Assignment 7

Georgiy Krylov

November 17, 2023

ASSIGNMENT IS TO BE COMPLETED INDIVIDUALLY BY ALL STUDENTS!

1 Description

This assignment is to develop a simulation of LRU page replacement algorithm with swapping support. **The assignment is Due by 11:59 p.m. on Thursday, the 23rd of November 2023 (one minute before Friday).**

2 Task

You are to simulate the Least Recently Used algorithm for page replacement. Design choices:

1. The machine's architecture is 32-bit.
2. You are to implement one level page tables (page number, and offset)
3. the page size is 4KB (2^{12}), therefore the frame size is also 4 KB.

Reminder: The most significant twenty bits of a virtual address (an **unsigned** 32-bit integer) denote page number, the last 12 bits are for the offset within the page table.

Your program will read the address requests and the operations that are performed on the data in those addresses. Your program should report the physical addresses that were assigned to the addresses.

This time, however, the number of the frames assigned to your process is limited, and should be read as a command line parameter. If there are not enough free frames in the main memory to service a request, your program needs to determine a "victim" page to be paged out from the memory. This should be done using the Least Recently Used page replacement algorithm. If the victim page is "dirty", i.e., the page contents were written to since the last time the page was loaded (or swapped in), the page needs to be copied into the swap space (swapped out). For simplicity, we say that once a page was swapped out, it always counts towards the number swapped pages, even if it was later

swapped in again. If the victim page is "clean", i.e., its contents are identical to the page in the long term storage, it can be discarded. I.e. if the page was only ever accessed for reading, we assume it surfaced from the executable file or the program is faulty, we do not swap it out.

For simplicity, you may assume that loading a page into a frame is the same as swapping it in: i.e., for swap space management you only need to count the number of times you had to page out a page with "dirty" bit set to true, no need to simulate anything further.

We are not dealing with the minor page faults in this assignment.

At the end of the program, you are supposed to report the number of pages swapped out over the course of execution, the number of page hits, the total number of page faults, and the average access time using the formula we learned in class:

$$\text{probabilityOfNoPageFault} * 10 + \text{probabilityOfMajorPageFaultWithOne Copy} * 1000 + \text{probabilityOfMajorPageFaultWithTwoCopyOperations} * 3000$$

3 Input/Output format

Consider the following input format:

```
r 1124955998
r 794845611
r 1540976830
w 1081961846
r 1124957700
r 794845440
r 1540976645
```

This input should be interpreted as follows: The program tries to access the specified addresses.**r** denotes "read from", **w** denotes "write to". The program should print the following:

```
Logical addresses -> Physical addresses:
1124955998 -> 1886
794845611 -> 4523
1540976830 -> 8382
1081961846 -> 3446
1124957700 -> 7684
794845440 -> 8448
1540976645 -> 5

Stats:
major page faults = 7
page hits = 0
pages swapped out = 1
Effective Access Time = 1285.7144
```

Sample program execution

```
./a.out 3 < fromText.In
```

(Note the sample output had 3 frames). For the exact input and output format please refer to the "a7-tests.zip".

4 Submission instructions

Please submit just your C and H files (if any) to D2L Assignment box. Passing all the tests within the Makefile (in a solution that is not hard-coded) will grant you a full grade. Make sure your code compiles and runs. Make sure your code follows the specified input/output format (once the sample files are posted). You must use C programming language to solve this assignment.

NOTE: THE INPUT AND OUTPUT OF YOUR PROGRAM IS SPECIFIED SUCH THAT THE MARKER CAN AUTO TEST YOUR SUBMISSIONS.