Abhi Munnangi

Friday, May 25. 2018

Assignment 5

The Purpose of this lab is to get familiarized with the Page replacement Algorithms

Here we investigated the FIFO,SC, and LRU

FIFO stands for first in first out Algorithm

Once the page table becomes full we remove the first page that entered the page once a page that is not inside the page table in inputted. This algorithm is repeated if a new page value is entered.

Implementation for this was very simple. I used a circular array with a headindex. This head index keeps track of the most recently used index. Once a page needs to be replaced , the value is replaced and the head index is incremented, at the location of the page to be replaced next.

SC stands for Second Chance

This is like FIFO except if the page is used at least once, the value will be sent to the back of the queue instead of being removed.

In order to implement this, I created a second array called Chance Table which is the same size as the page table. This chance table keeps track of the number of times each value is used in the page table, using corresponding index to match the values. So when a new page value is inputted into the table, page Table is traversed, but every value of the Chance table is checked. If the value at the index is greater than 1, then it gets reset to 0 and the program continues to traverse. If the value if 0, then that is the page that will be replaced. Also, the Value in Chance table, will be set to 0 to accommodate the new value.

If the page value is already in the table, the value in ChanceTable, corresponding to the index in page table, will be incremented by 1.

LRU stands for Least Recently Used

Whenever a page is used, it is placed in the back of the queue, indicating that is the most recently used. Whenever a page needs to be placed, the value on front of the queue will be deleted and the page inserted will be put into the page of the queue.

In order for this algorithm to work. I maintained an array.

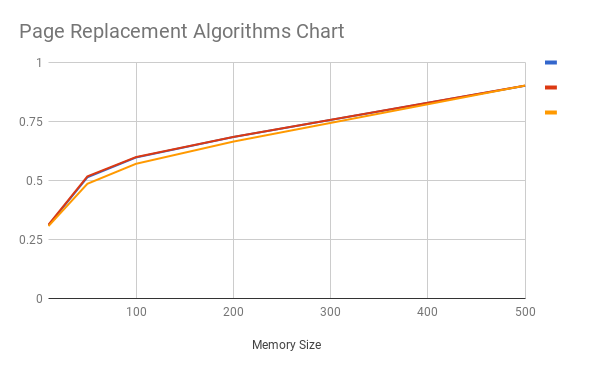
If a page that is already in the table is inserted, then I shift the array to the left and place the value page value at the end. This was done with a simple for loop.

If the page is not in the table, then the array will be shifted to the left and the page inserted will be placed in the back of the queue. This was also done with a simple for loop.

I tested and graphed certain values

I got this Data Testing the Values

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | LRU | SC | FIFO |
| 10 | 0.31015 | 0.312724 | 0.306953 |
| 50 | 0.513629 | 0.517429 | 0.486111 |
| 100 | 0.597563 | 0.599303 | 0.570789 |
| 200 | 0.684208 | 0.684509 | 0.664779 |
| 500 | 0.901275 | 0.90199 | 0.902122 |



X AXIS TABLE/Memory SIZE

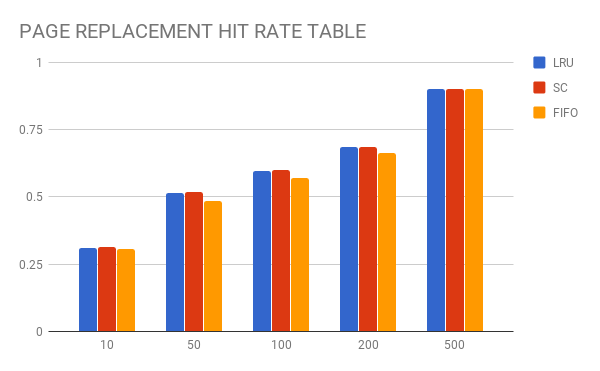
Y AXIS HIT RATE

LEGEND

BLUE is LRU

RED Is SC

ORANGE is FIFO



X axis is table/Memory Size

Y Axis is the Hit Rate

From the Graph and the Data, the hit rates are very similar to one another.

This implies that all these algorithms have similar hit rates, but they do differ in terms of efficiency and memory space.

But in terms of the hit rate, they are all similar so there is no “best” algorithm

However, with small memory sizes, such as with values less than 100, we can see that FIFO is slightly worse than the other two, but it doesn’t make that much of a difference.