Manager

(Due date: 2020/1/2 23:59)

Overview



Requirements

- Implement a paging based memory manager
  - Use an one-level page table for mapping virtual pages to physical frames
  - Implement three page replacement policies
    - FIFO
    - Enhanced second chance algorithm
    - Segmented LRU
- Show the page fault rate and other information under the policies

Requirements

- Input: a sequence of virtual page accesses (trace file)

- Execute Command Format (using I/O Redirection)
  - ./memory_manager < INPUT_FILE > OUTPUT_FILE

```
./memory_manager < input.txt > output.txt
```

  - < INPUT_FILE : redirect stdin to INPUT_FILE
  - > OUTPUT_FILE : redirect stdout to OUTPUT_FILE

Assumptions

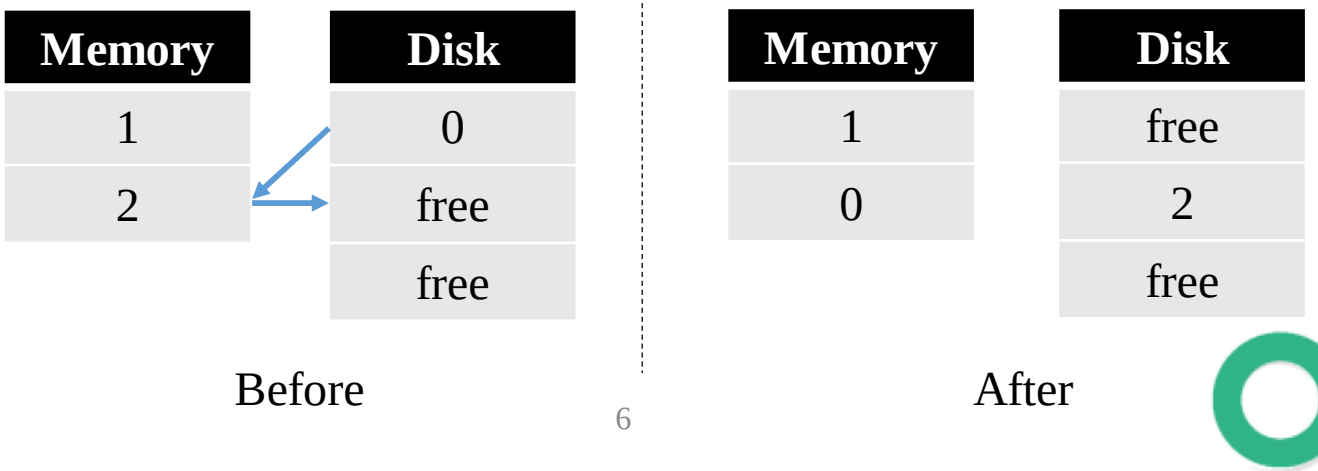
- No TLB support
  - Each page access needs to consult the page table
- Only a single process
- Evicted page should write back to the disk whether it is dirty page or not, which is not a general method in real world
- Disk always has enough space for evicted pages
- There will be M virtual pages and N physical frames
  - M and N will be given in the trace file
  - M > N

Requirements - Memory Manager

- If a page fault occurs, and physical memory
  - **is not full**: page-in the fault page from the disk to the frame with smallest free frame number
  - **Is full**: page-in the fault page from the disk to the frame selected by page replacement policy

- To page-out a page, select an free disk block with the smallest disk block number

Example:
Reference 0



Requirements - Page Table

- In-Use Bit
  - **1**: the page table entry is in-use, set as first time reference to this page
  - **0**: the page table entry is not in-use
- Present Bit
  - **1**: the page is in physical memory
  - **0**: the page is not in physical memory

VPI	PFI / DBI	In-Use	Present
0	4	1	0
1	0	1	0
2	0	0	0
	...		
M - 1	2	1	1

VPI : virtual page index
PFI : physical frame index
DBI : disk block index

Input Trace File Format

- Line 1~3 are configs
  - Which Policy?
  - Number of Virtual Page M
    - M ≧ 2
  - Number of Physical Frame N
    - N ≧ 1

```
1 Policy: FIFO | ESCA | SLRU
2 Number of Virtual Page: M
3 Number of Physical Frame: N
4 -----Trace-----
5 Read 0
6 Write 1
…
Z Write 2
```

- Line 5~Z will be traces
  - ***Read / Write X (VPI)***: to reference virtual page ***X***

Output File Format

- Show **Miss/Hit** and related information for each reference
  - Format for a hit: **Hit, VPI=>PFI**
  - Format for a miss: **Miss, PFI, Evicted VPI>>Destination, VPI<<Source**
    - **Source:** the block index of the page which is **page-in** from disk
    - **Destination:** the block index where the evicted page **page-out**
    - For first reference, there might be no **source** || **destination** || **Evicted VPI**, set the value as -1

Output File Format

- At the end, show the **page fault rate**

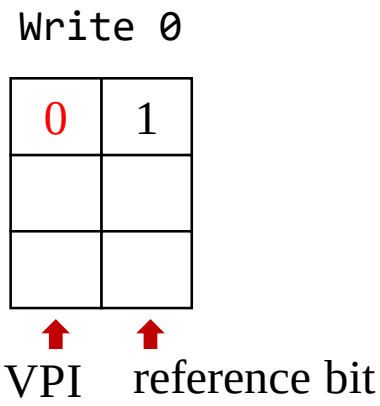
```
1 Policy: FIFO
2 Number of Virtual Page: 3
3 Number of Physical Frame: 2
4 -----Trace-----
5 Write 2
6 Write 0
7 Write 1
8 Read 2
9 Write 2
```

```
1 Miss, 0, -1>>-1, 2<<-1
2 Miss, 1, -1>>-1, 0<<-1
3 Miss, 0, 2>>0, 1<<-1
4 Miss, 1, 0>>1, 2<<0
5 Hit, 2=>1
6 Page Fault Rate: 0.800
7
```

three decimal place accuracy

Second Chance Algorithm

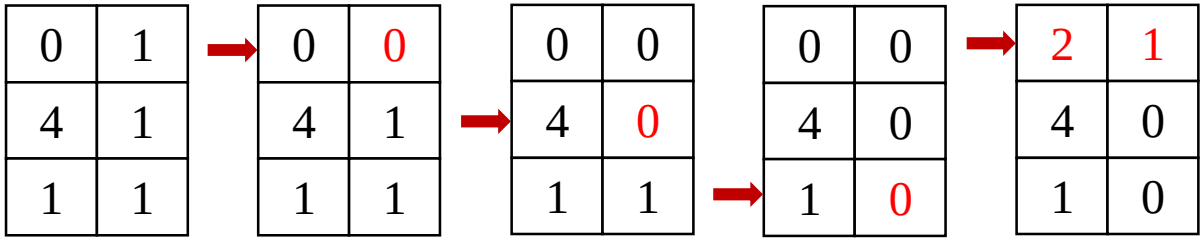
- Assumption
  - The reference bit will be set as 1 when a page is swapped in or referenced
  - Clock algorithm for replacement



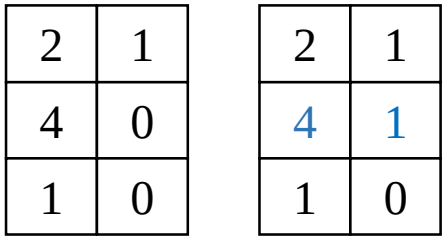
Second Chance Algorithm

```
Policy: SCA
Number of Virtual Page: 5
Number of Physical Frame: 3
-----Trace-----
Write 0
Write 4
Write 1
Write 2
Read 4
Read 3
```

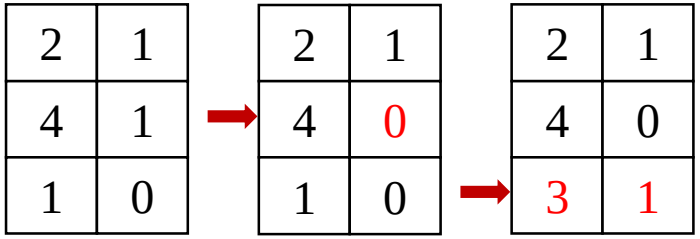
Write 2



Read 4



Read 3

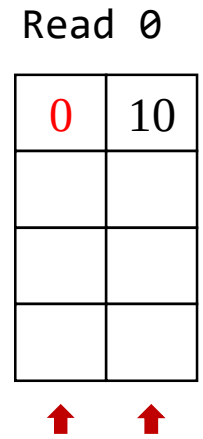


Enhanced Second Chance Algorithm

- Assumption
  - The reference bit will be set as 1 when a page is swapped in or referenced
  - The dirty bit will be set as 1 when a page is written
  - Clock algorithm for replacement

- Four classes (reference bit, dirty bit)
  - (0, 0)
  - (0, 1)
  - (1, 0)
  - (1, 1)

Higher class is better to replace



Enhanced Second Chance Algorithm

- Rules
  - (a) Cycle through the frame looking for (0, 0)
    If one is found, use that page
  - (b) Cycle through the frame looking for (0, 1)
    Set the reference bit to 0 for all frames bypassed
  - (c) If step (b) failed, all reference bits will now be zero and repetition of steps (a) and (b) are guaranteed to find a frame for replacement

Enhanced Second Chance Algorithm

(a) looking for (0, 0)

Write 2

0	11
1	10
3	10
6	10

→

0	11
1	10
3	10
6	10

```
Policy: ESCA
Number of Virtual Page: 7
Number of Physical Frame: 4
-----Trace-----
Write 0
Read 1
Read 3
Read 6
Write 2
```

(b) looking for (0, 1)

0	11
1	10
3	10
6	10

→

0	01
1	10
3	10
6	10

→

0	01
1	00
3	10
6	10

→

0	01
1	00
3	00
6	10

→

0	01
1	00
3	00
6	00

(c) Repeat steps (a) and (b)

0	01
1	00
3	00
6	00

→

0	01
1	00
3	00
6	00

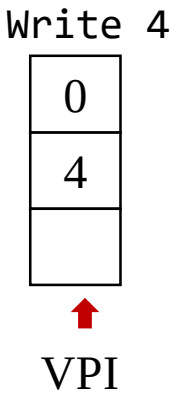
→

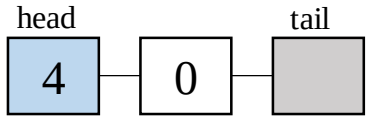
0	01
2	11
3	00
6	00

LRU

- Replace the page that has not been used for the longest period of time

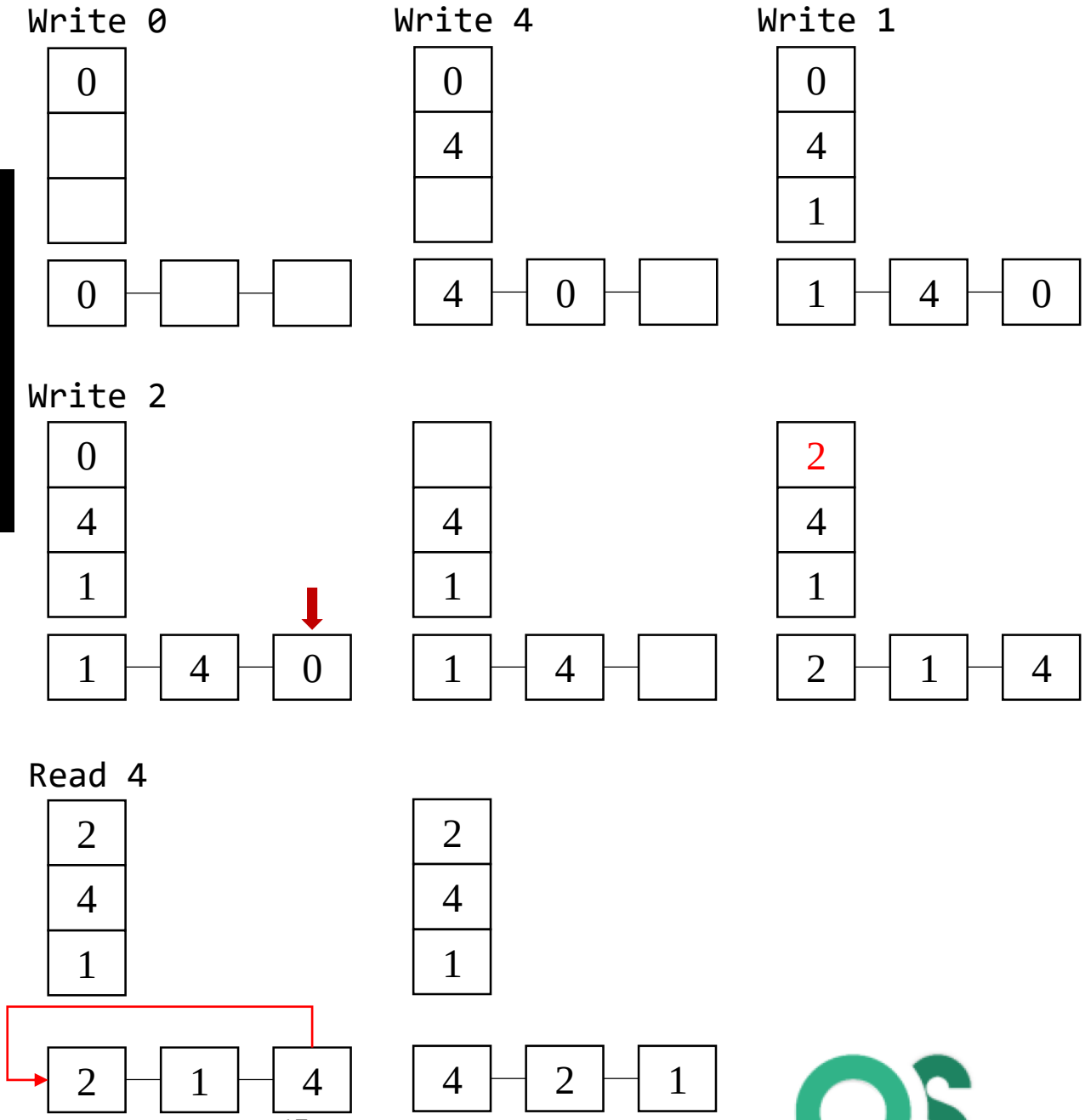
- Once a page is referenced, it will be added to the head of the LRU list





LRU

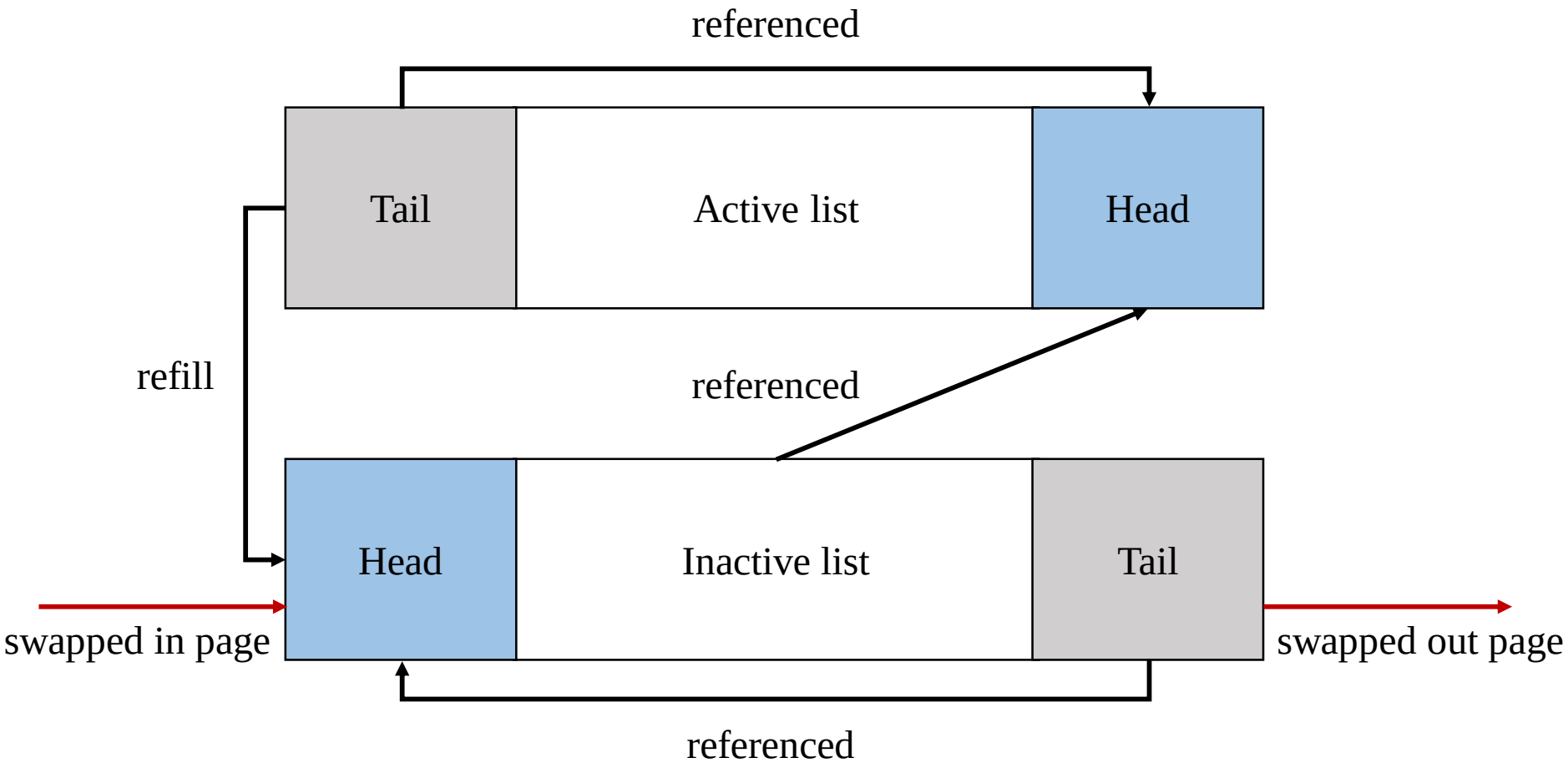
```
Policy: LRU
Number of Virtual Page: 5
Number of Physical Frame: 3
-----Trace-----
Write 0
Write 4
Write 1
Write 2
Read 4
```



Property of LRU

- Appropriate for accessing hot data
- Not appropriate for periodically batch access
- Sporadic access may easily cause buffer pollution

Segmented LRU



Segmented LRU

- Assumption
  - The reference bit will be set as 1 when a page is swapped in
  - The slot number of inactive list will be ⌈ frame number / 2 ⌉
  - The slot number of active list will be ⌊ frame number / 2 ⌋

Segmented LRU

- Rules
  - Page hit in active list
    if the page has reference bit = 0
      set reference bit and move the page to active list head
    if the page has reference bit = 1
      move the page to active list head
  - Page hit in inactive list
    if the page has reference bit = 0
      set reference bit and move the page to inactive list head
    if the page has reference bit = 1
      clear reference bit and move the page to active list head

Segmented LRU

- Rules
  - Replacement in active list
    (occurs when the page in inactive list with reference bit = 1 page hit
    but there is no free space in active list)
    - Search the tail of the active list for evicted page
      if the page has reference bit = 0
        refill the page to inactive list head
      if the page has reference bit = 1
        clear reference bit and move the page to active list head
        search the tail of the active list for another evicted page

Segmented LRU

- Rules
  - Replacement in inactive list

    (occurs when page miss but there is no free space in inactive list)
    - Search the tail of the inactive list for evicted page
      if the page has reference bit = 0
        swapped out the page
      if the page has reference bit = 1
        clear reference bit and move the page to inactive list head

        search the tail of the inactive list for another evicted page

    - Swapped in page will be placed in the physical frame which contained the evicted page

Segmented LRU