OPERATING SYSTEM LAB 5
Abu Shahid
B20CS003

DEMAND PAGING FOR VIRTUAL MEMORY

Assignment Description

In this lab, we had to write code to implement demand paging for virtual memory.

Execution

`g++ page_demand.cpp -o pg`

`./pd`

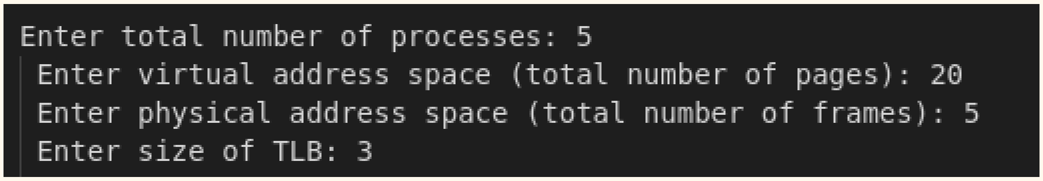
- After execution, the program expects 5 inputs
  - num_processes: number of processes
  - vir_addr_space: total number of pages
  - phy_addr_space: total number of frames
  - TLB_size: size of TLB
    - Here vir_addr_space > phy_addr_space > TLB_size
- The program maintains
  - terminated_processes: array of processes that have finished execution
  - total_page_faults, total_page_hits, total_tlb_hits
- Program will randomly pick a process if it has not been executed yet and generate its pid
  - For our implementation, process PID= serial number associated with the process
  - Then pages_in_process PID is evaluated;
  - At start of each process, we also initialize list of free frames as initially all frames are available.
  - TLB is initialized as random pages are loaded in the TLB
  - Page_table is also initialized

    - Reference string is generated whose length can vary from 2*V to 10*V where V is size of the virtual address space.
- Each page in the reference list is then iterated
    - If it is in TLB, TLB hit is issued and the page is pushed into the LRU queue.
    - If it is not in TLB but in page_table, page_hit is issued and the corresponding frame is freed, but the page is loaded in the TLB
    - If it is not in the page table and TLB, page fault is issued;
        - If free frame is available, it is allocated to the page and the page is also loaded in the TLB
        - Else victim page is found and is ejected from both TLB and page_table
        - The frame freed thus is assigned to the incoming page required and the page is also loaded in the TLB
- If at any point TLB miss occurs and the new page cannot be accommodated in the TLB, least recently used page is ejected.

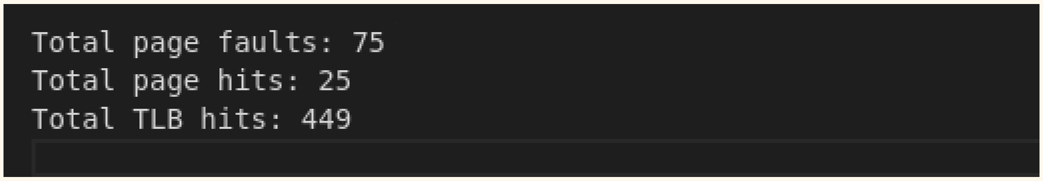
Analysis and results

The page demand program was run for various parameters which can be summarized before:

Case1:



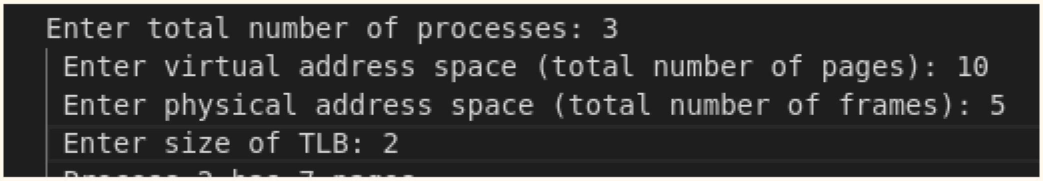
```
Enter total number of processes: 5
 Enter virtual address space (total number of pages): 20
 Enter physical address space (total number of frames): 5
 Enter size of TLB: 3
```



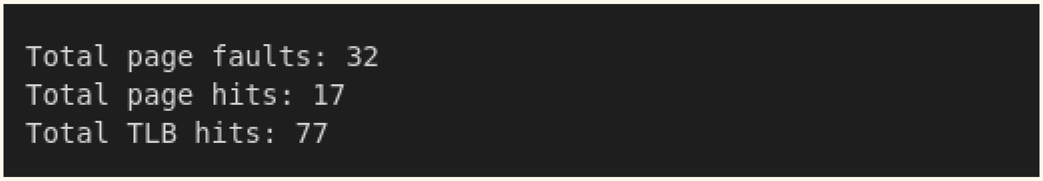
```
Total page faults: 75
Total page hits: 25
Total TLB hits: 449
```

For details of the execution of each process, please refer to output1.txt

Case2:



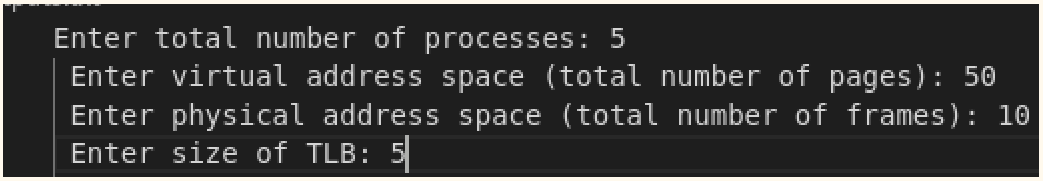
```
 Enter total number of processes: 3
  Enter virtual address space (total number of pages): 10
  Enter physical address space (total number of frames): 5
  Enter size of TLB: 2
```



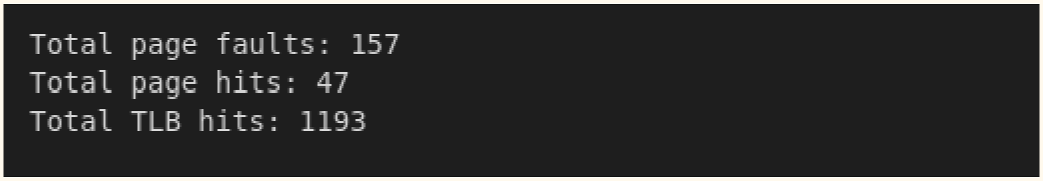
```
Total page faults: 32
Total page hits: 17
Total TLB hits: 77
```

For details of each process, please refer to output2.txt

Case3:



```
 Enter total number of processes: 5
  Enter virtual address space (total number of pages): 50
  Enter physical address space (total number of frames): 10
  Enter size of TLB: 5
```



```
Total page faults: 157
Total page hits: 47
Total TLB hits: 1193
```

For details of each process, please refer to output3.txt

- Clearly higher TLB size and virtual address size lead to lower page faults.