Luke Manzitto

**Lab 8 Report**

FIFO Implementation:

FIFO was relatively simple to implement. I kept track of which index of the cache was next in the queue to be replaced during a page fault using an int I called “place\_in\_array”. When a page fault occurred, I would replace the index of the cache with the new page and increment place\_in\_array. This allowed me to keep track of which page in the cache would be replaced next as the next index in the cache would be the next page in the queue to be disposed of. When the place\_in\_array reached the size of the cache, I reset it back to 0, as the first item in the cache would by that point be the next page in the queue to be replaced.

LRU Implementation:

I decided to add an integer member named age within the ref\_page struct for this implementation. Doing this allowed me to keep track of the ages of each page within the cache when a page fault occurred and needed to know which index to replace. When a page was found that was already cached, I would increment the age of every other page within the cache by 1 and reset the age of the cached page back to zero. This was done to make sure the most recently cached page was not still deemed the oldest despite not needing to be replaced. If the entire cache was found not to have the new page in the cache, every element of the cache had their age incremented by 1, and the oldest element was then replaced, and their age set to 0. The oldest element would be found using an integer variable named “oldest” that would be used while iterating through the cache to find the index of the oldest page.

Second Chance:

Like the LRU I added a new member to the ref\_page struct that was a Boolean named “second\_chance”. This would allow me to mark which pages within the cache were to be given a second chance, and which were not. If the new page fetched was already in the cache, the second\_chance member would be set to true for that page within the cache, and then move on to fetching the next page to check. Should a newly fetched page not be found within the cache, the program would then iterate through the cache, if the element’s second\_chance member was true, that element would then have its second\_chance set to false but continue to stay within the cache. An integer variable named “place\_in\_array” was also used in a similar capacity to the FIFO implementation, as it would track the first index that contained an element without a second chance to then be replaced by the new page.

**Results:**

**Text

Description automatically generated**

**Hit Rate Data (%):**

Chart, bar chart

Description automatically generated

Table

Description automatically generated

**Fault Rate Data (%):**

Chart, bar chart

Description automatically generated

Table

Description automatically generated