CS 342 - Operating Systems

Project 3

Full Name: Arda İynem
ID: 22002717
Section: 1

1. Effect of Frame Count on Response Time

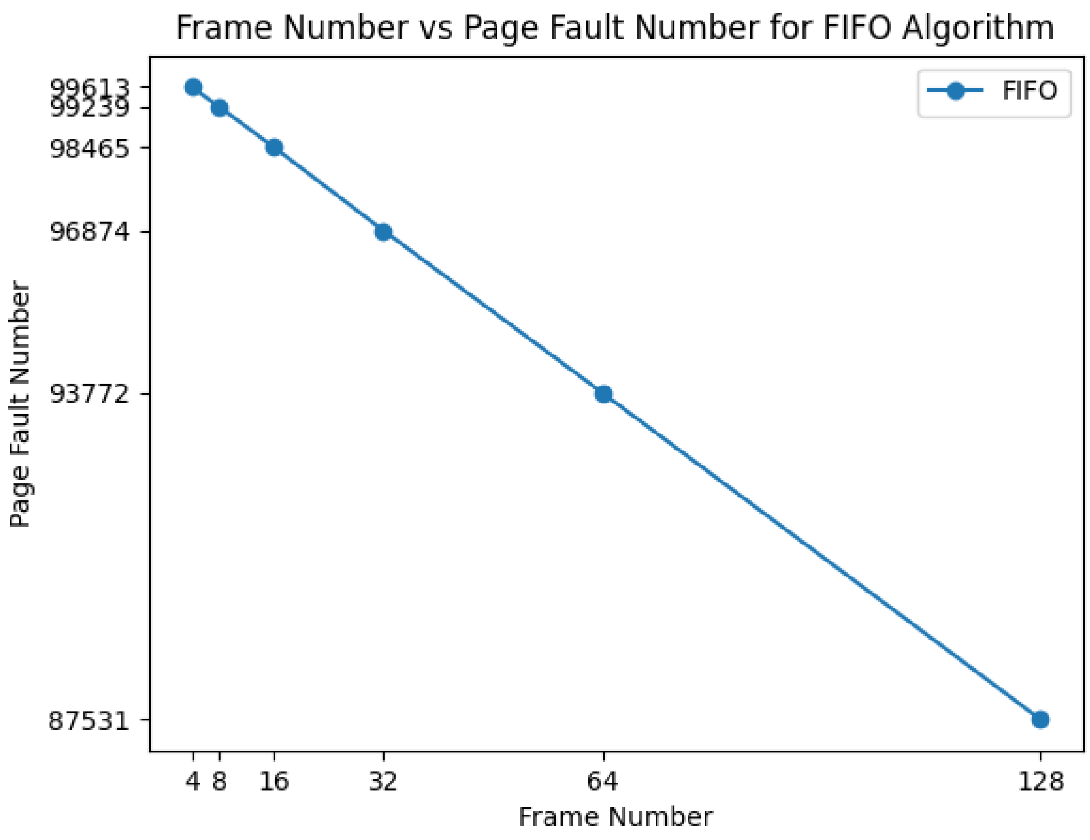
In this experiment the independent variable is frame amount (FCOUNT) which is varied to see the effect on the dependent variable page fault amount. To obtain 100.000 memory references (%80 read, %20 write operations), I created a large address file with the algorithm below. Each reference has a random VPN with boundaries [0, 1023], random offset with boundaries [0, 63] and random value with boundaries [0, 255] (write operation references only).

```
int main()
{
        FILE *refFile = fopen("ref.txt", "w+");
        char memRef[15];
        unsigned short virtualAddress;
        unsigned short value;

        for (int i = 0; i < 100000; i++)
        {
        value = rand() % 256;
        virtualAddress = rand() % 1024;
        virtualAddress = virtualAddress << 6;
        virtualAddress += rand() % 64;
        if (i % 5 == 0)
        {
        fprintf(refFile, "%c 0x%04hx 0x%hx\n", 'w', virtualAddress, value);
        }
        else
        {
        fprintf(refFile, "%c 0x%04hx\n", 'r', virtualAddress);
        }
        printf("%d\n", virtualAddress);
        }

        fclose(refFile);
}
```

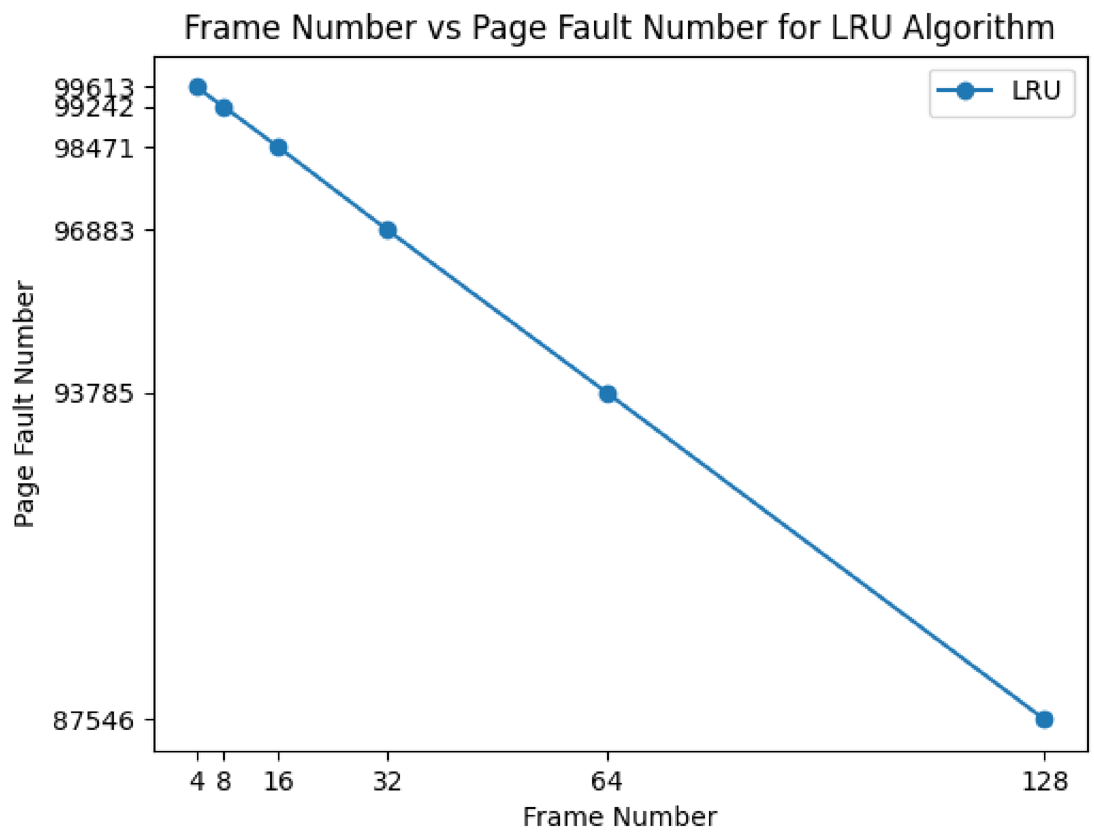
1.1 FIFO Algorithm

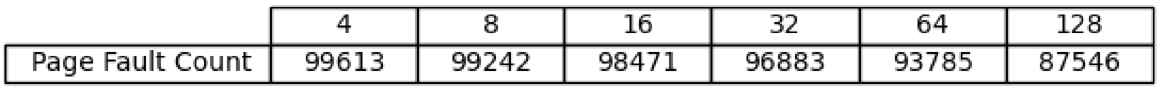


	4	8	16	32	64	128
Page Fault Count	99613	99239	98465	96874	93772	87531

The figures clearly depict that increase of frame number results in decrease of page fault number as expected. As the frame number increases, the possibility of any page to exist in the physical memory increases inevitably; This results in a decrease of page faults. FIFO algorithm simply selects the victim page to be replaced by tracking a circular queue of the pages in first in first out fashion and the selection of the victim page also has an impact on the page fault amount. However, it is less significant than that of frame count for this case as the memory references are created randomly and do not represent any real life statistics of page access information.

1.2 LRU Algorithm

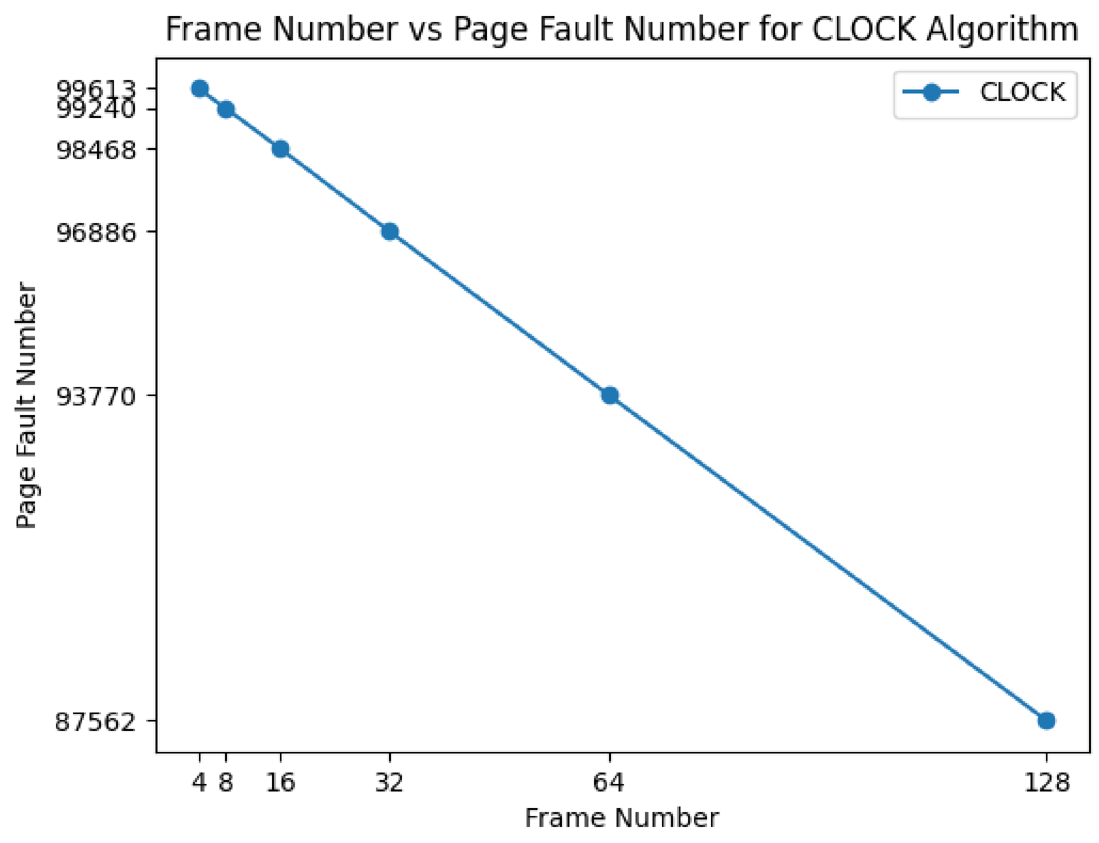




	4	8	16	32	64	128
Page Fault Count	99613	99242	98471	96883	93785	87546

The figures clearly depict that increase of frame number results in decrease of page fault number as expected. As the frame number increases, the possibility of any page to exist in the physical memory increases inevitably; This results in a decrease of page faults. LRU algorithm simply selects the victim page to be replaced by tracking a stack of the pages while maintaining the stack order according to referencing, and the selection of the victim page also has an impact on the page fault amount. However, it is less significant than that of frame count for this case as the memory references are created randomly and do not represent any real life statistics of page access information.

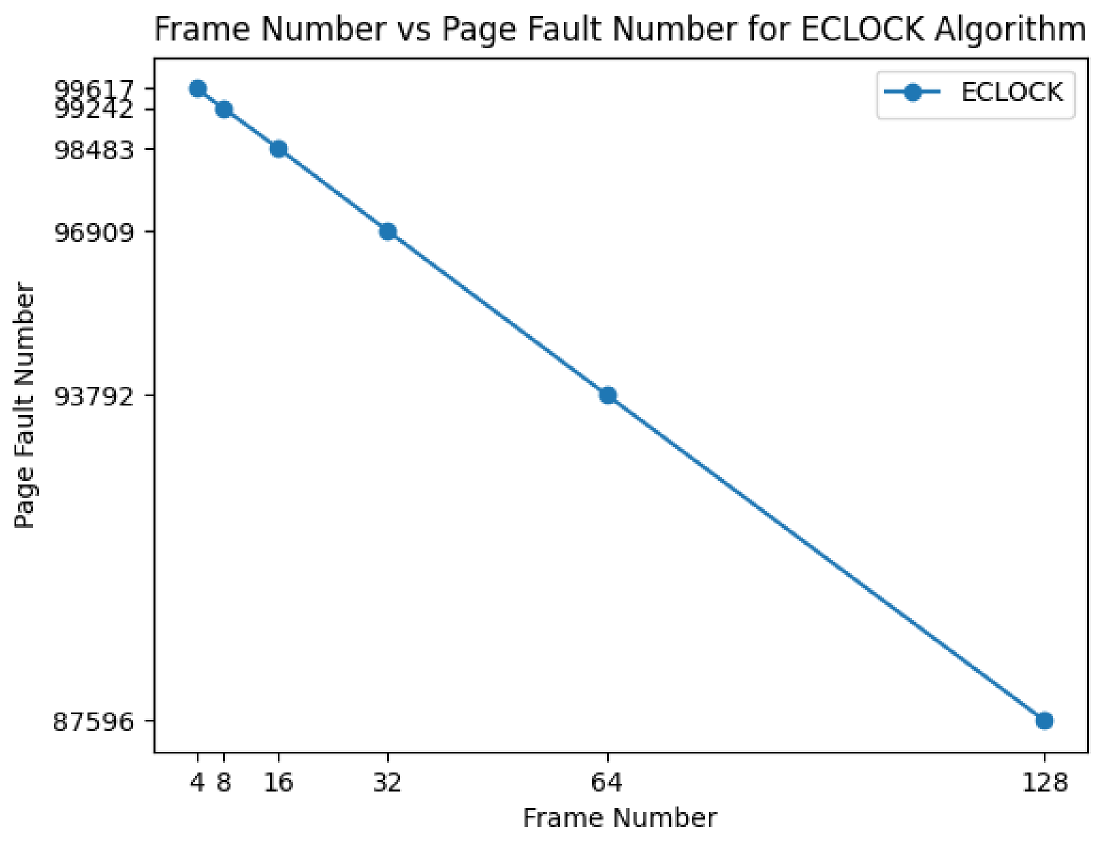
1.3 CLOCK Algorithm

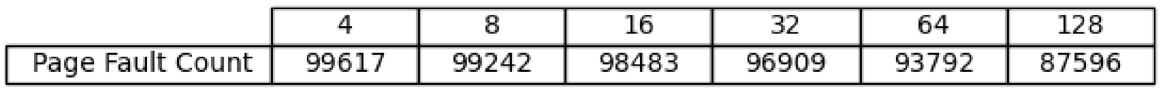


	4	8	16	32	64	128
Page Fault Count	99613	99240	98468	96886	93770	87562

The figures clearly depict that increase of frame number results in decrease of page fault number as expected. As the frame number increases, the possibility of any page to exist in the physical memory increases inevitably; This results in a decrease of page faults. CLOCK algorithm simply selects the victim page to be replaced by tracking a circular queue of the pages in first in first out fashion while considering the R bits of the corresponding page table entries too, and the selection of the victim page also has an impact on the page fault amount. However, it is less significant than that of frame count for this case as the memory references are created randomly and do not represent any real life statistics of page access information.

1.4 ECLOCK Algorithm

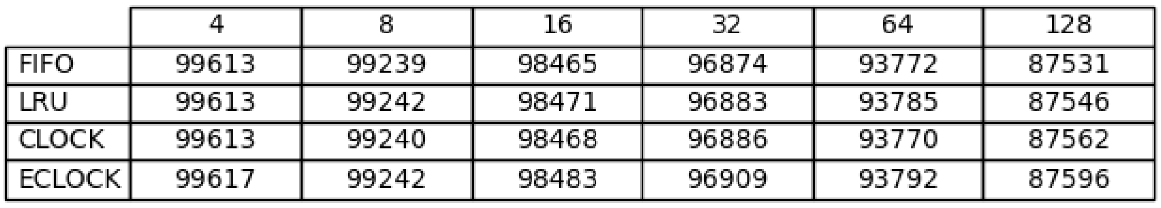




	4	8	16	32	64	128
Page Fault Count	99617	99242	98483	96909	93792	87596

The figures clearly depict that increase of frame number results in decrease of page fault number as expected. As the frame number increases, the possibility of any page to exist in the physical memory increases inevitably; This results in a decrease of page faults. ECLOCK algorithm simply selects the victim page to be replaced by tracking a circular queue of the pages in first in first out fashion while considering the R and M bits of the corresponding page table entries too, and the selection of the victim page also has an impact on the page fault amount. However, it is less significant than that of frame count for this case as the memory references are created randomly and do not represent any real life statistics of page access information.

2. Comparison of Algorithms



	4	8	16	32	64	128
FIFO	99613	99239	98465	96874	93772	87531
LRU	99613	99242	98471	96883	93785	87546
CLOCK	99613	99240	98468	96886	93770	87562
ECLOCK	99617	99242	98483	96909	93792	87596

The table depicts that increase of frame number results in decrease of page fault number for each replacement algorithm as expected. As the frame number increases, the possibility of any page to exist in the physical memory increases inevitably; This results in a decrease of page faults.

Since more sophisticated replacement algorithms intend to select a specific victim page so that the further memory references are less likely to access the swapped out page which would normally result in less page faults and a more efficient memory management process. Namely, LRU algorithm somehow predicts this ideal victim page according to past memory access information while CLOCK and ECLOCK intends to approximate LRU in a more efficient way; In contrast, FIFO is a much simpler algorithm which does not make any predictions about the victim page to decrease the page fault amount probabilistically but rather just allocates the frames in first-in first-out manner.

However, these algorithms did not produce the intended results as they normally would do in a real-life memory access scenario. This is because that memory reference file is randomly generated for the experiment, yet in the real life memory references follow certain patterns and some assumptions can be made heuristically to decrease page fault amount as normally memory references would be logically related and clustered. Yet the randomness in references caused all algorithms to produce very similar page faults and the FIFO algorithm seems to generally perform the best among all for this specific experiment due to unrealistic randomness of the adjacent memory references as explained.

In conclusion, as some of the replacement algorithms could not utilize their sophisticated prediction logic due to randomness of memory references, the only contributing factor in the page fault amount is frame number in this experiment. Therefore all algorithms performed very similarly, yet this would not be the case in a real life scenario where memory references are logically related and the replacement algorithms would matter significantly.