DESIGNING A VIRTUAL MEMORY MANAGER

Project submitted to the
SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

Candidate Name	
M. Sai Chakradhar	**AP21110011360**
K. Rajesh	**AP21110011398**
K. Vivekananda	**AP21110011356**
J. Venkata Ramesh	**AP21110011395**

Under the
Guidance of

Dr. Tapas Kumar Mishra

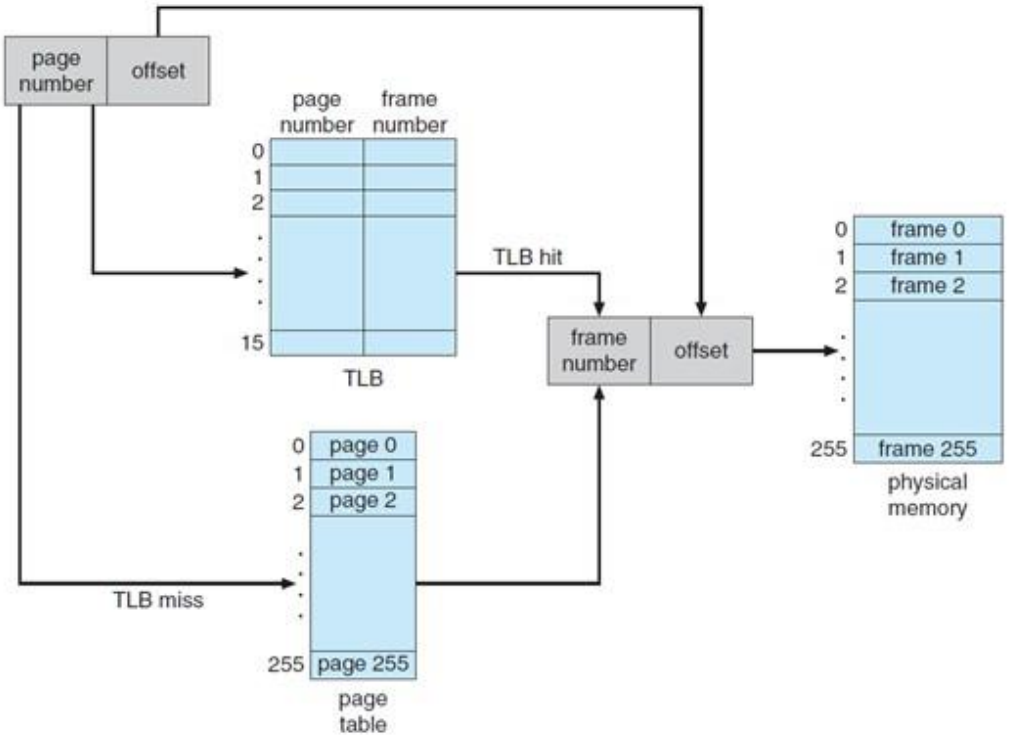
SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

OBJECTIVE:

The objective of this project is to design and implement a virtual memory manager that simulates the translation of logical addresses to physical addresses. The program will read a file containing logical addresses, use a Translation Lookaside Buffer (TLB) and a page table to perform the address translation, and output the value of the byte stored at the corresponding physical address.



OVERVIEW:

1. Initialization: The program will set up the TLB and page table to prepare for address translation. The TLB will act as a cache for recently used translations, while the page table will store the mapping between virtual and physical addresses.
2. Reading the File: The program will read a file that contains a series of logical addresses. Each logical address represents a location in the virtual address space that needs to be translated.
3. Address Translation: For each logical address read from the file, the program will first check if the corresponding physical address is present in the
   TLB. If it is, the translation is retrieved from the TLB. Otherwise, the program will consult the page table to find the mapping for the virtual address.
4. Handling Page Faults: If the translation is not found in the TLB or the page table, a page fault occurs. In this case, the program will handle the page fault by loading the required page from secondary storage (e.g., a disk) into physical memory and updating the page table and TLB accordingly.

5. Retrieving the Value: Once the physical address is determined, the program will access the value stored at that address and output it.
6. Statistics and Reporting: The program may also track and report statistics such as the number of TLB hits, TLB misses, page faults, and other relevant metrics to evaluate the performance of the virtual memory manager.

The virtual memory manager program will consist of several key components and steps:

By implementing this virtual memory manager, you will gain a better understanding of how logical addresses are translated to physical addresses using TLBs and page tables, as well as how page faults are handled in a virtual memory system.

CODE:

```
#include <bits/stdc++.h>
using namespace std;
int page_size = 128;
int page_table_size = 512; //total
int num_of_pages = page_table_size / page_size;
int actual_size = 1024; ///actual mem
int main_offset = actual_size / num_of_pages;
int *actual;
unordered_map<int,int> pagetable;
list<int> lt; // Using a list to implement the LRU queue
int f = 0;
class logical{
public:
int page;
int offset;
public:
logical(){}
logical(int page){
this->page = page;
offset = 0;
}
logical(int page,int offset){
this->page = page;
```

```
this->offset = offset;
}
};
int isfree(){
for(int i = 0; i < num_of_pages; i++)
if(pagetable[i] == -1)
return i;
return -1;
}
int find(int x){
for(int i = 0; i < num_of_pages; i++)
if(pagetable[i] == x)
return i;
return -1;
}
int physical_address(logical o){
int index = find(o.page);
if(o.offset < main_offset)
return index * main_offset + o.offset;
else
cout << "Offset out of range" << endl;
return -1;
}
void clear_mem(int t){
```

```
int start = t * main_offset;
int end = start + main_offset;
memset(&actual[start], -1, main_offset * sizeof(int));
}
int insertinTable(logical o){
int k = isfree();
if(find(o.page) != -1){
// If page is already in memory, move it to the front of the LRU queue
lt.remove(find(o.page));
lt.push_front(find(o.page));
}else if(lt.size() == num_of_pages){
// If memory is full, evict the least recently used page and load the
new page
f++;
int t = lt.back();
lt.pop_back();
clear_mem(t);
pagetable[t] = o.page;
lt.push_front(t);
} else {
// If memory is not full, load the new page
f++;
pagetable[k] = o.page;
lt.push_front(k);
```

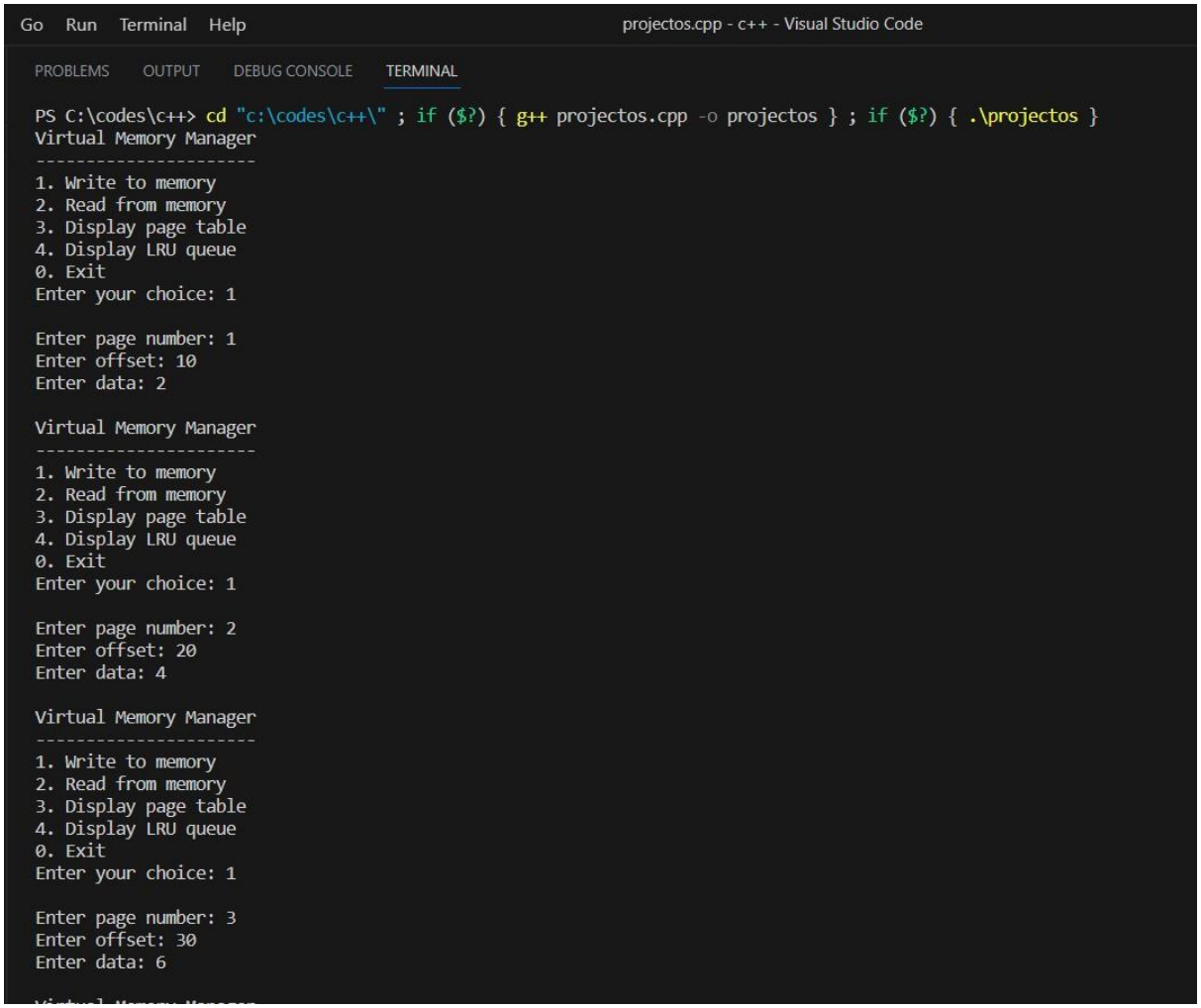
```
}
return physical_address(o);
}
void write(int address,int data){
actual[address] = data;
}
void read(int address){
cout << actual[address] << endl;
}
void page_fault(){
cout<<"Number of page faults: "<<f<<endl;
}
void display_queue() {
cout << "LRU Queue:" << endl;
for(int page_index: lt) {
cout << page_index << " ";
}
cout << endl;
}
void display_pagetable() {
cout << "Page Table:" << endl;
for(int i = 0; i < num_of_pages; i++) {
if(pagetable[i] != -1) {
cout << "Index:" << i << " Page:" << pagetable[i] << endl;
```

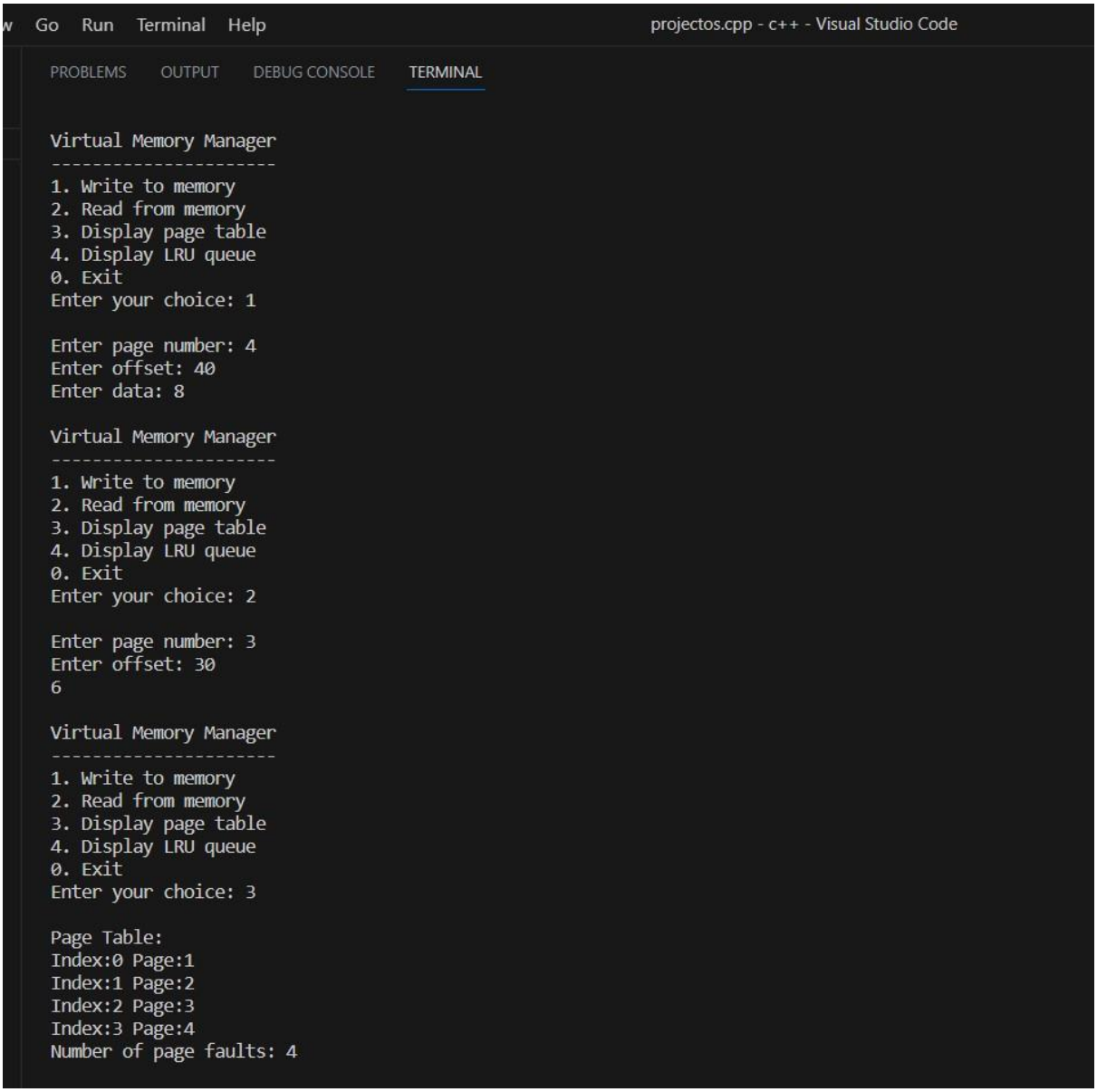
```
}
}
page_fault();
}
int main(){
actual = new int[actual_size];
memset(actual, -1, actual_size * sizeof(int));
for(int i = 0; i < num_of_pages; i++) {
pagetable[i] = -1;
}
int choice = -1;
while(choice != 0){
cout << "Virtual Memory Manager" << endl;
cout << "----------------------" << endl;
cout << "1. Write to memory" << endl;
cout << "2. Read from memory" << endl;
cout << "3. Display page table" << endl;
cout << "4. Display LRU queue" << endl;
cout << "0. Exit" << endl;
cout << "Enter your choice: ";
cin >> choice;
cout << endl;
switch(choice){
case 1: {
```

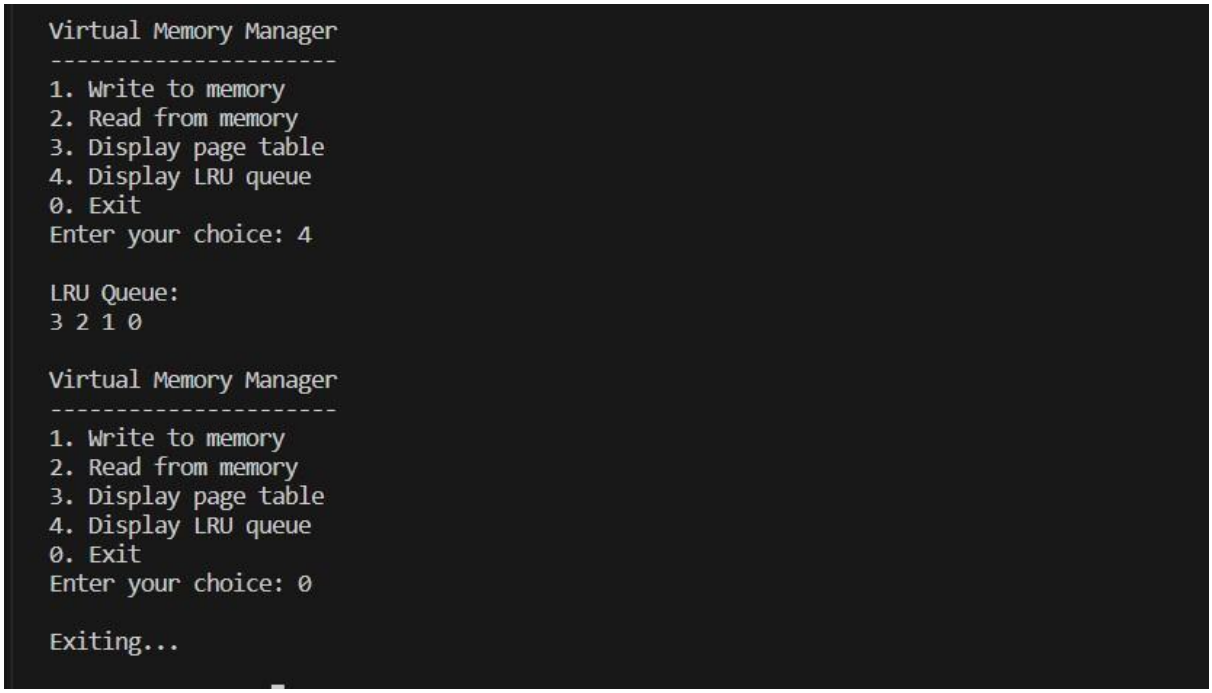
```
int page, offset, data;
cout << "Enter page number: ";
cin >> page;
cout << "Enter offset: ";
cin >> offset;
cout << "Enter data: ";
cin >> data;
int address = insertinTable(logical(page, offset));
write(address, data);
break;
}
case 2: {
int page, offset;
cout << "Enter page number: ";
cin >> page;
cout << "Enter offset: ";
cin >> offset;
int address = physical_address(logical(page, offset));
read(address);
break;
}
case 3:
display_pagetable();
break;
```

```
case 4:
display_queue();
break;
case 5:
page_fault();
case 0:
cout << "Exiting..." << endl;
break;
default:
cout << "Invalid choice" << endl;
}
cout << endl;
}
return 0;
}
```

OUTPUT:


Go Run Terminal Help
projectos.cpp - c++ - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\codes\c++> cd "c:\codes\c++\" ; if ($?) { g++ projectos.cpp -o projectos } ; if ($?) { .\projectos }
Virtual Memory Manager
1. Write to memory
2. Read from memory
3. Display page table
4. Display LRU queue
0. Exit
Enter your choice: 1
Enter page number: 1
Enter offset: 10
Enter data: 2
Virtual Memory Manager
1. Write to memory
2. Read from memory
3. Display page table
4. Display LRU queue
0. Exit
Enter your choice: 1
Enter page number: 2
Enter offset: 20
Enter data: 4
Virtual Memory Manager
1. Write to memory
2. Read from memory
3. Display page table
4. Display LRU queue
0. Exit
Enter your choice: 1
Enter page number: 3
Enter offset: 30
Enter data: 6


w Go Run Terminal Help
projectos.cpp - c++ - Visual Studio Code
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Virtual Memory Manager
1. Write to memory
2. Read from memory
3. Display page table
4. Display LRU queue
0. Exit
Enter your choice: 1
Enter page number: 4
Enter offset: 40
Enter data: 8
Virtual Memory Manager
1. Write to memory
2. Read from memory
3. Display page table
4. Display LRU queue
0. Exit
Enter your choice: 2
Enter page number: 3
Enter offset: 30
6
Virtual Memory Manager
1. Write to memory
2. Read from memory
3. Display page table
4. Display LRU queue
0. Exit
Enter your choice: 3
Page Table:
Index:0 Page:1
Index:1 Page:2
Index:2 Page:3
Index:3 Page:4
Number of page faults: 4


Virtual Memory Manager
1. Write to memory
2. Read from memory
3. Display page table
4. Display LRU queue
0. Exit
Enter your choice: 4
LRU Queue:
3 2 1 0
Virtual Memory Manager
1. Write to memory
2. Read from memory
3. Display page table
4. Display LRU queue
0. Exit
Enter your choice: 0
Exiting...

‘’’ THANK YOU’’’