Page Replacement Algorithms:

Design of Simulator: Before diving into the logic of page replacement algorithms, a quick overview of all the java files.

PageTableEntry.java

This java file contains all the entries needed for a single page table entry. Each page table entry stores five properties that are Physical Frame Number (PFN), Virtual Page Number (VPN), Referenced timestamp, Modified timestamp and the Page Load Time (timestamp when the page was first loaded into the page table).

PageTable.java

This java file stores the list of all the page table entries and it also stores the global timer and last clock tick time. This java file also performs operations like add a page table entry into a page table, modifying the page table entry and updating the clock tick time. This file is responsible for writing and modifying the Referenced, Modified, PFN, VPN and load time variables. The global timer variable is incremented on every page request and is used to update all the variables.

PageStatistics.java

This java file computes the hit ratio and miss ratio of the page replacement algorithms.

PageReplacementAlgorithms.java

This java file contains the implementation of all the page replacement algorithms. I have implemented all the three page replacement algorithms and their implementation details are given below. The logic of each page replacement algorithm is given below.

FIFO:

The First In First Out (FIFO) algorithm evicts the oldest page from the page table.

The FIFO algorithm first retrieves all the page table entries from the page table and stores them in a list. It then declares two temporary variables selectedPte and lowLoadTime. The selectedPte, initially, holds the first page table entry from the list and the lowLoadTime, initially, holds the load time of the selectedPte variable. Then the program iterates through the remaining page table entries in the list and compares the load time of every page table entry with the lowLoadTime variable. If the load time of any page table entry is less than the lowLoadTime variable, then the selectedPte variable and lowLoadTime variable are updated accordingly. After the iteration is complete, it returns the selectedPte (which is the oldest page) variable.

LRU:

The Least Recently Used (LRU) algorithm evicts the page that was used in the distant past. The LRU algorithm first retrieves all the page table entries from the page table and stores them in a list. It then declares two temporary variables selectedPte and lowReference. The selectedPte, initially, holds the first page table entry from the list and the lowReference, initially, holds the referenced time of the selectedPte variable. Then the program iterates through the remaining page table entries in the list and compares the referenced time of every page table entry with the lowReference variable. If the referenced time of any page table entry is less than the lowReference variable, then the selectedPte variable and lowLoadTime variable are updated accordingly. After the iteration is complete, it returns the selectedPte (which is the oldest page) variable.

Clock:

The Clock algorithm evicts the page that was not used from the previous clock tick. The clock algorithm first retrieves all the page table entries from the page table and stores them in a list. It then declares a temporary variable selectedPte. The algorithm also maintains the current location of the clock using a variable clockLocation. Whenever a page needs to be evicted, the clock algorithm starts searching the page table entries from the current clock location. The program then declares a do-while loop to search for a page table entry that has not been referenced since the last clock tick. If it finds any, it updates the selectedPte variable and breaks the while loop and returns the selectedPte variable.

PageReplacementTest.java

This java file parses the sample input.txt file, reads each line from the file and invokes the corresponding page replacement algorithm and prints the statistics at the end of the program.

Output Comparison:

In order to compare all the page replacement algorithms, I have used a metric called hit ratio which is defined as the ratio to the number of page hits to the total number of page requests in a given input.

Number of page hits

Total number of page requests

Hit Ratio =

The sample input.txt that will be used for this project has 99 page requests. For this project analysis, I have taken the page frame size to be 16. For each of the three page replacement algorithms, we try to analyze the number of page hits and calculate the hit ratio for the given input file.

FIFO Algorithm:

Number of requests 99

Number of hits 24

Hit Ratio 0.243

LRU Algorithm:

Number of requests 99

Number of hits 38

Hit Ratio 0.384

CLOCK Algorithm:

Number of requests 99

Number of hits 19

Hit Ratio 0.192

Listing all the three page replacement algorithms hits, misses and hit ratio in the form of a table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Number of Pages | Number of Hits | Number of Misses | Hit Ratio |
| FIFO | 99 | 24 | 75 | 0.243 |
| LRU | 99 | 38 | 61 | 0.384 |
| CLOCK | 99 | 19 | 80 | 0.192 |

For the given input, taking a look at the hit ratio, LRU algorithm seems to be the best followed by the FIFO algorithm and then the CLOCK algorithm.

Though we have noticed the above results for this project, the efficiency of a page replacement algorithm could vary if we choose a different input sequence or if we choose a different number of page frames.