Mirza Imran Biag - 1002086480

Mohammad Taqiuddin-1002082581

**PROJECT 1 REPORT**

**Policy 1 – Random:**

**pickVictim():**

For the random page replacement policy, we have a loop in the pickVictim() function that iterates until the buffer capacity is reached. The frame from the buffer is chosen at random from the frametab for each repetition. This approach does not take advantage of the usage history of frames, in contrast to certain other page replacement methods. Therefore, frames are chosen regardless of whether they have been recently used or pinned. Until the buffer is filled, this random selection keeps happening. If we encounter a unpinned frame, then we make it state as AVAILABLE. Notably, since the selection is completely random in a truly random policy, there is no requirement to look for unpinned frames.

**Policy 2 – Fifo:**

**pickVictim():**

A loop is used to fill the buffer up to its limit within the pickVictim() method for the FIFO page replacement policy. As frames are found during the loop, they are appended to the frametab's end. These recently added frames are effectively "pinned" as they are added, maintaining their location in memory. The function searches for frames that are unpinned, or eligible for replacement, after the buffer is full. We have another loop which is used to control page replacement once an unpinned frame has been located. The FIFO strategy makes sure that when a page needs to be replaced, the oldest one in memory—i.e., the one that was initially brought in—is chosen for replacement.

**BHR – Buffer to Hit Ratio:**

In this task, we are tasked with determining the Buffer Hit Ratio (BHR) for various page replacement policies using different buffer sizes. The BHR is calculated using a specific formula: it's the ratio of the total number of Page Hits to the total number of Page Requests, resulting in the aggregate BHR.

**Page Hit Definition:** A Page Hit occurs when an attempt is made to load a page into the buffer, and it is already present in the buffer. In simpler terms, it's when the desired page is found in the buffer.

**Page Fault Definition:** A Page Fault takes place when an attempt is made to load a page into the buffer, but the requested page is not present in the buffer. Essentially, it's when the buffer lacks the page that's being requested.

**File description:**

We have four directories in total two for random replacement policy and two for Fifo replacement policy.

Directory 1: FIFO\_22 This directory has the buffer replacement policy and code implementation with the buffer\_size=22

Directory 2: FIFO\_32 This directory has the buffer replacement policy and code implementation with the buffer\_size=32

Directory 3: Random\_22 This directory has the buffer replacement policy and code implementation with the buffer\_size=22

Directory 4: Random\_32 This directory has the buffer replacement policy and code implementation with the buffer\_size=32

**Division of labor:**

We distributed the work in such a manner that we have to work on each policy’s and at the end work on the (BHR) buffer hit ratio code as a group.

Random replacement policy is covered by Mirza Imran Baig (1002086480) and

Fifo is covered by Mohammed Taqiuddin (1002082581)

We worked on the project for a period of two weeks, spending roughly three to four hours per day on it. Getting together each day to work on different activities connected to putting various policies into effect was part of our daily routine. In the end, as a group, we put the BHR code into action successfully.

**Report of the analysis on buffer sizes with different policy’s**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Policy/Test**   |  | | --- | | *Buffer Size* | | **RANDOM**   |  |  | | --- | --- | | 22 | 32 | | **FIFO**   |  |  | | --- | --- | | 22 | 32 | |
| |  | | --- | | *Test 1* | | *Test 2* | | *Test 3* |   **BM Test** | |  |  | | --- | --- | | 0.18099 | 0.22171 | | 0.00990 | 0.00990 | | 0.19090 | 0.20000 | | |  |  | | --- | --- | | 0.00452 | 0.00452 | | 0.00990 | 0.00990 | | 0.02727 | 0.02727 | |
| |  | | --- | | *Test 1* | | *Test 2* | | *Test 3* | | *Test 4* |   **BHR Test** | |  |  | | --- | --- | | 0.00753 | 0.00509 | | 0.72456 | 0.72474 | | 0.72639 | 0.72622 | | 0.87234 | 0.87234 | | |  |  | | --- | --- | | 0.00000 | 0.00000 | | 0.72383 | 0.72383 | | 0.72551 | 0.72551 | | 0.87234 | 0.87234 | |

**Analysis report of policy with respect to the buffer hit ratios**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test/policy | Random   |  |  | | --- | --- | | PAGE NUMBER | REFERENCES | | Fifo   |  |  | | --- | --- | | PAGE NUMBER | REFERENCE | |
| Test 1 | |  |  | | --- | --- | | 508 | 50 | | 507 | 50 | | 506 | 50 | | 505 | 50 | | 504 | 50 | | |  |  | | --- | --- | | 508 | 50 | | 507 | 50 | | 506 | 50 | | 504 | 50 | | 503 | 50 | |
| Test 2 | |  |  | | --- | --- | | 46 | 22 | | 463 | 20 | | 137 | 20 | | 466 | 19 | | 181 | 19 | | |  |  | | --- | --- | | 46 | 19 | | 463 | 17 | | 137 | 17 | | 466 | 16 | | 181 | 16 | |
| Test 3 | |  |  | | --- | --- | | 191 | 23 | | 46 | 22 | | 172 | 22 | | 385 | 21 | | 110 | 21 | | |  |  | | --- | --- | | 191 | 19 | | 46 | 19 | | 172 | 18 | | 385 | 17 | | 110 | 17 | |
| Test 4 | |  |  | | --- | --- | | 70 | 15 | | 9 | 14 | | 21 | 13 | | 33 | 12 | | 45 | 11 | | |  |  | | --- | --- | | 70 | 14 | | 9 | 13 | | 21 | 12 | | 33 | 11 | | 45 | 10 | |

**Logical errors**

Error 1: A problem with the random number generator came up as I was putting the random replacement policy into practice. In particular, I had trouble stopping the generator from constantly producing the same number in later cycles after getting the first random number. This problem caused the random value generator to keep producing the same numbers over and over again.

Error 2: During program execution, we encountered two persistent issues. Firstly, consistent test results producing identical page numbers were caused by a bug, prompting us to use print statements for error identification and resolution. Secondly, as we neared the completion of our code, 'page not present' errors surfaced. Initially attributing this to the skeleton code, we utilized 'print statements' for debugging and ultimately identified the root cause: the 'nframes' variable had not been properly incremented. Once we addressed this omission, our code ran smoothly, successfully resolving the previously perplexing issue.

Error 3: We encountered a problem while trying to count and sort how often the top 'k' pages were referenced, and this issue was caused by a flaw in our sorting method. To fully address this problem, we opted to completely revamp the sorting code. This overhaul entailed a meticulous process of reviewing and testing to ensure that the revised sorting code not only fixed the error but also functioned consistently and with precision.