UNIVERSITY OF GHANA
SCHOOL OF ENGINEERING SCIENCES
COLLEGE OF BASIC AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING
FIRST SEMESTER 2023/2024 ACADEMIC YEAR

COURSE CODE: CPEN 307 - OPERATING SYSTEMS
COURSE INSTRUCTOR: DR.GIFTY OSEI

LAB: 1

NAME:
DOE AGUDEY DANIEL
ID:
10956661

DATE: 28/10/2023

ANALYZING MEMORY HIERARCHY IN COMPUTING SYSTEMS

ABSTRACT

This laboratory exercise delves into the realm of memory hierarchy and its impact on system performance. Three distinct exercises were undertaken, each independently conducted. The first and second involved the calculation of average memory access times for varying computer configurations, and the third simulated data retrieval within the memory hierarchy. By examining cache structures, RAM, and the significance of memory access, valuable insights into the crucial role that hardware and software interplay plays in memory performance were gained. The results highlight the significance of cache size, RAM capacity, storage type, and CPU speed in determining access times, offering practical lessons for optimizing memory utilization in computing systems.

INTRODUCTION

The effective management of memory is a cornerstone of modern computer systems. Memory hierarchies, comprising various levels of storage from cache to RAM and secondary storage, play a pivotal role in determining how quickly data can be accessed. In this laboratory exercise, we independently explored the intricacies of memory hierarchy and the influence of hardware specifications on memory access times.

The first and second exercise involved the development of a program to compute the average memory access time for diverse computer configurations. By evaluating the impact of memory levels, cache, and access speeds, valuable insights were gained into how these factors affect overall system performance.

The third exercise focused on simulating the data retrieval process within a computer's memory hierarchy. By searching for specific data in a designated sequence, the real-world implications of cache levels, RAM, and the order of data retrieval were explored.

RESULTS AND DISCUSSIONS

Exercise 1

1. Integer array size 100,000,000: 0.32 seconds
2. Integer array size 50,000,000: 0.18 seconds
3. Double data type
   - (i) array size 100,000,000: 0.827 seconds
   - (ii) array size 50,000,000: 0.412 seconds

   Float data type
   - (i) array size 100,000,000: 0.32 seconds
   - (ii) array size 50,000,000: 0.18 seconds

Yes, it changes.
REASON
Altering the data type of the array to double and float had considerable effects on access times. The latency associated with accessing and storing data in memory is intrinsically linked to both the size of the data type and the underlying hardware attributes. Specifically, larger data types like double(8 bytes) necessitate a more extensive memory allocation, leading to comparatively delayed access times in contrast to more compact data types, such as float and int which mostly have a size of 4 bytes.

4.

	Cache Size			**RAM**	**Secondary Stage**	**CPU Speed**
	L1	L2	L3			
Colleague 1	256kb	1mb	6mb	8GB	1TB HDD	1.6GHz
Colleague 2	320kb	5mb	8mb	8GB	236TB SSD	2.5GHz
Colleague 3	128kb	512kb	3mb	16GB	256 GB SSD	2.7GHz

For Colleague 1:
Access Times:
Integer array size 100,000,000: 0.3 seconds
Integer array size 50,000,000: 0.24 seconds

Double data type array
(i) array size 100,000,000: 0.7 seconds
(ii) array size 50,000,000: 0.48 seconds
Float data type array
(i) array size 100,000,000: 0.3 seconds
(ii) array size 50,000,000: 0.24 seconds

For Colleague 2:
Access Times:
Integer array size 100,000,000: 0.29 seconds
Integer array size 50,000,000: 0.162 seconds

Double data type array
(i) array size 100,000,000: 0.6 seconds
(ii) array size 50,000,000: 0.34 seconds
Float data type array
(i) array size 100,000,000: 0.29 seconds
(ii) array size 50,000,000: 0.162 seconds

For Colleague 3:
Access Times:
Integer array size 100,000,000: 0.27 seconds
Integer array size 50,000,000: 0.14 seconds
Double data type array
(i) array size 100,000,000: 0.46 seconds
(ii) array size 50,000,000: 0.24 seconds
Float data type array

(i) array size 100,000,000: 0.27 seconds
(ii) array size 50,000,000: 0.14 seconds

The differences and similarities in access times between my results and those of Colleague 1, Colleague 2, and Colleague 3 can be attributed to various factors related to their computer specifications and the principles of operating systems:

Cache Size and Organization:

- My colleagues have varying cache sizes and organizations. Colleague 1 has a moderately sized cache configuration. Colleague 2 has a slightly larger cache, and Colleague 3, while having smaller L1 and L2 caches, benefits from a more substantial L3 cache. These cache differences can affect access times, as a larger and more organized cache structure can lead to improved cache hit rates and, consequently, faster memory access.

RAM Size:

- Colleague 3 stands out with 16GB of RAM, while I have 4GB. Having more RAM allows for a greater amount of data to be stored in memory, reducing the need for frequent data transfers between RAM and secondary storage, which can lead to quicker access times, especially when working with large arrays.

Storage Type:

- Colleague 2 has a notable advantage with an SSD compared to my HDD. SSDs offer significantly faster data retrieval, particularly when loading data from secondary storage into RAM. This can substantially improve access times. In contrast, my HDD storage may result in slower data retrieval.

CPU Speed:

- The clock speed of the CPU plays a crucial role in data processing. Colleague 3's CPU is notably faster at 2.7GHz, while I have a 1.6GHz CPU. Colleague 2 has a 2.5GHz CPU. A faster CPU can process data more quickly, which can translate to quicker access times, contributing to the disparities observed in memory access times.

In summary, the differences in memory access times are primarily due to variations in cache sizes, RAM capacity, storage types, and CPU speeds among me and my colleagues. These hardware specifications significantly impact the efficiency of memory access. Moreover, the presence of SSDs, which provide faster data retrieval, can result in shorter access times compared to traditional HDD storage.

Exercise 2: Average Memory Access Time

In Exercise 2, I developed a software tool to compute the average memory access time for a range of computer setups. These configurations encompassed different memory hierarchies, including primary memory, single-level cache, and two-level cache systems. The application guided me through the process, allowing me to specify the number of memory levels, input access times, and speed ratios for each memory component. Subsequently, the program carried out the necessary computations and presented the average access time.

SCREENSHOTS

```
"C:\Users\Daniel\Desktop\L300 first Semester\Operating Systems\DOE_10956661_LAB1\exercise...
Choose a memory configuration (a, b, or c):
a) Computer with only main memory
b) Computer with a one-level cache and main memory
c) Computer with a two-level cache and main memory
b
Enter access time for memory level 1 (T1): 0.6
Enter hit ratio for memory level 1 (H1): 0.98
Enter access time for memory level 2 (T2): 3.0
Enter hit ratio for memory level 2 (H2): 0.2
Average Memory Access Time: 0.66 microseconds

Process returned 0 (0x0)   execution time : 64.592 s
Press any key to continue.
```

Screenshot 1: I entered access times and speed ratios for each option b, one level cache and main memory, and the average access time was calculated to be 0.66 microseconds.

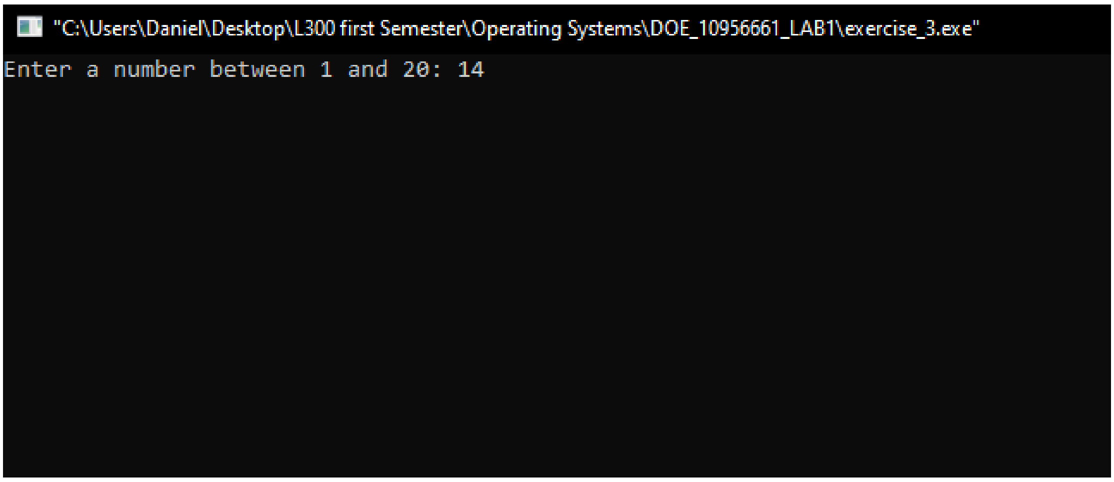
```
"C:\Users\Daniel\Desktop\L300 first Semester\Operating Systems\DOE_10956661_LAB1\exercise...
Choose a memory configuration (a, b, or c):
a) Computer with only main memory
b) Computer with a one-level cache and main memory
c) Computer with a two-level cache and main memory
c
Enter access time for memory level 1 (T1): 10
Enter hit ratio for memory level 1 (H1): 0.9
Enter access time for memory level 2 (T2): 100
Enter hit ratio for memory level 2 (H2): 0.8
Enter access time for memory level 3 (T3): 1000
Average Memory Access Time: 41 microseconds

Process returned 0 (0x0)   execution time : 48.926 s
Press any key to continue.
```

Screenshot 2: I entered access times and speed ratios for each option c, two level cache and main memory, and the average access time was calculated to be 41 microseconds.

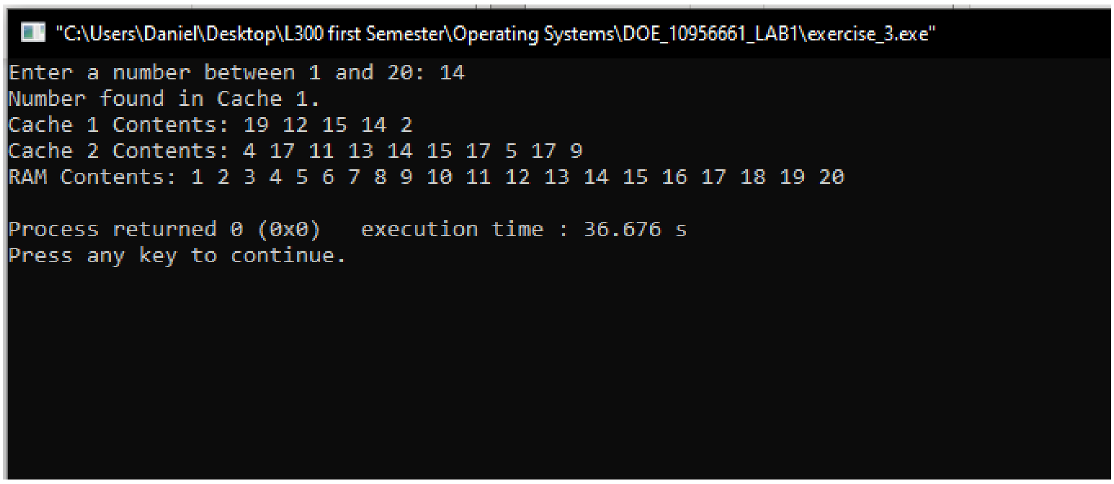
Exercise 3: Programming the Memory Hierarchy

For Exercise 3, I undertook a programming task centered around simulating the data retrieval mechanism within a computer's memory hierarchy, considering various cache levels and the main system memory (RAM). The program created and populated three distinct arrays, namely cache1, cache2, and RAM, with random data. It then prompted me to input a numerical value ranging from 1 to 20, initiating a search process that followed the designated order of exploration, beginning with cache1, moving to cache2, and culminating in RAM. The program's output provided information about the search results and indicated the specific memory location where the sought-after number was located.

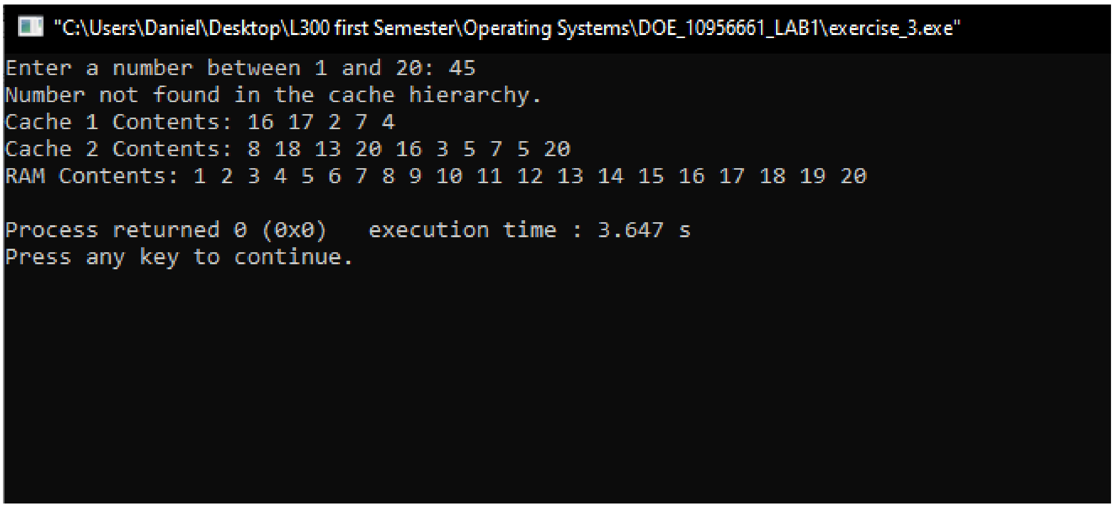
SCREENSHOTS OF STEPS



Screenshot for step 3: I provided input for a number to search for.



Screenshot for step 4: The program displayed search results and the memory location where the number was found.



Screenshot for error testing: if the user enters a number out of the range

CONCLUSION

In conclusion, these laboratory exercises have provided a comprehensive understanding of memory hierarchy and its profound impact on computer system performance. Independently conducting the exercises allowed for hands-on experience and in-depth learning.

It was observed that the size and organization of CPU caches, the amount of RAM available, the type of secondary storage, and CPU clock speed significantly affect memory access times. The practical knowledge gained from these experiments equips us with the tools to optimize memory utilization and enhance the performance of computing systems. As technology continues to advance, understanding memory hierarchy remains a fundamental aspect of designing and using computer systems effectively.