COMPE571 – Embedded Operating Systems – Fall 2025 – Programming Assignment 4

NOTE: You should form groups of 2 or 3 for this assignment.

Assignment Description

In this assignment, you will develop a virtual memory simulation tool and do some experiments with it. To complete this assignment, you can use any programming language you want.

Your program will read an input file (you are given 2 sample files in the assignment description: data1.txt and data2.txt). The input file consists of a long series of memory references. Each line of the file has three values, delimited by a space. The first field is an integer, showing the process number. The second value is also an integer, showing a memory reference (address). And the third is a char, either "R" (Read) or "W" (Write). If this char is a "W", it means that the referenced memory address is written to, i.e., modified.

You can assume that each memory reference happens in one time unit.

All addresses are 16 bits; so, all processes have a virtual address space of 64KB (2^{16} bytes). All addresses read from the file will be in the range 0 ... 65535. The system uses 1-level page tables, where each virtual address is organized as (virtual page number, offset). Each page is 512 (2^9) bytes, so each process has a page table with 128 ($2^{16}/2^9$) entries. In other words, in each virtual address, the 7 most significant bits determine the virtual page number, and the remaining 9 bits determine the offset of the address within the page frame.

Each page table entry should also have a dirty bit and a reference bit. You may want to include other data as well to help with your implementation. Please keep in mind that page table entries keep address translations, as well as auxiliary information about each page, such as whether the page has been written (using the dirty bit) or whether the page has been recently referenced (using the reference bit).

There are 32 addressable physical pages in the main memory, i.e., the physical memory size is 32pages*512bytes/page = 16KB.

In your simulation, you have to keep track of three statistics: the total number of page faults, the total number of disk references, and the total number of dirty page writes. Every page fault has at least one disk reference, but if the page to be replaced is dirty, there will be two disk references, one to copy the dirty page back to the disk and one to load the new page.

Each process has its own page table, so address 12340 in process 1 is different from address 12340 in process 2.

You should implement and analyze the following page replacement algorithms:

- Random (RAND): The victim page (i.e., the page to be replaced) is selected in random.
- First in first out (FIFO): The oldest page in the memory gets replaced.
- Least recently used (LRU): The least recently used page in the memory gets replaced. If there are two pages with the same value (the last used time unit), replace the one that is

not dirty. If both pages are or neither page is dirty, replace the lower numbered page.

- Periodic reference reset (PER): Whenever a page is referenced, its reference bit is set to 1. After every 200 memory references, your program should set all of the reference bits to 0. When a page fault occurs, choose the page to be replaced in the following order:
    - Look for an unused page (this will only happen early in your simulation).
    - Look for an unreferenced page (reference bit is 0) where the dirty bit is 0.
    - Look for an unreferenced page (reference bit is 0) where the dirty bit is 1.
    - Look for a referenced page (reference bit is 1) where the dirty bit is 0.
    - Look for a page that is both referenced (reference bit is 1) and dirty (dirty bit is 1).
    - In order to make the results deterministic, always replace the lowest numbered page in a particular category.

25% Additional Credit: If you create your own page replacement algorithm and can show that your algorithm can outperform the above four in terms of number of page faults, number of dirty page writes, and number of disk accesses.

Report

Your report should include the following:

- Detailed explanation of your simulation code
    - How you are representing page tables
    - How you are making virtual to physical address translation
    - How you are determining when there is a need to replace a page
    - How you are determining the replaced page
    - Explanation of your own replacement algorithm (if applicable)
- Results
    - Following statistics for each replacement algorithm
        - Total number of page faults
        - Total number of disk accesses
        - Total number of dirty page writes
    - Three graphs comparing the above variables for different replacement algorithms (one graph for each variable)

Deliverables

Your assignment submission must include **TWO** files:

- A zip file containing functional program(s)/script(s) (Note: you can use any programming language you want. Just make sure that your zip file includes a README file showing how to execute your program/script).
- A project report containing your implementation details and discussing your results. Note that you should include the main results in the report as well (as described above). Please submit your report in DOCX or PDF (preferred) format.

Your final score will be determined by the combination of your report and turning in a functional script on time. **Note that the report should be at most 4 pages long. If you are including extra credit results, your report can be at most 5 pages.**