EX.NO: 9(a) IMPLEMENTATION OF MEMORY MANAGEMENT-PAGING

Date: 30.09.2024

AIM:

To write a C program to implement paging concept for memory management.

ALGORITHM:

```
Step 1: Start the program.
Step 2: Enter the logical memory address i.e no of pages in memory.
Step 3: Enter the page table which has offset and page frame.
Step 4: The corresponding physical address can be calculate by, PA = [
pageframe* No. of page size ] + Page offset.
Step 5: Print the physical address for the corresponding logical address. Step
6: Terminate the program.
```

PROGRAM:

```
#include <stdio.h>
#define MAX 50

int main() {
    int page[MAX], i, n, f, ps, off, pno;

    printf("Enter the number of pages in memory: ");
    scanf("%d", &n);
    printf("Enter page size: ");
    scanf("%d", &ps);
    printf("Enter number of frames: ");
    scanf("%d", &f);

    // Initialize page table
    for (i = 0; i < n; i++) {
        page[i] = -1;
    }

    printf("\nEnter the page table (Enter frame no as -1 if that page is not present in any
frame)\n\n");
    printf("Page No\tFrame No\n-------\t-------\n");
    for (i = 0; i < n; i++) {
        printf("%d\t\t", i);
        scanf("%d", &page[i]);
    }
    printf("\nEnter the logical address (i.e., page no and offset): ");
    scanf("%d%d", &pno, &off);

    // Check if the page is present
    if (pno < 0 || pno >= n || page[pno] == -1) {
        printf("\nThe required page is not available in any of the frames.\n");
    } else {
        printf("\nPhysical address (i.e., frame no and offset): %d, %d", page[pno], off);
printf("\nPhysical Address is %d\n", (page[pno] * ps) + off);
    }
    return 0;
}
```

OUTPUT:

```
Enter the number of pages in memory: 4
Enter page size: 2
Enter number of frames: 4
Enter the page table (Enter frame no as -1 if that page is not present in any frame)

Page No Frame No
------- -------
0      3
1      5
2      1
3      7

Enter the logical address (i.e., page no and offset): 2 19

Physical address (i.e., frame no and offset): 1, 19
Physical Address is 21
```

RESULT:

Thus C program for implementing paging concept for memory management has beenexecuted successfully.

EX.NO: 9 (b) PAGE REPLACEMENT ALGORITHM - (First In First Out)

AIM:

To write a C program to implement FIFO page replacement ALGORITHM.

ALGORITHM:

Step1: Read the size of the frame, no. of elements and elements one by one.
Step2: Initialize the frames with value -1.

Step3: Insert each element into frame, if it’s already not present.

Step4: If the frame is full and the new element is not already present then replace the oldest element by the new element.

Step5: Increment no. of page faults by one while inserting each element into the frames.

Step6: Display the contents of frames during processing and the total no. of page faults.

PROGRAM:

```
#include <stdio.h>
int main() {
   int reference_string[10], page_faults = 0, m, n, s, pages, frames;
   printf("Enter Total Number of Pages: ");
   scanf("%d", &pages);
   printf("Enter values of Reference String:\n");
   for (m = 0; m < pages; m++) {
      printf("Value No. [%d]: ", m + 1);
      scanf("%d", &reference_string[m]);
   }

   printf("Enter Total Number of Frames: ");
   scanf("%d", &frames);

   int temp[frames];
   for (m = 0; m < frames; m++) {
      temp[m] = -1; // Initialize frames to -1 (indicating empty)
   }
   for (m = 0; m < pages; m++) {
      s = 0; // Reset flag for page found
      for (n = 0; n < frames; n++) {
if (reference_string[m] == temp[n]) {
            s = 1; // Page hit
            break;
         }
      }
```

```
        // If page is not found, it is a page fault
        if (s == 0) {
            temp[page_faults % frames] = reference_string[m]; // Replace using
FIFO
            page_faults++;
        }

        // Display current frame state
        printf("\nCurrent Frame State: ");
        for (n = 0; n < frames; n++) {
            printf("%d\t", temp[n]);
        }
    }

    printf("\nTotal Page Faults: %d\n", page_faults);
    return 0;
}
```

OUTPUT:

```
Enter Total Number of Pages: 5
Enter values of Reference String:
Value No. [1]: 3
Value No. [2]: 5
Value No. [3]: 2
Value No. [4]: 3
Value No. [5]: 4
Enter Total Number of Frames: 3

Current Frame State: 3      -1      -1
Current Frame State: 3      5       -1
Current Frame State: 3      5       2
Current Frame State: 3      5       2
Current Frame State: 4      5       2
Total Page Faults: 4
```

RESULT:

Thus the program to implement FIFO page replacement ALGORITHM was executed successfully.

EX.NO: 9 (C) Least Recently Used (LRU)

AIM:

To write a C program to implement LRU page replacement ALGORITHM.

ALGORITHM:

Step1: Read the size of the frame, no. of elements and elements one by one.
Step2: Initialize the frames with value -1.
Step3: Insert each element into frame, if it’s already not present.
Step4: If the frame is full and new element is not already present then replace the least recently used element by the new element.
Step5: Increment no. of page faults by one while inserting each element into the frames.
Step6: Display the contents of frames during processing and the total no. of page faults.

PROGRAM:

```
#include <stdio.h>

int main() {
    int frames[10], pages[10], temp[10];
    int total_pages, total_frames;
    int page_faults = 0;

    printf("Enter Total Number of Frames: ");
    scanf("%d", &total_frames);
    for (int m = 0; m < total_frames; m++) {
        frames[m] = -1; // Initialize frames to -1 (empty)
    }
    printf("Enter Total Number of Pages: ");
    scanf("%d", &total_pages);

    printf("Enter Values for Reference String:\n");
    for (int m = 0; m < total_pages; m++) {
        printf("Value No.[%d]: ", m + 1);
        scanf("%d", &pages[m]);
    }

    for (int n = 0; n < total_pages; n++) {
        int page_found = 0; // Flag to check if page is in frames
        // Check if the page is already in one of the frames
        for (int m = 0; m < total_frames; m++) {
            if (frames[m] == pages[n]) {
                page_found = 1;
                break;
            }
        }
        // If the page is not found, we have a page fault
        if (!page_found) {
            int empty_frame = -1;
// Check for an empty frame
            for (int m = 0; m < total_frames; m++) {
                if (frames[m] == -1) {
                    empty_frame = m;
                    break;
                }
            }

            // If there is an empty frame, use it
            if (empty_frame != -1) {
                frames[empty_frame] = pages[n];
            } else {
                // If no empty frame, find the LRU page to replace
                for (int m = 0; m < total_frames; m++) {
                    temp[m] = 0; // Reset temp array
                }

                for (int k = n - 1, l = 1; l <= total_frames - 1; l++, k--) {
                    for (int m = 0; m < total_frames; m++) {
                        if (frames[m] == pages[k]) {
                            temp[m] = 1; // Mark page as used
                        }
                    }
                }

                // Find the first unused page to replace
                for (int m = 0; m < total_frames; m++) {
                    if (temp[m] == 0) {
                        frames[m] = pages[n]; // Replace the LRU page
                        break;
                    }
                }
            }
            page_faults++; // Increment page fault count
        }
        // Display current frame state
        printf("\nCurrent Frame State: ");
        for (int m = 0; m < total_frames; m++) {
            printf("%d\t", frames[m]);
        }
    }
    printf("\nTotal Number of Page Faults: %d\n", page_faults);
    return 0;
}
}
```

OUTPUT:

```
Enter Total Number of Frames: 3
Enter Total Number of Pages: 7
Enter Values for Reference String:
Value No.[1]: 5
Value No.[2]: 6
Value No.[3]: 3
Value No.[4]: 2
Value No.[5]: 5
Value No.[6]: 1
Value No.[7]: 8

Current Frame State: 5      -1      -1
Current Frame State: 5      6       -1
Current Frame State: 5      6       3
Current Frame State: 2      6       3
Current Frame State: 2      6       5
Current Frame State: 2      1       5
Current Frame State: 8      1       5
Total Number of Page Faults: 4
```

RESULT

Thus the program to implement LRU page replacement ALGORITHM was executed successfully.