

**UNIVERSITY OF GHANA**

(All Rights Reserved)

**COURSE CODE AND TITLE:** CPEN 307-Microelectronics

**COURSE INSTRUCTOR:** Mrs. Gifty Osei

**LAB INSTRUCTOR:** Desmond Xeflide

**ID**:10970285

**Name** :Kuayi Raphael

**Date:** 3rd November 2023.

**SYSTEM CALL INTERFACE**

1. **ABSTRACT**

The main ideas of the system call interface, which is a component of the operating system along with the kernel, are examined in this lab report. SystemCall (1).cpp, main (1).cpp, and systemCall (1).h are the three files that are included. Functions like acceptInput(), checkInput(), informMemory(), and repeatProgram () are included in the systemCall (1).h file.

The if-else phrases layered with else-if and else make up the main (1).cpp file. Switch break statements made it easier to transition between the various code-specified situations. This code portion also makes use of regular expressions.

# INTRODUCTION

# An Application Program Interface that offers a flexible means of communication between operating systems and user applications. A collection of functions for requesting a service from the operating system's kernel is known as a system call interface. It offers a crucial interface that connects the operating system and the and other things via a system call.

# When writing C++ code, headers and directives are often defined at the beginning of the file. Take #include <iostream> and #include "Num.h" as examples. Preprocessor directives such as #include and #define are handled by a preprocessor during the compilation process. When working with C++ source code, the preprocessor substitutes After compiling the code, #include directives with the content of the corresponding header/directive files. By doing this, most IDEs will only recompile the classes that have changed when you adjust your C++ class implementation. Since headers and directives have already been compiled and have not changed, they won't need to be recompiled.

# The fact that the header's implementation is located in a separate file makes this feasible. It is possible to create custom header files and integrate them into your C++ programs.

# CODE IMPLEMENTATION OF SYSTEM CALL INTERFACE

This generates a function to generate random numbers to populate cache1, cache2 and a RAM. The function that searches cache1, cache2 and ram for an integer. The code must have a function that replaces the first element of cache1 with the integer searched for if the integer could not be found in cache1.The code consists of a

header file named “systemCall.h”. This is the header file that implements the system call interface.

In the “systemCall.h” that create a function to accept input from the user. Next, write a function definition to check the validity of the input thus check if the input is a string or special character such as a float. Make use of regular expressions.

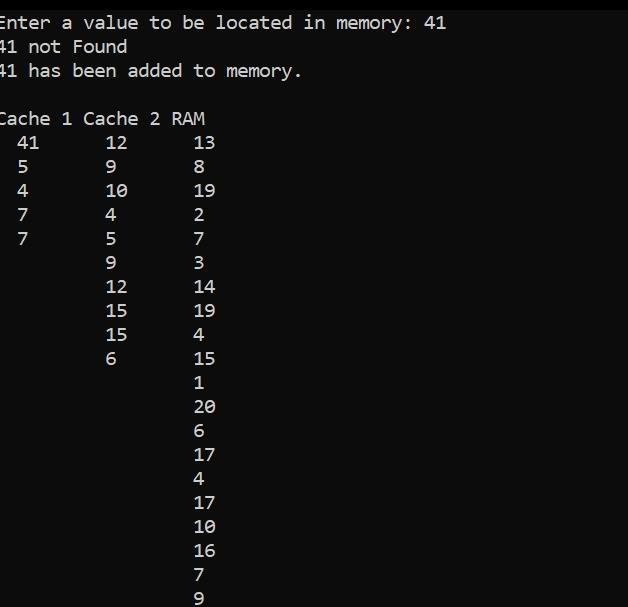
Create a function that informs the user about the memory in which the number was found and displays the contents of cache1, cache2 and RAM. Create a C++ file

called “systemCall.cpp” that implements a function definition in “systemCall.h”. The “systemCall.h” that is included in your kernel space program. All requests to the user for the input and are all displayed to the user to be handled by the “systemCall” program and processes it and sends the output to the “systemCall” program to display it.

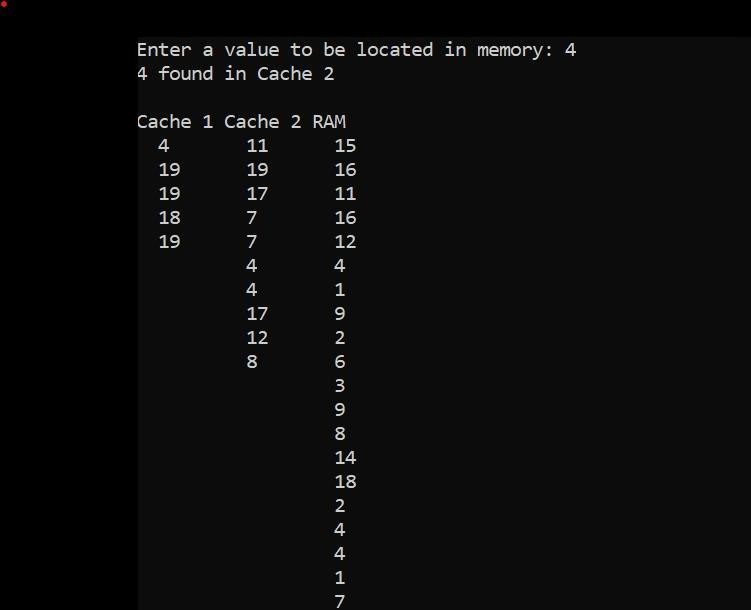
# OUTPUT OF CODE IMPLEMENTATION

In the terminal, the program asks users to enter a value to be located in memory .If that number is not found, the value is added to cache1.The cache1 has a fixed size of 5,cache2 holds a static size of 10 and the RAM has a size of 20.Below is an image of the code output.

* 1. **IMAGE SHOWING VALUE NOT FOUND IN MEMORY**



* 1. **IMAGE SHOWING VALUE FOUND IN MEMORY**



**ress 1 to Re-run or any other number to exit: 1 nter a value to be located in memory: 12**

**2 found in RAM**

**ache** 1 **Cache** *2* RAM

12 14 12

1 5 13

18 7 1

8 6 5

19 1 18

14 9

4 15

1 3

15 6

2 4

10

11

3

8

*2*

19

14

15

15

13

# CONCLUSION

# This lab is crucial because it closes the knowledge gap between the theoretical concepts covered in class and the real system call interface. A glimpse of writing code to implement the system call interface was given to the students. Students noticed that the kernel and the System Call Interface made up the operating system. Process manager, memory manager, network manager, file manager, and device manager are components of the kernel.