**Assignment 3**

#include <stdio.h>

#include <stdint.h>

#include <stdlib.h>

#define PAGE\_SIZE 4096 // 4KB = 2^12

#define PT\_ENTRIES 1024 // 2^10 entries per table

#define OFFSET\_BITS 12

#define PT\_INDEX\_BITS 10

// Simulated second-level page table (array of physical frame numbers)

uint32\_t second\_level\_page\_table[PT\_ENTRIES];

// Simulated first-level page table (each entry points to a second-level table)

uint32\_t\* first\_level\_page\_table[PT\_ENTRIES];

// Simulate initialization of page tables with dummy frame numbers

void initialize\_page\_tables() {

for (int i = 0; i < PT\_ENTRIES; i++) {

first\_level\_page\_table[i] = malloc(sizeof(uint32\_t) \* PT\_ENTRIES);

for (int j = 0; j < PT\_ENTRIES; j++) {

first\_level\_page\_table[i][j] = ((i \* PT\_ENTRIES) + j); // Just a simulated PFN

}

}

}

// Simulated virtual to physical address translation

uint32\_t translate\_virtual\_address(uint32\_t virtual\_address) {

uint32\_t offset = virtual\_address & 0xFFF; // Last 12 bits

uint32\_t pt2\_index = (virtual\_address >> OFFSET\_BITS) & 0x3FF; // Next 10 bits

uint32\_t pt1\_index = (virtual\_address >> (OFFSET\_BITS + PT\_INDEX\_BITS)) & 0x3FF; // Top 10 bits

uint32\_t frame\_number = first\_level\_page\_table[pt1\_index][pt2\_index];

uint32\_t physical\_address = (frame\_number << OFFSET\_BITS) | offset;

return physical\_address;

}

int main() {

initialize\_page\_tables();

uint32\_t virtual\_address;

printf("Enter a 32-bit virtual address in hex (e.g., 0x12345678): ");

scanf("%x", &virtual\_address);

uint32\_t physical\_address = translate\_virtual\_address(virtual\_address);

printf("Translated Physical Address: 0x%08X\n", physical\_address);

// Free allocated memory

for (int i = 0; i < PT\_ENTRIES; i++) {

free(first\_level\_page\_table[i]);

}

return 0;

}

File name = ass3

gcc ass3.c -o ass3

./ass3