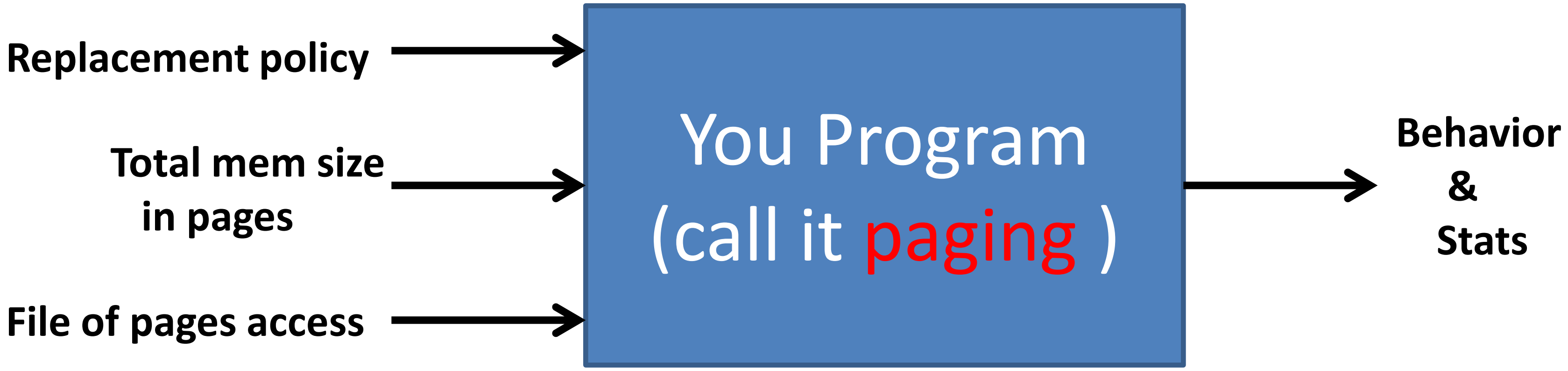
Page Replacement

The Aim Of This Lab

Implement two replacement policies (NRU and aging) and compare them in terms of performance.



Exact command line: ./***paging policy size file***

Replacement Policy: NRU

0: NRU

Lecture “Memory Management III”

slides 14 and 15 (The one in this lab is a bit modified from the one in slides 14 and 15)

- Two bits per page table entry: R & M
  - R: set to 1 when page is read
  - M: set to 1 when page is written
- When we need to replace a page, the one with the lowest number is picked. If more than one page have the same low number, pick the first page in memory with the lowest number. That is, if pages at mem[5] and mem[8] both have lowest counter value, replace the page at mem[5].

R	**M**
0	0
0	1
1	0
1	1

Replacement Policy: aging

1: aging
Lecture “Memory Management III”
slide 26 (The one in this lab is a bit modified from the one in slide 26)

- Initially: counter = 0 for each page table entry
- At each page access:
  - The page that is accessed (read/write)
    - its R = 1
    - shift its counter to the right by 1
    - add the R to the leftmost of the counter.
  - All other pages:
    - shift its counter to the right by 1
    - add the page R bit to the leftmost bit of the counter
    - Clear R bit
- At page fault: replace the page with the lowest counter. If more than one page have the lowest counter, victimize the page at the lowest page in the memory. That is, if pages at mem[5] and mem[8] both have lowest counter value, replace the page at mem[5].

Total Memory Size in Page

- A positive non-zero number
- Indicates the total number of pages that can exist in memory at the same time

Input file

- A text file
- Each line in this file contains two numbers:
  - a page number
  - 0: page is read by CPU 1: page is written by CPU
- Page number > 0
- Example:

  7 0 ← page 7 is read

  1 0 ← page 1 is read

  9 1 ← page 9 is written

Simplifying Assumptions

- The page table contains only the pages in the physical memory. This makes your life easier to pick the page to replace.
- The counter in the aging algorithm is only 8 bits.
- The page number will be [1, 20]

Behavior and Stats

- The output file is generated automatically. You don’t need to write code for it.
- The output file has the same name as the input with an extra extension .nru or .aging
- The first part of the file contains the memory content (per line) after each memory access.
- The second part of the file contains stats.

Example: NRU

Memory can contain three physical pages

Memory Initially: 0 0 0

Number of accesses: 5

Number of page faults: 4

Input file

1 0

2 1

3 0

1 0

4 0

Page table:

Page#	R	M
1	1	0

Page table:

Page#	R	M
1	1	0
2	0	1

Page table:

Page#	R	M
1	1	0
2	0	1
3	1	0

Page table:

Page#	R	M
1	1	0
2	0	1
3	1	0

Page table:

Page#	R	M
1	1	0
4	1	0
3	1	0

Example: Aging

Memory can contain three physical pages
Memory Initially: 0 0 0

Number of accesses: 5
Number of page faults: 4

Input file
1 0
2 1
3 0
1 0
4 0

Page table:

Page#	R	Counter
1	1	10000000

Page table:

Page#	R	Counter
1	0	11000000
2	1	10000000

Page table:

Page#	R	Counter
1	0	01100000
2	0	11000000
3	1	10000000

Page table:

Page#	R	Counter
1	0	01110000
4	1	10000000
3	0	01100000

Page table:

Page#	R	Counter
1	1	10110000
2	0	01100000
3	0	11000000

To Help You

- You are given a file paging.c that reads the command line arguments and loops over the memory accesses, reads the input file, create the output file, and puts the needed information in the output file.
- You need to write four functions:
  - nru()
  - nru_pt_update()
  - aging()
  - aging_pt_update()
- When you compile paging.c before adding your code, you may get some warnings but there are empty files (the ones you need to implement) that are suppose to return an int (the page number that will be replaced).
- We are providing you with an executable ./ref to generate the model answer.
  - For example: `./ref 1 3 in1` will generate a file in1.nru showing the memory and stats giving the page accesses in file in1 and using NRU replacement.
  - Note: before using ./ref you may want to type `chmod 777 ./ref`
- Compile with: `gcc –o paging –Wall –std=c99 paging.c`

The Data Structure

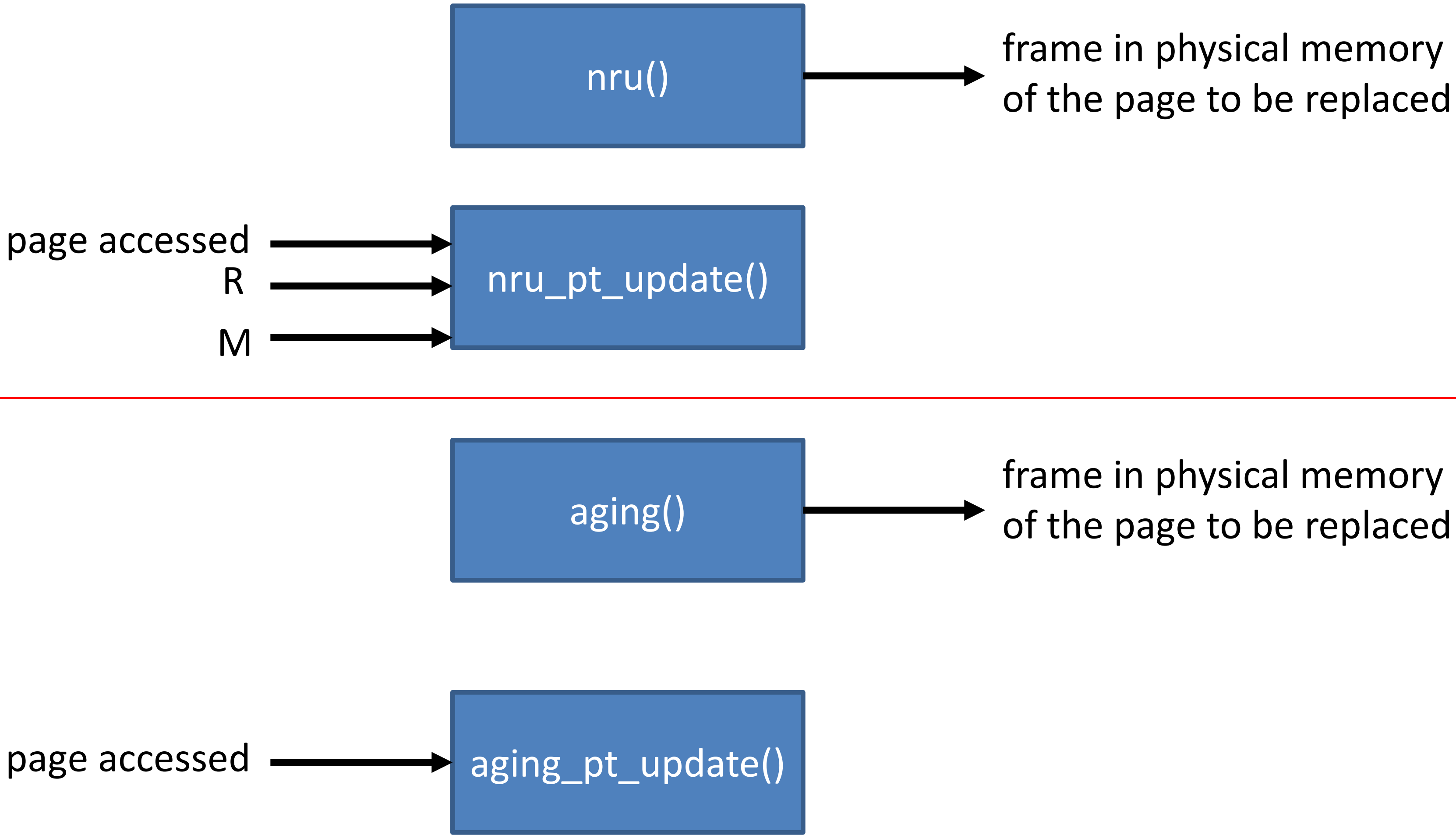
The Physical Memory
int mem[] ← has mem_size entries
mem[0] = x
mem[1] = y ← page y
.
.
.
mem[mem_size-1] = k

Page Table
Array of structures of mem_size entries

Array name: **page_table**

int R	int M	unsigned char counter
…	…	…

What you need to implement:


nru()
frame in physical memory of the page to be replaced
page accessed
R
M
nru_pt_update()
aging()
frame in physical memory of the page to be replaced
page accessed
aging_pt_update()

What To Submit:
Through Brightspace

- paging.c