Assignment 4

Please complete a report in English in English in English in English and upload the corresponding codes.

The files should be uploaded directly without compression without compression without compression without compression

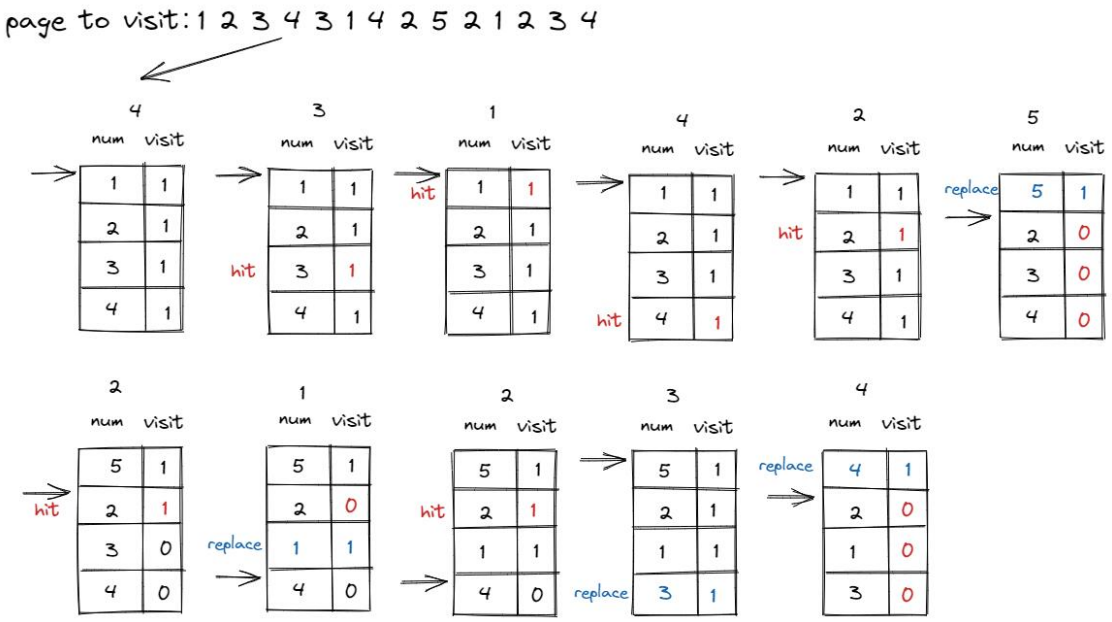
The files to be submitted for this assignment are:

1. report.pdf
2. swap_clock.c
3. swap_lru.c

---

1. **[20pts]** Read Chapter 21 of "Three Easy Pieces" (https://pages.cs.wisc.edu/~remzi/OSTEP/vm-beyondphys.pdf) and explain what happens when the process accesses a memory page not present in the physical memory.
2. **[20pts]** Consider the following reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1. Consider 4 pages. What are the number of page faults with the following policy: Optimal (MIN), LRU, FIFO.
3. **[30pts]** Realize Clock algorithm in `swap_clock.c`

   Clock： Clock algorithm organizes pages into a circular linked list, just like a clock. The pointer points to the pages loaded the earliest. Besides, Clock algorithm requires a flag bit in PTE to indicate if the corresponding page is used. When the page is used, MMU in CPU will set the flag to 1. When system needs to replace a page, system read the PTE pointer points to, and replace it if the flag is 0, otherwise read the next PTE. The algorithm demonstrates the idea of LRU, and is easy to implement and costs less, but require hardware support to set an reference bit. Clock algorithm is the same as LRU essentially, but it skips pages with reference bit of 1.



   Code you may need：

```
list_entry_t *curr_ptr;

*ptr_page = le2page(le, pra_page_link);
pte_t* ptep = get_pte(mm->pgdir, ptr_page->pra_vaddr, 0);
bool accessed = *pte & PTE_A;
```

Please realize algorithm in `swap_clock.c`. Change `sm` in `swap.c` to clock to test your code.

Please take screen-shot of your code(with annotations) and the running result.

4. **[30pts]** Realize LRU algorithm in `swap_lru.c`

   LRU (Least Recently Used) Page Replacement Algorithm: Using locality, to predict the future access according to previous access, we suppose that recently accessed pages are more likely to be accessed in the future, and pages not accessed for long time are less likely to be accessed. Therefore by comparing last access time of each page, the page with the longest last access time is found and replaced.

   Here we implement a simulated strategy. We set a constant. Every time we visit a virtual address, we will write this constant to the page and then +1. LRU algorithm is realized through this constant.

   Code you may need：

```
int temp = *(unsigned char *)tmp_page->pra_vaddr; //get the constant in page
```

Please realize algorithm in `swap_lru.c`. Change `sm` in `swap.c` to LRU to test your code.

Please take screen-shot of your code(with annotations) and the running result.