CS6600 Computer Architecture (Jul-Nov 2021)
Assignment-3

Karthikeyan R EE18B015
Nithin Babu EE18B021

MMU Simulator

Steps followed

- The following are the essential variables required for our MMU simulator:
    - free_page_frame = [ ] # Free page frame list
    - dirty_page = [ ] # Dirty page list
    - lru_counter = [ ] # 4-bit saturation LRU counter for pages
    - p1 = {} # Dictionary for paging logic - Process 1
    - p2 = {} # Dictionary for paging logic - Process 2
    - page_hit # Counter for any page hits
    - page_miss # Counter for any page misses
    - read_req # Counter for read requests
    - write_req # Counter for write requests
    - p1_req # Counter for Process 1 requests
    - p2_req # Counter for Process 2 requests
    - dirty_page_evictions # Counter for any dirty page evictions

- There is also a function that is initialized, *lru_update(page_frame)*, which is used for the LRU policy .This function takes a page frame as input, and increments all the mapped page-frames, except for the input *page_frame,* in the *lru_counter* list by 1.

- The procedure followed for MMU simulator is as follows:
    - The free page-frame list is updated with the entries of all the page frames. Here, the dirty page list as well as lru counter list is initialized with 0 for all page frames.
    - The input file containing the list of instructions is accessed and read one by one in a loop.
    - The page directory, page table, mode (r/w) and process ID are collected.
    - If in read mode, then *read_req* is incremented by 1, else *write_req* is incremented.
    - The programs check if process ID is either 1 or 2, and the same steps are followed for both processes, with the dictionary being p1 or p2 respectively.
    - If the p1/p2 dictionary already contains data for the virtual address provided, then it is a page hit, and *page_hit* is incremented by 1. If mode is ‘w’, then the corresponding page frame in the *dirty_page* list is made 1. After this, the *lru_update()* is called.
    - If the p1/p2 dictionary does not contain any data for the virtual address provided, then it is a page miss, and the *page_miss* is incremented by 1. If there is a free page in the free page-frame list, then that page frame is mapped to the virtual address and stored in the p1/p2 dictionary. Again, the *dirty_page* list is updated if mode is ‘w’ which is followed by calling *lru_update()*.

- If there are no free pages in the free page-frame list, then the page frame with the maximum value in the corresponding *lru_counter* list is selected as a victim page. This page frame is then mapped to the current address, and it's corresponding *dirty_page*, *lru_counter* list elements are reset to 0. Whenever a dirty page frame is to be evicted, then *dirty_page_evictions* is incremented by 1. Again, the *dirty_page* list is updated if mode is 'w' which is followed by calling *lru_update()*.

Output

The output obtained for the given 'input.txt' file is as follows:

- Total requests: 36
- Page miss rate: 58.33%
- Page hit rate: 41.67%
- Read requests: 18
- Write requests: 18
- PID: 1: 26 requests
- PID: 2: 10 requests
- Dirty page evictions: 0
- Space for paging logic: 84 bytes

Submission

Please find the MMU simulator mmusim.py here.
