

Name – Aryan Raj Roll – 211000012

Branch - CSE

Design a Virtual Memory Manager

Setup :

There are 28 entries on page table.

Page size is of 28 bytes.

There are 16 entries in TLB.

Frame size is of 28 bytes.

Number of frames is 256.

Input file :

The following program will read a file(address.txt) containing several 32-bit integer numbers that represent logical addresses. The lower 16 bits are divided into

(a) an 8-bit page number

(b) 8-bit page offset

Output file :

The output is generated in AddressOutput.txt, which consists of logical address and the converted physical address.

Platform used :

Here I have used command prompt in Windows 10 to run the C program.

To compile :

gcc Vir_Mem_Man.c

To run :

a address.txt

Code :

```
// C program to design a Virtual Memory Manager

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <unistd.h>

#define TLB_SIZE 16
#define PAGE_SIZE 256
#define FRAME_SIZE 256
#define PHYSICAL_MEMORY_SIZE PAGE_SIZE*FRAME_SIZE

int logicalAddress = 0;
int offsetNumber = 0;
int pageNumber = 0;
int physicalAddress = 0;
int Frame = 0;
int Value = 0;
int Hit = 0;
int tlbIndex = 0;
int tlbSize = 0;

unsigned pageNumberMask = 65280;
unsigned offsetMask = 255;

int tlbHitCount = 0;
float tlbHitRate = 0;
int addressCount = 0;
int pageFaultCount = 0;
float pageFaultRate = 0;
```

```

struct tlbTable {
    unsigned int pageNum;
    unsigned int frameNum;
};

int main (int argc, char *argv[])
{
    if (argc != 2)
    {
        fprintf(stderr, "Usage C_File_name <Filename.txt>
\n");
        exit(1);
    }
    //Open addresses.txt, BACKING_STORE.bin, and
    //Create AddressOutput.txt to store program results
    FILE *addresses = fopen(argv[1], "r");
    FILE *BACKINGSTORE = fopen("BACKING_STORE.bin", "rb");
    FILE *Output = fopen("AddressOutput.txt", "w");

    int physicalMemory[PHYSICAL_MEMORY_SIZE];
    char Buffer[256];
    int Index;

    int pageTable[PAGE_SIZE];
    memset(pageTable, -1, 256*sizeof(int));

    struct tlbTable tlb[TLB_SIZE];
    memset (pageTable, -1, 16*sizeof(char));

```

```

while(fscanf(addresses, "%d", &logicalAddress) == 1)
{
    addressCount++;

    //set the page number and offset for each logical
address
    pageNumber = logicalAddress & pageNumberMask;
    pageNumber = pageNumber >> 8;
    offsetNumber = logicalAddress & offsetMask;
    Hit = -1;

    //Check to see if the page number is already in
the tlb
    //If it is in tlb, then it is tlb hit
    for(Index = 0; Index < tlbSize; Index++)
    {
        if(tlb[Index].pageNum == pageNumber)
        {
            Hit = tlb[Index].frameNum;
            physicalAddress = Hit*256 + offsetNumber;
        }
    }

    if(!(Hit == -1))
    {
        tlbHitCount++;
    }

    //Gets the physical page number from page table
    else if(pageTable[pageNumber] == -1)
    {
        fseek(BACKINGSTORE, pageNumber*256, SEEK_SET);
        fread(Buffer, sizeof(char), 256,
BACKINGSTORE);
    }
}

```

```

        pageTable[pageNumber] = Frame;

        for(Index = 0; Index < 256; Index++)
        {
            physicalMemory[Frame*256 + Index] =
Buffer[Index];
        }
        pageFaultCount++;
        Frame++;

        //FIFO algorithm for the tlb
        if(tlbSize == 16)
            tlbSize--;

        for(tlbIndex = tlbSize; tlbIndex > 0;
tlbIndex--)
        {
            tlb[tlbIndex].pageNum = tlb[tlbIndex-
1].pageNum;
            tlb[tlbIndex].frameNum = tlb[tlbIndex-
1].frameNum;
        }

        if (tlbSize <= 15)
            tlbSize++;

        tlb[0].pageNum = pageNumber;
        tlb[0].frameNum = pageTable[pageNumber];
        physicalAddress = pageTable[pageNumber]*256 +
offsetNumber;
    }
    else
    {

```

```
        physicalAddress = pageTable[pageNumber]*256 +
offsetNumber;
    }
```

```
    Value = physicalMemory[physicalAddress];
```

```
    fprintf(Output, "Virtual Address: %d Physical
Address: %d Value: %d \n", logicalAddress,
physicalAddress, Value);
}
```

```
    pageFaultRate = pageFaultCount*1.0f /
addressCount;
    tlbHitRate = tlbHitCount*1.0f / addressCount;
```

```
fclose(addresses);
fclose(BACKINGSTORE);
```

```
//Print the statistics of the program to AddressOutput.txt
fprintf(Output, "Number of Addresses: %d\n",
addressCount);
fprintf(Output, "Number of Page Faults: %d\n",
pageFaultCount);
fprintf(Output, "Page Fault Rate: %f\n", pageFaultRate);
fprintf(Output, "TLB Hits: %d\n", tlbHitCount);
fprintf(Output, "TLB Hit Rate %f\n", tlbHitRate);
```

```
printf("Number of Addresses: %d\n", addressCount);
printf("Number of Page Faults: %d\n", pageFaultCount);
printf("Page Fault Rate: %f\n", pageFaultRate);
printf("Number of Page Hits: %d\n", addressCount-
pageFaultCount);
```

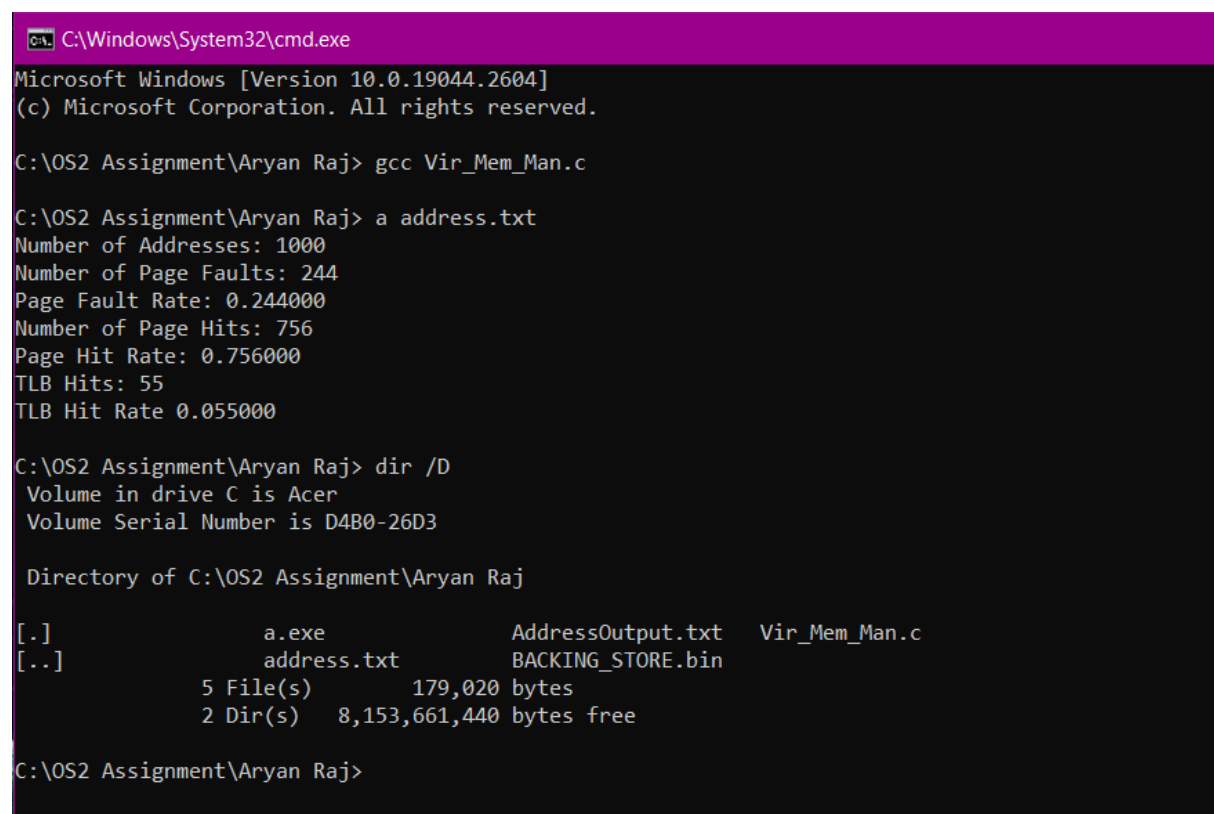
```
printf("Page Hit Rate: %f\n", 1-pageFaultRate);
printf("TLB Hits: %d\n", tlbHitCount);
printf("TLB Hit Rate %f\n", tlbHitRate);

//Close Output.txt
fclose(Output);

return 0;

}
```

Output :



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\OS2 Assignment\Aryan Raj> gcc Vir_Mem_Man.c

C:\OS2 Assignment\Aryan Raj> a address.txt
Number of Addresses: 1000
Number of Page Faults: 244
Page Fault Rate: 0.244000
Number of Page Hits: 756
Page Hit Rate: 0.756000
TLB Hits: 55
TLB Hit Rate 0.055000

C:\OS2 Assignment\Aryan Raj> dir /D
Volume in drive C is Acer
Volume Serial Number is D4B0-26D3

Directory of C:\OS2 Assignment\Aryan Raj

[.]                a.exe                AddressOutput.txt  Vir_Mem_Man.c
[..]               address.txt            BACKING_STORE.bin
                   5 File(s)            179,020 bytes
                   2 Dir(s)             8,153,661,440 bytes free

C:\OS2 Assignment\Aryan Raj>
```