

OS LAB-5

Mruganshi Gohel

B20CS014

How to run?

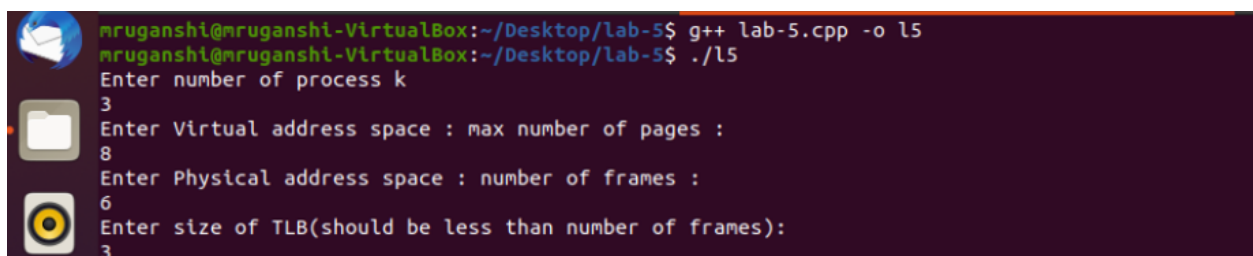
type the following commands in the terminal.

```
g++ lab-5.cpp -o l5
./l5
```

Inputs

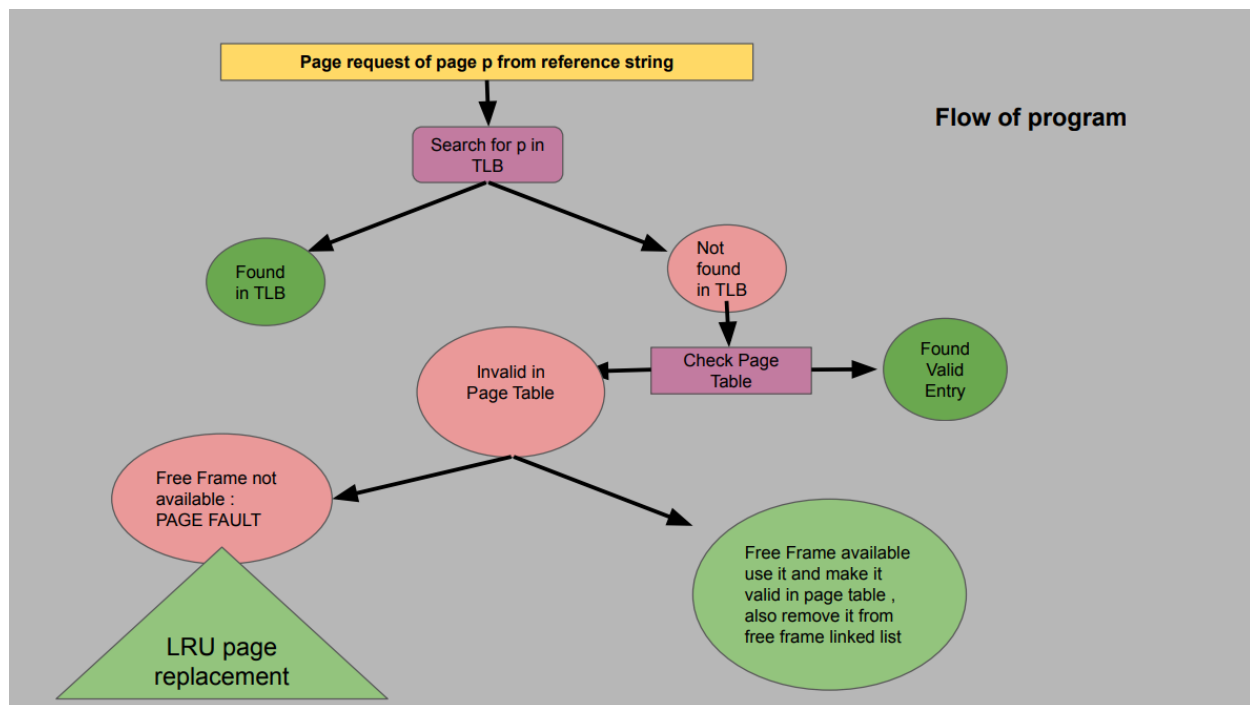
- Enter the number of processes
- Enter virtual address space (maximum number of pages required per process) (m)
- Enter the physical address space (total number of frames in main memory) (f) ($m > f$)
- Enter size of TLB (s) ($s < f$)

as shown in the below screenshot



```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-5$ g++ lab-5.cpp -o l5
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-5$ ./l5
Enter number of process k
3
Enter Virtual address space : max number of pages :
8
Enter Physical address space : number of frames :
6
Enter size of TLB(should be less than number of frames):
3
```

The flow of the program



| Component | Data Structure used |
|------------------------|--|
| TLB | Vector of pairs - {page number, frame number} |
| Page Table | Vector of pairs - {frame number, validity bit} |
| Free Frame list | Linked List |
| Frame-Page link record | Struct FP |

Process

first, we will take input for the number of processes. and through the map, I will keep track of the processes which have already run.

then we will take the other three inputs (m, f, s). after that I will initialize the reference string size twice the number of pages it requires.

I will create a reference string. initially, no page will be linked to the free frame map and initialize the free frame linked list of full size.

then I will create a page table in which all frames will be invalid initially except TLB. then I will send reference string TLB ffill and page table for processing.

after that will refresh everything for the new process.

- in processRS() function there will be 2 cases:

Case-1 :

```
if size of freeFrameLinkedList is not 0 yet
    cout<<"TLB Miss with no page fault , free frame is available\n";
    insert requested page : page_num into struct FP at corresponding next_free_frame_num :
        next_free_frame_num = freeFrameLinkedList[0];
    FP[next_free_frame_num].page_linked = page_num
    and make validity of that next_free_frame_num in page_table to 1 :
        page_table[next_free_frame_num].second = 1
    Also delete that next_free_frame_num from freeFrameLinkedList since
        its not free anymore
    it = freeFrameLinkedList.begin() ;
    freeFrameLinkedList.erase(it);
```

Case-2 :

```
else if size of freeFrameLinkedList is 0 now , Page Fault !!!!!
    cout<<"TLB Miss : Page Fault ...Performing Page Replacement with LRU\n";
    LRU ---> least ref page number ----> VictimFrame
        ( get this from page table since it must be full now)
    Fp[VictimFrame].page_linked = page_num //new ref page
    page_table[VictimFrame ] --> must be already valid 1
```

- in updateLRU() function

if we found a required page at the head of the linked list then do nothing else if we found it at the tail then we will change the tail to the previous element it will be changed only if the previous tail is not null.

if they exist then we connect previously to the next and vice versa and make the current element a new head.

Demand Paging Implementation Correctness

Frames are being brought only when they are being used. Initially, all of them are invalid and are present in the free frame linked list.

During random TLB assignment as mentioned in the question, the assigned frames are removed and made valid in the page table and linked with the corresponding page

number.

On subsequent TLB miss, the pages are being linked to pages as per the algorithm diagram shown in previous slides.

Hence, frames are linked to demand, and demand paging is implemented.

Output on terminal

The complete output is shown in result.txt in the submitted folder.

```
6
Enter size of TLB(should be less than number of frames):
3
Current process is: 2
Current process requires 6 pages
Reference String Generated : 2 0 4 0 0 1 1 5 2 0 4 6
Free Frames : 3 4 5
TLB : page - frame
2 - 0
4 - 1
6 - 2

Page Table : Page number - Validity (Only frames chosen for TLB are valid)
0 - 1
1 - 1
2 - 1
3 - 0
4 - 0
5 - 0
Starting Process number : 0
Process : 2 for page reference 2 , TLB hit with frame no. 0
Process : 2 for page reference 0 , TLB miss -> Now check page_table
TLB Miss with no page fault , free frame is available
Used the new frame 3
added 0 Current state of LRU:
0
Process : 2 for page reference 4 , TLB hit with frame no. 1
Process : 2 for page reference 0 , TLB miss -> Now check page_table
TLB Miss with no page fault
Process : 2 for page reference 0 , TLB miss -> page table valid -> with frame no. 3
Process : 2 for page reference 0 , TLB miss -> Now check page_table
TLB Miss with no page fault
Process : 2 for page reference 0 , TLB miss -> page table valid -> with frame no. 3
Process : 2 for page reference 1 , TLB miss -> Now check page_table
TLB Miss with no page fault , free frame is available
Used the new frame 4
added 1 Current state of LRU:
1 0
```