**Task 1:** Implement paging and segmentation code in C.

**Code:**

#include <stdio.h>

void paging(int logical\_address, int page\_size, int page\_table\_size, int num\_pages) {

int page\_table[page\_table\_size];

for (int i = 0; i < page\_table\_size; i++) {

page\_table[i] = i; // Map logical page number to physical frame number

}

int page\_number = logical\_address / page\_size;

int offset = logical\_address % page\_size;

if (page\_number >= 0 && page\_number < num\_pages) {

int physical\_frame = page\_table[page\_number];

int physical\_address = physical\_frame \* page\_size + offset;

printf("Paging: Physical address: %d\n", physical\_address);

} else {

printf("Invalid page number\n");}}

void segmentation(int logical\_address, int segment\_size, int num\_segments) {

int segment\_table[num\_segments];

for (int i = 0; i < num\_segments; i++) {

segment\_table[i] = i \* segment\_size;

}

int segment\_number = logical\_address / segment\_size;

int offset = logical\_address % segment\_size;

if (segment\_number >= 0 && segment\_number < num\_segments) {

int base\_address = segment\_table[segment\_number];

int physical\_address = base\_address + offset;

printf("Segmentation: Physical address: %d\n", physical\_address);

} else {

printf("Invalid segment number\n");

}}

int main() {

int logical\_address;

int page\_size, page\_table\_size, num\_pages;

int segment\_size, num\_segments;

printf("Enter logical address: ");

scanf("%d", &logical\_address);

printf("Enter page size: ");

scanf("%d", &page\_size);

printf("Enter page table size: ");

scanf("%d", &page\_table\_size);

printf("Enter number of pages: ");

scanf("%d", &num\_pages);

printf("Enter segment size: ");

scanf("%d", &segment\_size);

printf("Enter number of segments: ");

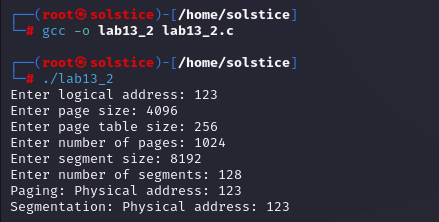
scanf("%d", &num\_segments);

paging(logical\_address, page\_size, page\_table\_size, num\_pages);

segmentation(logical\_address, segment\_size, num\_segments);

return 0;}

**Output:**



**Task2:** Implement the FIFO and LRU policies described above in C language.

**Code:**

**FIFO:**

#include <stdio.h>

#define MAX\_FRAMES 10

#define MAX\_REF\_STRING 100

void fifo(int ref\_string[], int ref\_length, int num\_frames) {

int frames[MAX\_FRAMES] = {-1}; int page\_faults = 0;

int counter = 0;

printf("FIFO Page Replacement Policy implementation\n");

for (int i = 0; i < ref\_length; ++i) {

int page\_number = ref\_string[i];

int page\_found = 0;

for (int j = 0; j < num\_frames; ++j) {

if (frames[j] == page\_number) {

page\_found = 1;

break; }}

if (!page\_found) {

if (counter == num\_frames) {

counter = 0;

}

frames[counter] = page\_number;

counter++;

page\_faults++;

}

printf("Frames: ");

for (int j = 0; j < num\_frames; ++j) {

if (frames[j] != -1)

printf("%d ", frames[j]);

else

printf("- ");

}

printf("\n");

}

printf("\nTotal Page Faults = %d\n", page\_faults);

}

int main() {

int ref\_string[MAX\_REF\_STRING];

int ref\_length, num\_frames;

printf("Enter length of reference string: ");

scanf("%d", &ref\_length);

printf("Enter reference string: ");

for (int i = 0; i < ref\_length; ++i) {

scanf("%d", &ref\_string[i]);

}

printf("Enter number of frames: ");

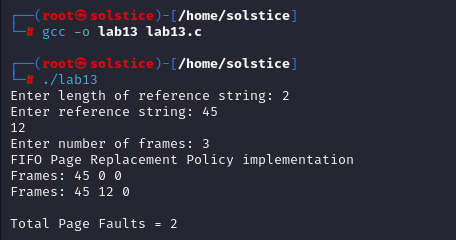
scanf("%d", &num\_frames);

fifo(ref\_string, ref\_length, num\_frames);

return 0;

}

**FIFO:**



**LRU Code:**

#include <stdio.h>

#define MAX\_FRAMES 10

#define MAX\_REF\_STRING 100

void lru(int ref\_string[], int ref\_length, int num\_frames) {

int frames[MAX\_FRAMES] = {-1}; int page\_faults = 0;

int counter = 0;

printf("LRU Page Replacement Policy implementation\n");

for (int i = 0; i < ref\_length; ++i) {

int page\_number = ref\_string[i];

int page\_found = 0;

for (int j = 0; j < num\_frames; ++j) {

if (frames[j] == page\_number) {

page\_found = 1;

break;}}

if (!page\_found) {

if (counter == num\_frames) {

counter = 0; // Reset counter if it exceeds the number of frames

}

frames[counter] = page\_number;

counter++;

page\_faults++;}

printf("Frames: ");

for (int j = 0; j < num\_frames; ++j) {

if (frames[j] != -1)

printf("%d ", frames[j]);

else

printf("- ");

}

printf("\n");

}

printf("\nTotal Page Faults = %d\n", page\_faults);

}

int main() {

int ref\_string[MAX\_REF\_STRING];

int ref\_length, num\_frames;

printf("Enter length of reference string: ");

scanf("%d", &ref\_length);

printf("Enter reference string: ");

for (int i = 0; i < ref\_length; ++i) {

scanf("%d", &ref\_string[i]);

}

printf("Enter number of frames: ");

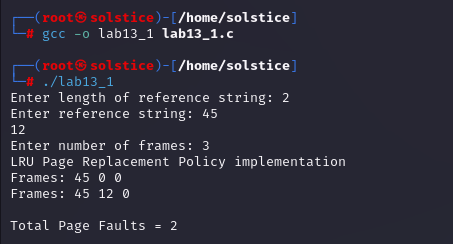
scanf("%d", &num\_frames);

lru(ref\_string, ref\_length, num\_frames);

return 0;

}

**LRU:**



**Task 3:** Execute both programs for the same set of reference strings. What difference did you observe? Comment.

**Answer:**

Executing both programs for the same set of reference strings allows us to compare the performance of different memory management schemes and page replacement policies.

By inputting the same reference strings into both the paging/segmentation code and the FIFO/LRU code, we can observe how each scheme handles page faults and memory access patterns.

**Difference:**

* Paging and segmentation focus on how memory is divided and mapped, while FIFO and LRU focus on how pages are replaced in memory.
* Paging and segmentation may result in different physical addresses for the same logical address, depending on their mapping strategies.
* FIFO and LRU may result in different numbers of page faults, depending on their replacement policies.