# COMP1562 Operating Systems

LOGBOOK

|  |  |  |
|---|---|---|
| NAME: | *CHISEL* | |
| SURNAME: | *CHAVUSH* | |
| STUDENT ID: | *001131628* | |
| COURSE: | *BEng H SOFTWARE ENGINEERING* | |
| DATE: | *15/03/2023* | |
| COURSE: | *COMP1562 Operating Systems* | |
| COURSEWORK ID*:* | *244742* | |
| COORDINATOR: | *Najma Taimoor* | |
| GROUP NUMBER: | *12* | |
| GROUP MEMBERS: | -CHISEL CHAVUSH | (001131628) |
| | -EKREM SAID IZ | (001141646) |
| | -EMIR MENEKSHELI | (001084980) |
| | -BARIS CELIKTUTAN | (001120868) |
| | -JORDAN LANKFORD | (001151917) |

## CONTENTS

# WEEK 1

## TASK 1.1

### Introduction/Explanation:

Memory locations of various sorts are critical when it comes to storing and processing data in a computer system. These three areas are data registration, buffers, and CPU registers, all of which are required for data handling within a computer. Data registration refers to the process of inputting information into a computer system. This data might come from a variety of sources, including human input, sensors, or other technology. Once this information has been recorded, the system can use it for processing, analysis, or other actions. Contrarily, buffers are short-term storage areas that hold data while it is being moved between various components of a computer system. Typically, buffers are employed to even out any variations in data transfer rates, ensuring that the system runs smoothly. (P. Heuring & F. Jordan, 1997).
Finally, CPU registers are tiny, high-speed storage areas that are integrated into a computer system's central processor unit (CPU). These registers are employed to store data that the CPU is actively processing, enabling quicker access and processing times. (Hyuk Lee et al., 2012)

Overall, data registration, buffers, and CPU registers are essential parts of contemporary computer systems that operate together to guarantee correct and timely data processing. (P. Heuring & F. Jordan, 1997)

For this Task first let's have look at the few terms that we are going to use;

In the given task, the system contains two I/O device buffers at address 005h and 006h. ***Buffer*** is a region of memory that temporarily hold data for a while it is being moved from one place to another. The initial values for I/O buffers is 0001h and 0001h. In the 1st step I have assigned the initial buffer values to at addresses 005h and 006h. As its requested in Task 1.1 **Instruction** has been placed in the memory location 300h, 301h, 302h. Memory locations to store and get data are 900h, 901h, 902h. So initial values in the memory  locations is respectively, 0100h, 0010h, and 0001h.

So this 16 bit hypothetical processor has three operations to do. Step 1 / 2 is first instruction.

Step 3 / 4 is second instruction and step 5 / 6 is the third instruction. Instruction in the one normal cycle comprised of fetch stage and execute stage.

If we have 16 bit processor, than the size of the accumulator instruction register also has to be 16 bit. Our Program Counter (PC) is 12 bit, its because also the addresses like 300,301,302 are 12 bit.

The instructions has 2 part, the first part is 4 bit opcode and the second part is 12 bit address which is also remainder. It will tell the process or the potentially what the address for the operation is.

The opcode (operation code) is the instruction that is executed by the CPU and the operand is the data or memory location used to execute that instruction.

## STEP 1: (FETCH STAGE)

1) For our first instruction, we are going to load the accumulator from device to buffer at address 005h. It is 12 bit in order to complete this to 16 bit, we have to load AC from I/O which is 0011. We have to convert this opcode to hexadecimal. Which is going to be 3. So our first address is going to be 3005.

2) Now we will add contents of memory location 901. This is 12 bit, as we require to complete this to 16 bit we have to pass to AC which 0101. When we convert this binary number to hexadecimal it is going to be 5. So our second address registration is going to be 5901.

3) In the last Instruction memory location, we are going to store AC to device buffer at address 006. In order to complete this to 16 bit, we will Store AC to I/O which is 0111. When we convert this binary number to hexadecimal we will get 7. So our third memory address register will be 7006.

Instruction Register (IR) in Step 1 is going to be same as first memory address register which is 3005. In the 1st step our Accumulator (AC) does not have any value, we will assign that in the 2nd step.

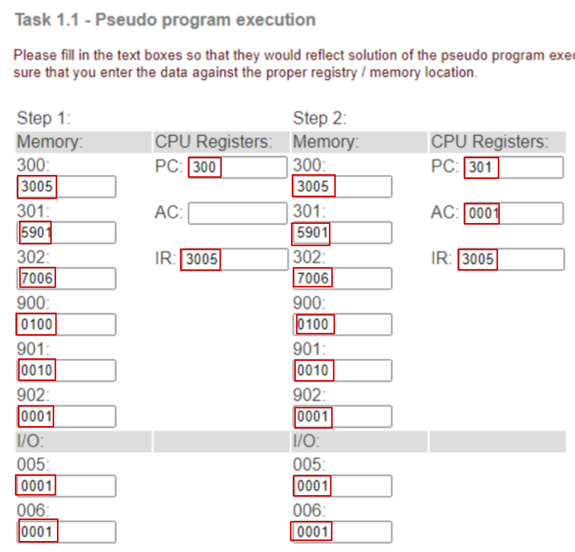
## STEP 2: (EXECUTE STAGE)

1) In step 2 our memory address location stays same, for 300:3005(16bit) / 301:5901(16bit) / 302:7006(16bit). Also memory locations to store / get data and initial values stays same. For 900:0100 / 901:0010 / 902:0001.

2) In second step our program counter (PC) is increased by 1. Memory location was 300 , in order to pass to second memory location which is 301 we increased PC by 1.

3) Than we load the AC from I/O. Initial values for I/O was 0001. That's why our AC is going to load with 0001.

4) Instruction register stays same in this stage because program is executing the values from step.



Task 1.1 - Pseudo program execution

Please fill in the text boxes so that they would reflect solution of the pseudo program execution task. Make sure that you enter the data against the proper registry / memory location.

| Step 1: | | Step 2: | |
|---|---|---|---|
| Memory: | CPU Registers: | Memory: | CPU Registers: |
| 300: 3005 | PC: 300 | 300: 3005 | PC: 301 |
| 301: 5901 | AC: | 301: 5901 | AC: 0001 |
| 302: 7006 | IR: 3005 | 302: 7006 | IR: 3005 |
| 900: 0100 | | 900: 0100 | |
| 901: 0010 | | 901: 0010 | |
| 902: 0001 | | 902: 0001 | |
| I/O: | | I/O: | |
| 005: 0001 | | 005: 0001 | |
| 006: 0001 | | 006: 0001 | |

Figure 1

## *STEP 3: (FETCH STAGE)*

1) Memory address location stays same, for 300:3005(16bit) / 301:5901(16bit) / 302:7006(16bit). Also memory locations to store / get data and initial values stays same. For 900:0100 / 901:0010 / 902:0001.

2) In Step 2, we were passed the second memory location which was 301. Now in Step 3,from CPU registers only Instruction Register (IS) is going to change. Our Instruction Register (IR) is going to be 5901, because processor is fetch second memory location(301). The value in memory location 301 is 5901.

3) In CPU Registers, Program Counter (PC) and Accumulator (AC) stays same.

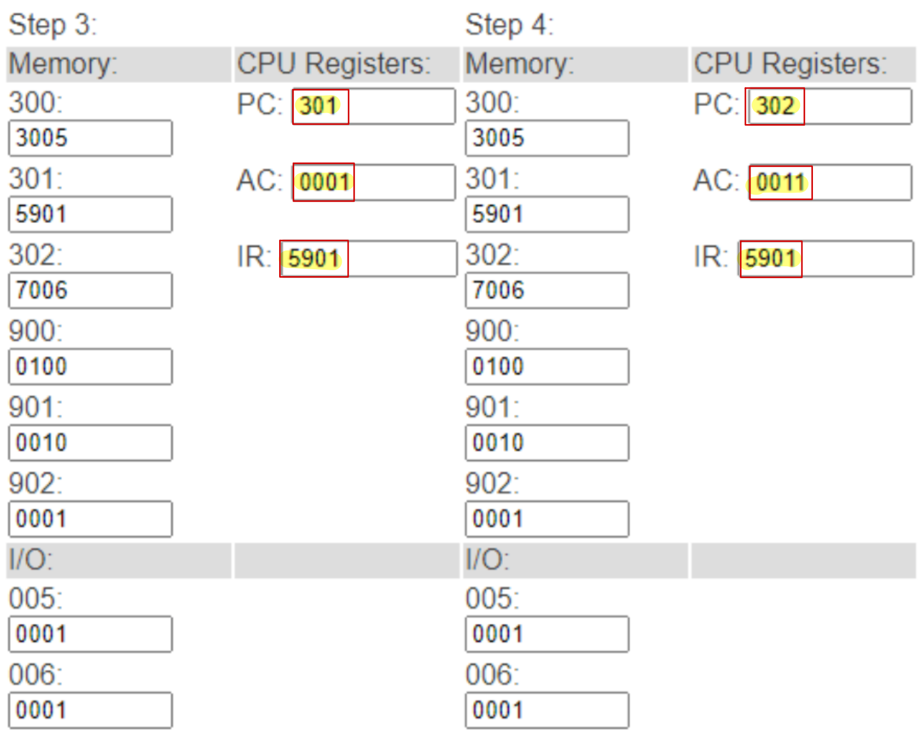
4) I/O values was already assigned, so nothing changes.

## *STEP 4: (EXECUTE STAGE)*

1) In step 4, we increase the Program Counter (PC) by one again in order to pass the next memory location which is 302.

2) In CPU Register, Accumulator (AC) is going to be 0011. Because in the instruction we were requested to Add contents of memory location 901h. The value in 901 is 0010. If we add 0001 (First AC) with 0010, it will give us new AC value which is 0011. We store that value in AC (step 4).

3) I/O values was already assigned, so nothing changes.



| Step 3: | | Step 4: | |
|---|---|---|---|
| Memory: | CPU Registers: | Memory: | CPU Registers: |
| 300: 3005 | PC: 301 | 300: 3005 | PC: 302 |
| 301: 5901 | AC: 0001 | 301: 5901 | AC: 0011 |
| 302: 7006 | IR: 5901 | 302: 7006 | IR: 5901 |
| 900: 0100 | | 900: 0100 | |
| 901: 0010 | | 901: 0010 | |
| 902: 0001 | | 902: 0001 | |
| I/O: | | I/O: | |
| 005: 0001 | | 005: 0001 | |
| 006: 0001 | | 006: 0001 | |

*Figure 2*

## *STEP 5 (FETCH STAGE)*

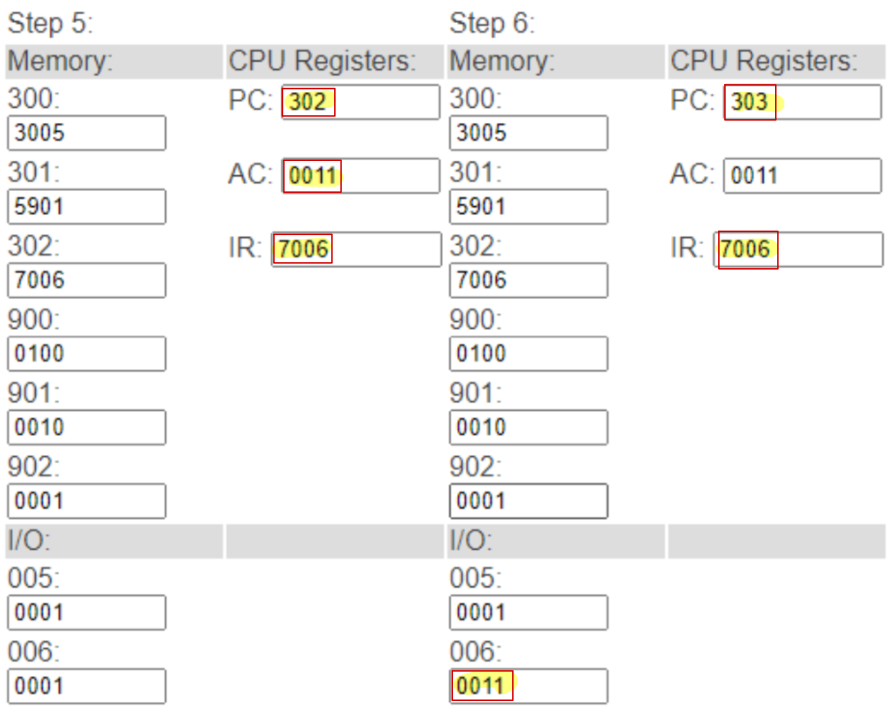
1) In Step 4, we were passed the second memory location which was 302. Now in Step 5,from CPU registers only Instruction Register (IS) is going to change. Our Instruction Register (IR) is going to be 7006, because processor is fetch second memory location(302). In this memory location the value is 7006 that's why in Instruction Register we store the value of 7006.

2) In CPU Registers, Program Counter (PC) and Accumulator (AC) stays same.

## *STEP 6 (EXECUTE STAGE)*

1) In step 6, we increase the Program Counter (PC) by one again in order to pass the next memory location which is 303.

2) In step 6, I/O buffer for memory location 006h is going to change, as we were requested to store AC to device buffer at address 006h. Our AC value was becoming 0011. So new value in I/O buffer for memory location 006 is going to be 0011.



Step 5:

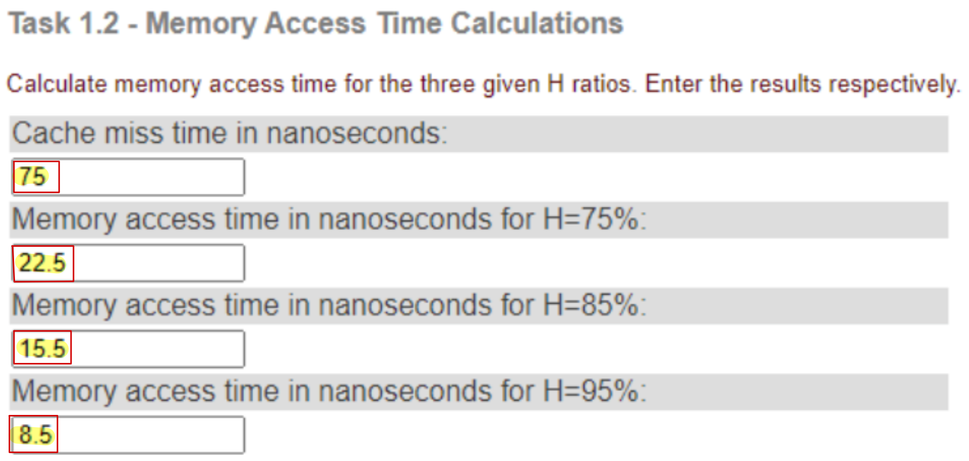
| Memory: | CPU Registers: |
|---|---|
| 300: 3005 | PC: 302 |
| 301: 5901 | AC: 0011 |
| 302: 7006 | IR: 7006 |
| 900: 0100 | |
| 901: 0010 | |
| 902: 0001 | |
| I/O: | |
| 005: 0001 | |
| 006: 0001 | |

Step 6:

| Memory: | CPU Registers: |
|---|---|
| 300: 3005 | PC: 303 |
| 301: 5901 | AC: 0011 |
| 302: 7006 | IR: 7006 |
| 900: 0100 | |
| 901: 0010 | |
| 902: 0001 | |
| I/O: | |
| 005: 0001 | |
| 006: 0011 | |

Figure 3

## *TASK 1.2*


Task 1.2 - Memory Access Time Calculations

Calculate memory access time for the three given H ratios. Enter the results respectively.

Cache miss time in nanoseconds:

75

Memory access time in nanoseconds for H=75%:

22.5

Memory access time in nanoseconds for H=85%:

15.5

Memory access time in nanoseconds for H=95%:

8.5

*Figure 4*

In this task we have been given some values.

**Consider a machine with:**

1. 1k words cache, *access time 5ns.*
2. 1M words memory, *access time 70ns.*
3. If the data is not in cache then the data is copied from memory.

Let's have a look at the term of Catching. It is a faster memory system can be used as cache memory for slower memory system. Just in case if there is any information needed, first cache memory is being searched; depending on search results the information is copied to cache.

Also we are going to use Term "Hit ratio": Hit ratio H, Fraction of all memory access that are found in cache

1) Cache miss time in nanoseconds:
   *If data is not in cache, then the data is copied from memory
   1M words memory, access time 70ns + 1k words cache, access time 5ns.

So in total -> 70ns + 5ns = 75ns

2) Memory access time in nanoseconds for H = 75%:

H = 75% -> 0.75 x 5 + 0.25 (70 + 5) = 22.5

3) Memory access time in nanoseconds for H = 85%:

H= 85% -> 0.85 x 5 + 0.15 (70 + 5) = 15.5

4) Memory access time in nanoseconds for H = 95%:

H = 95% ->0.95 x 5 + 0.05 (70 + 5) = 8.5

## ***Conclusion:***

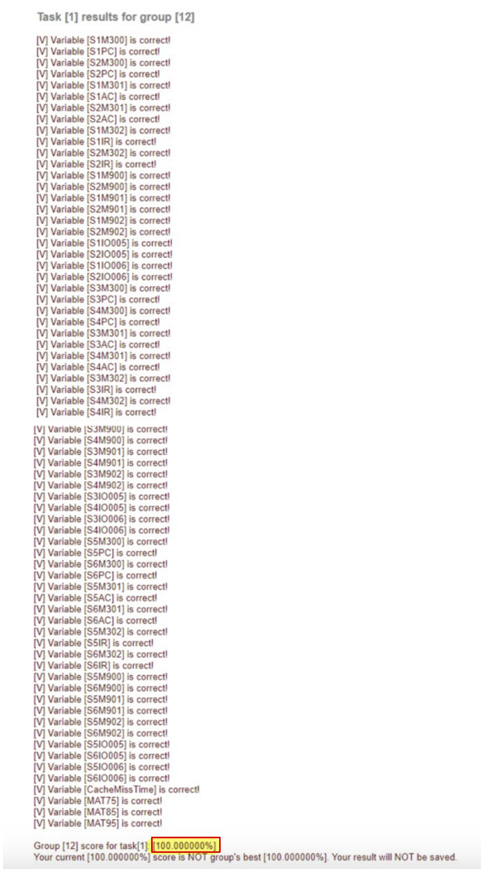
In conclusion, data registration, buffers, and CPU registers are all essential elements of computer systems that assist manage and process data effectively. Data storage and retrieval are made possible by data registration, data movement between components of a system is facilitated by buffers, and CPU registers, which offer quick access to frequently used data, speed up data processing. A computer system can handle more complicated tasks and operate more effectively when these components are used in combination to boost its overall performance and speed. These components will probably keep evolving and improving for better computing capabilities as technology develops.(Patterson, 2013)

## ***What I learned:***

In week 1 , I have been learned about Data Registrations in the memory, also buffers and CPU Registers. It was little bit challenging at the beginning because of some unknown terms, but now I am feeling quite comfortable with the topic. Doing this weekly tasks helped me about to understand the topic in practise as well. When I compare myself before this weekly tasks and after, I can see that with these practise I clearly understand the topic of this subject and learn how to implement them in real world examples. I clearly understand the process of the Data Registration with Task 1.1

While I was doing the Task 1.2, I was aware of the terms but, at the beginning I was little bit confused about how to calculate Memory Access. By doing this task I fairly understand the calculation process of Memory Access.

RESULT : 100%



```
Task [1] results for group [12]

[V] Variable [S1M300] is correct!
[V] Variable [S1PC] is correct!
[V] Variable [S2M300] is correct!
[V] Variable [S2PC] is correct!
[V] Variable [S1M301] is correct!
[V] Variable [S1AC] is correct!
[V] Variable [S2M301] is correct!
[V] Variable [S2AC] is correct!
[V] Variable [S1M302] is correct!
[V] Variable [S1IR] is correct!
[V] Variable [S2M302] is correct!
[V] Variable [S2IR] is correct!
[V] Variable [S1M900] is correct!
[V] Variable [S2M900] is correct!
[V] Variable [S1M901] is correct!
[V] Variable [S2M901] is correct!
[V] Variable [S1M902] is correct!
[V] Variable [S2M902] is correct!
[V] Variable [S1IO005] is correct!
[V] Variable [S2IO005] is correct!
[V] Variable [S1IO006] is correct!
[V] Variable [S2IO006] is correct!
[V] Variable [S3M300] is correct!
[V] Variable [S3PC] is correct!
[V] Variable [S4M300] is correct!
[V] Variable [S4PC] is correct!
[V] Variable [S3M301] is correct!
[V] Variable [S3AC] is correct!
[V] Variable [S4M301] is correct!
[V] Variable [S4AC] is correct!
[V] Variable [S3M302] is correct!
[V] Variable [S3IR] is correct!
[V] Variable [S4M302] is correct!
[V] Variable [S4IR] is correct!
[V] Variable [S3M900] is correct!
[V] Variable [S4M900] is correct!
[V] Variable [S3M901] is correct!
[V] Variable [S4M901] is correct!
[V] Variable [S3M902] is correct!
[V] Variable [S4M902] is correct!
[V] Variable [S3IO005] is correct!
[V] Variable [S4IO005] is correct!
[V] Variable [S3IO006] is correct!
[V] Variable [S4IO006] is correct!
[V] Variable [S5M300] is correct!
[V] Variable [S5PC] is correct!
[V] Variable [S6M300] is correct!
[V] Variable [S6PC] is correct!
[V] Variable [S5M301] is correct!
[V] Variable [S5AC] is correct!
[V] Variable [S6M301] is correct!
[V] Variable [S6AC] is correct!
[V] Variable [S5M302] is correct!
[V] Variable [S5IR] is correct!
[V] Variable [S6M302] is correct!
[V] Variable [S6IR] is correct!
[V] Variable [S5M900] is correct!
[V] Variable [S6M900] is correct!
[V] Variable [S5M901] is correct!
[V] Variable [S6M901] is correct!
[V] Variable [S5M902] is correct!
[V] Variable [S6M902] is correct!
[V] Variable [S5IO005] is correct!
[V] Variable [S6IO005] is correct!
[V] Variable [S5IO006] is correct!
[V] Variable [S6IO006] is correct!
[V] Variable [CacheMissTime] is correct!
[V] Variable [MAT75] is correct!
[V] Variable [MAT85] is correct!
[V] Variable [MAT95] is correct!

Group [12] score for task[1]: [100.000000%]
Your current [100.000000%] score is NOT group's best [100.000000%]. Your result will NOT be saved.
```

*Figure 5*

## *WEEK 2*

### *TASK 2.1*

### *Introduction/Explanation:*

Assembly language is one of the low-level programming language. It is often referred to as a "symbolic representation" of machine code, as it uses human-readable text to represent the binary machine code that a computer can understand. Assembly language allows programmers to write instructions that are specific to a particular computer architecture, making it a highly efficient way to develop software that requires direct hardware access or that needs to run at high speeds. While it is not as portable or flexible as higher-level programming languages, assembly language remains an important tool for developers who need to optimize performance-critical code. (Rodríguez, 2007)

A quadrilateral with at least one pair of parallel sides is referred to as a trapezoid. In Euclidean geometry, a trapezoid is a convex quadrilateral by definition. The bases of the trapezoid are the parallel sides. (Keedy, 1966)

In this task, we have been requested to write part of the program that calculates area of trapezoid of the parameters.

Where the parameters are;
- shorter side a=3,

- longer side b=7,

- height h=4.

$$\boxed{\boldsymbol{result} = \frac{a+b}{2} x\ h}$$

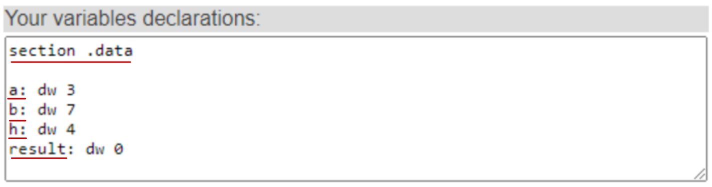
With the given formula above we can calculate the area of trapezoid.
Assembly language, type of low-level computer programming language consisting mostly of symbolic equivalents of a particular computer's machine language. (Gregersen, 2023)

In an assembly language program, a label is simply a name for an address. (``x'' is a name for the address of a memory location)

In the first part where we declare the variables, The data section is used for declaring initialized data or constants. This data does not change at runtime. Declaring data section syntax is "section .data".

In this task in the data declaration section, we are using "dw" - >Define word (allocates 2 bytes – 16 bits). Each byte of character is stored as its ASCII value in hexadecimal.

In the first part I declare the parameters. Where a is the shorter side (a=3), b is the longer side (b=7) and h is the height (h=4). Also, we initialise the result variable and assign 0. (it can be any integer). In here I say that result will start from 0.



```
section .data

a: dw 3
b: dw 7
h: dw 4
result: dw 0
```

Figure 6

In the section called “Your code to calculate the equation”, we are writing the instructions that will do the requested calculation.

The text section is used for keeping the actual code. That’s why I started with “section .data”.

1) We assign variable a to ax (accumulator)with command “mov”, 3 is stored in ax now.
*In the syntax I have given( mov ax, word [a]) in order to make It clear, but the assembly language program also works without “word”. Because we already declared that a,b,h and result has Define word (2 byte-16 bit storage space) in the variable declaration section.

2) Then we add variable b to ax (accumulator)with command called “add”, a + b is going to store at ax.

3) Then I assigned the value of 2, to the bx. Which is base register. As we remember formula was (a+b / 2) * h. By doing that we have assign the value for divisor.

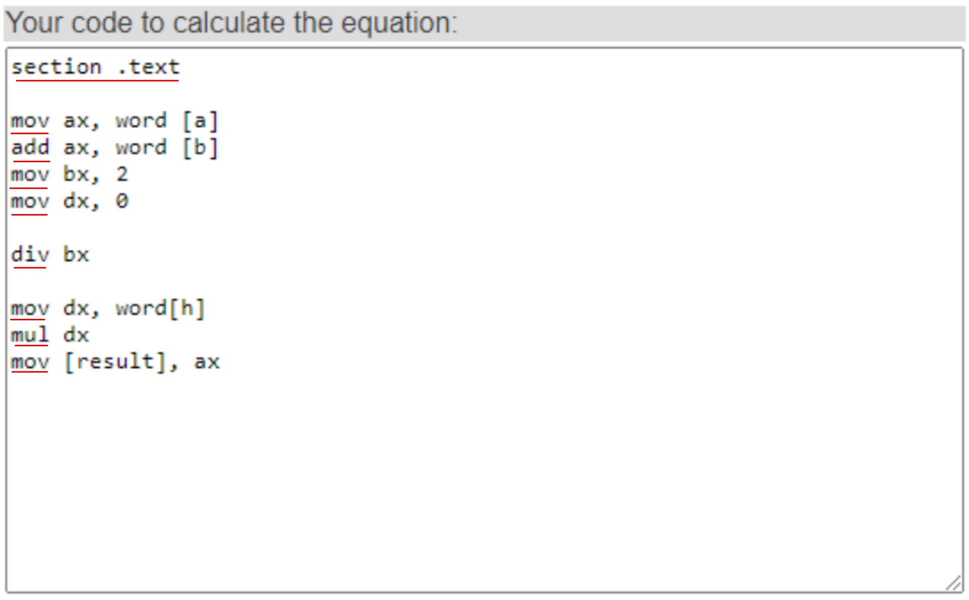
4) We have assign 0 to dx (data register), because dx is going to store the reminder. In order to not have any interference, Dx is going to clean before division.

5) Now we are going to divide ax by the bx. (div bx), where accumulator divided by the base.

6) We assign the parameter h (which is height = 4) to the dx (data register) and store in dx.

7) In this stage where we said “mul dx”, our program is multiplying dx with ax, because previously our result was stored in ax (accumulator). Again result is going to stored in ax (accumulator).

8) In the last stage we assign the ax (accumulator which is our final result now) to the variable called result. So ax is going to store in [results].


Your code to calculate the equation:

```
section .text

mov ax, word [a]
add ax, word [b]
mov bx, 2
mov dx, 0

div bx

mov dx, word[h]
mul dx
mov [result], ax
```

Figure 7

## *Conclusion:*

Since assembly language allows programmers to directly access and alter a computer's low-level hardware resources, it is a suitable language for designing applications that require direct access to the underlying hardware. Assembly language is particularly beneficial for designing scientific and technical applications that demand extensive number crunching due to its ability to conduct complicated arithmetic operations at rapid rates. While assembly code is more difficult to understand and write than higher-level programming languages, the speed and efficiency advantages it may accomplish make it a crucial talent for developers who need to optimise performance-critical programmes. Overall, assembly language with math computations remains a significant tool in programmers' and developers' toolboxes. (Gregersen, 2023)

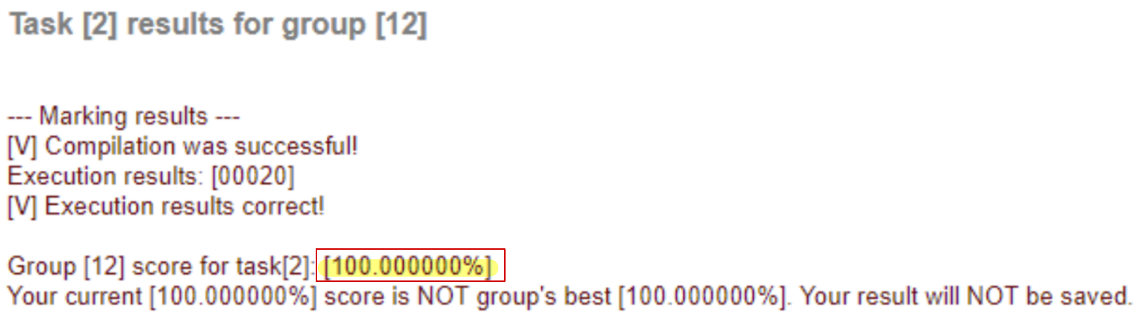
## *What I learned:*

While I was doing this task for week 2, I have learned;

- How to declare variables in Assembly language
- How to calculate area of trapezoid
- How to write instructions that will do the requested calculations (Instructions in assembly)

Once I learn the logic of storing, moving, shifting the data in assembly it was easy to calculate the area of Trapezoid. Also I find really useful that how we can implement the mathematical formulas into assembly language.

RESULT:



```
Task [2] results for group [12]

--- Marking results ---
[V] Compilation was successful!
Execution results: [00020]
[V] Execution results correct!

Group [12] score for task[2]: [100.000000%]
Your current [100.000000%] score is NOT group's best [100.000000%]. Your result will NOT be saved.
```

Figure 8

# WEEK 3

## Task 3.1

### Introduction/Explanation:

This script is calculating the basic arithmetic operations (addition, subtraction, multiplication, division, and power) based on the third command line argument. The first two arguments are the operands, and the third argument is the operator. If the third argument is "+", the script performs addition and prints the result. Similarly, if the third argument is "-", the script performs subtraction and prints the result, and so on for the other operations. The result of each operation is stored in a variable, and the echo command is used to print the value of the result.

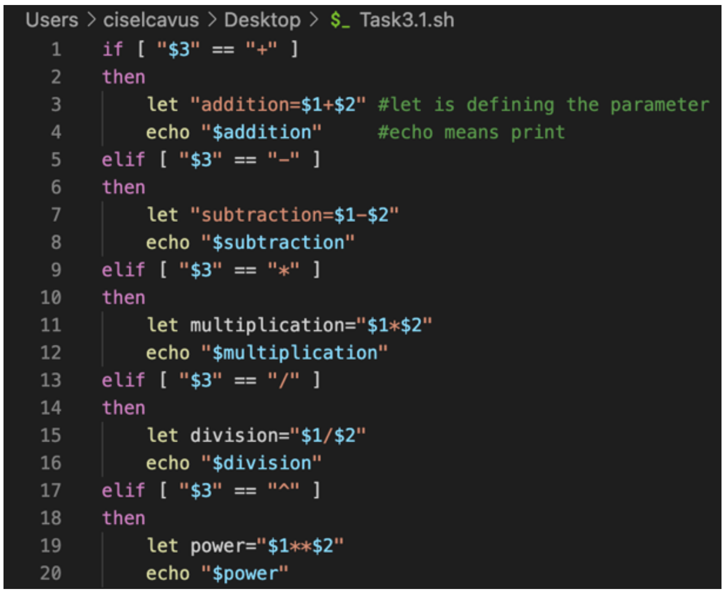
Example for addition operation;

This script using the if statement to check the value of the third command line argument which is “$3” in this case(stored in the shell variable `$3`).

If “$3” is equal to "+", the script performs addition by using the let command to assign the sum of “$1” and “$2” to the variable addition.

If “$3” doesn't match any of the values in the if statement, the script can not do anything and finish.

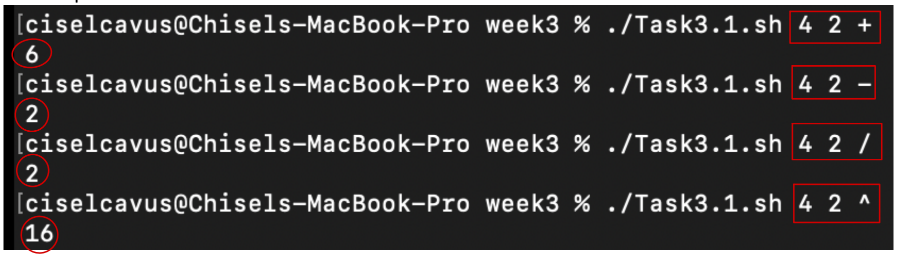
So in summary, the script takes three command line arguments, performs the specified arithmetic operation based on the third argument, and prints the result to the console.



```
Users > ciselcavus > Desktop > $_ Task3.1.sh
if [ "$3" == "+" ]
then
    let "addition=$1+$2" #let is defining the parameter
    echo "$addition"     #echo means print
elif [ "$3" == "-" ]
then
    let "subtraction=$1-$2"
    echo "$subtraction"
elif [ "$3" == "*" ]
then
    let multiplication="$1*$2"
    echo "$multiplication"
elif [ "$3" == "/" ]
then
    let division="$1/$2"
    echo "$division"
elif [ "$3" == "^" ]
then
    let power="$1**$2"
    echo "$power"
```

Figure 9

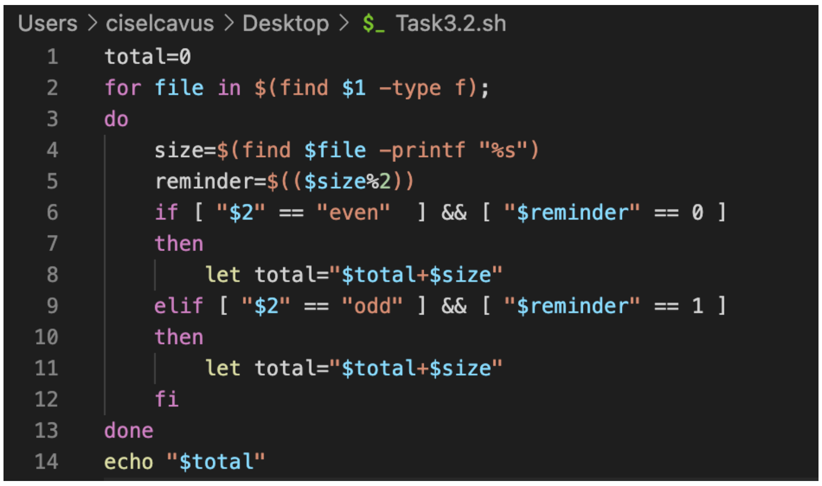
Example of the result:



```
[ciselcavus@Chisels-MacBook-Pro week3 % ./Task3.1.sh 4 2 +
6
[ciselcavus@Chisels-MacBook-Pro week3 % ./Task3.1.sh 4 2 -
2
[ciselcavus@Chisels-MacBook-Pro week3 % ./Task3.1.sh 4 2 /
2
[ciselcavus@Chisels-MacBook-Pro week3 % ./Task3.1.sh 4 2 ^
16
```

*Figure 10*

## *TASK 3.2*



Users > ciselcavus > Desktop > $_ Task3.2.sh

```
total=0
for file in $(find $1 -type f);
do
    size=$(find $file -printf "%s")
    reminder=$(($size%2))
    if [ "$2" == "even"  ] && [ "$reminder" == 0 ]
    then
        let total="$total+$size"
    elif [ "$2" == "odd" ] && [ "$reminder" == 1 ]
    then
        let total="$total+$size"
    fi
done
echo "$total"
```

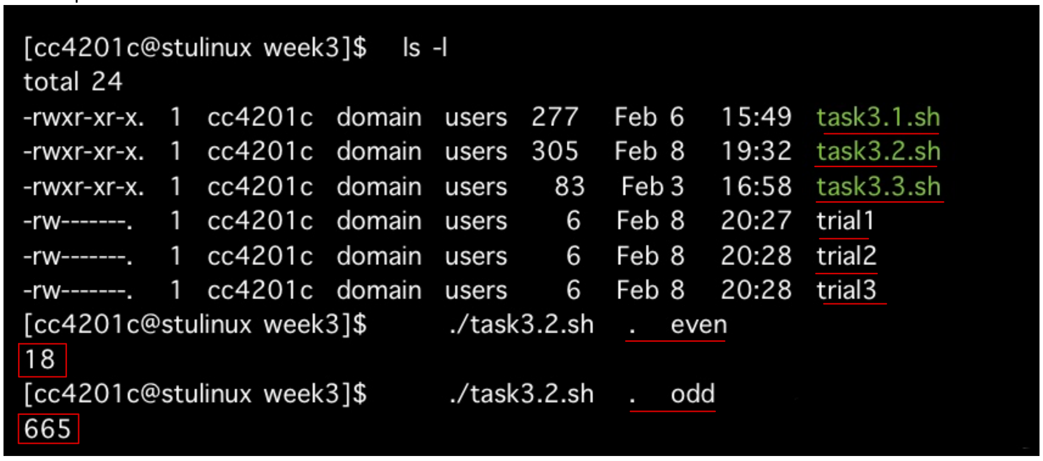
*Figure 11*

This script calculates the total size of files in a directory (specified by the first command line argument, “$1”) based on the parity of their size.
The second command line argument, `$2`, specifies whether to include “even” or “odd”-sized files in the total.

1. I initialized the variable called ‘**total**’ to 0.
2. The script uses a “**for** loop” to iterate over all files in the directory specified by “**$1”**, using the “**find”** command.
3. For each file, the size of the file is calculated using the **find** command with the **-printf** option. The “**%s”** format specifier is used to print the size of the file in bytes. The result is stored in the variable **size**.
4. The remainder of the file size divided by 2 is calculated and stored in the variable “**reminder”**.
5. The script then uses an **if** statement to check the value of “**$2”**.
6. If “**$2”** is equal to **"even"** and **reminder** is equal to 0, the script adds the size of the file to the **total** using the “**let”** command.
7. If “**$2”** is equal to **"odd"** and **reminder** is equal to 1, the script adds the size of the file to the **total** using the “**let”** command.
8. The loop repeats for each file in the directory specified by “**$1”**.
9. After all files have been processed, the script uses the “**echo”** command to print the final value of “**total”**, which represents the sum of the sizes of the files that match the specified parity.

In summary, the script takes two command line arguments: the name of a directory and whether to include even or odd-sized files. It calculates the total size of the files in the directory based on the parity of their size, and prints the result to the console.

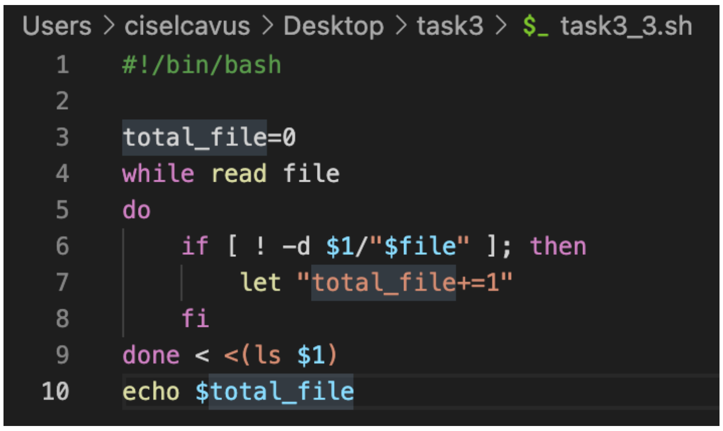
Example of result:



```
[cc4201c@stulinux week3]$   ls -l
total 24
-rwxr-xr-x.  1  cc4201c  domain  users  277   Feb 6   15:49  task3.1.sh
-rwxr-xr-x.  1  cc4201c  domain  users  305   Feb 8   19:32  task3.2.sh
-rwxr-xr-x.  1  cc4201c  domain  users    83   Feb 3   16:58  task3.3.sh
-rw-------.  1  cc4201c  domain  users     6   Feb 8   20:27  trial1
-rw-------.  1  cc4201c  domain  users     6   Feb 8   20:28  trial2
-rw-------.  1  cc4201c  domain  users     6   Feb 8   20:28  trial3
[cc4201c@stulinux week3]$      ./task3.2.sh   .   even
18
[cc4201c@stulinux week3]$      ./task3.2.sh   .   odd
665
```

Figure 12

## *TASK 3.3*



Users > ciselcavus > Desktop > task3 > $_ task3_3.sh

```bash
#!/bin/bash

total_file=0
while read file
do
    if [ ! -d $1/"$file" ]; then
        let "total_file+=1"
    fi
done < <(ls $1)
echo $total_file
```

Figure 13

This script calculates the number of regular files in a directory and outputs the calculation of  the result.
Let's have a look at the structure(How it works?);

- I initialized to the variable called "**total_file**" is to 0 to start the program.

-The script uses a **while** loop and the **read** command to iterate over the list of files in the directory specified by "**$1**". The **ls** command is used to list the contents of the directory.

-For each file, the script uses an **if** statement to check if the file is a directory. This is done using the "**-d**" option of the **[** (test) command, which returns true if the specified file is a directory.

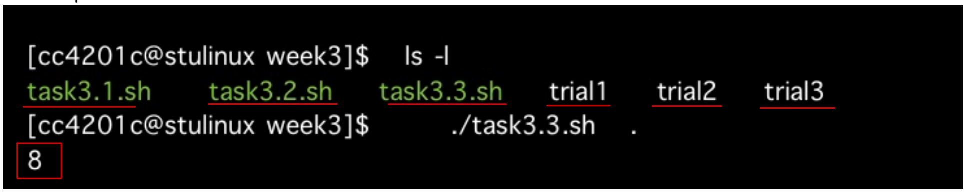
-If the file is not a directory, the script increments the value of `**total_file**` by 1 using the `**let**` command.

-The loop repeats for each file in the directory specified by `**$1**`.

-After all files have been processed, the script uses the **echo** command to print the final value of `**total_file**`, which represents the number of regular files in the directory.

So, the script takes one command line argument: the name of a directory. It calculates the number of regular files in the directory and prints the result to the console.

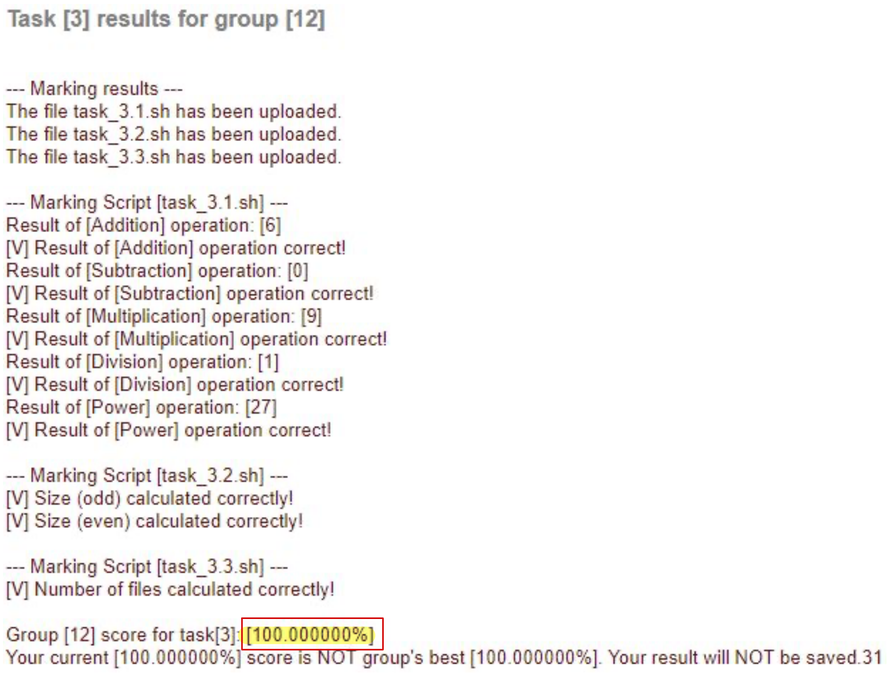
Example of result:



```
[cc4201c@stulinux week3]$   ls -l
task3.1.sh    task3.2.sh    task3.3.sh    trial1    trial2    trial3
[cc4201c@stulinux week3]$     ./task3.3.sh   .
8
```

Figure 14

RESULTS:



```
Task [3] results for group [12]

--- Marking results ---
The file task_3.1.sh has been uploaded.
The file task_3.2.sh has been uploaded.
The file task_3.3.sh has been uploaded.

--- Marking Script [task_3.1.sh] ---
Result of [Addition] operation: [6]
[V] Result of [Addition] operation correct!
Result of [Subtraction] operation: [0]
[V] Result of [Subtraction] operation correct!
Result of [Multiplication] operation: [9]
[V] Result of [Multiplication] operation correct!
Result of [Division] operation: [1]
[V] Result of [Division] operation correct!
Result of [Power] operation: [27]
[V] Result of [Power] operation correct!

--- Marking Script [task_3.2.sh] ---
[V] Size (odd) calculated correctly!
[V] Size (even) calculated correctly!

--- Marking Script [task_3.3.sh] ---
[V] Number of files calculated correctly!

Group [12] score for task[3]: [100.000000%]
Your current [100.000000%] score is NOT group's best [100.000000%]. Your result will NOT be saved.31
```

Figure 15

## *Conclusion:*

Assembly language with math calculations is a strong and efficient tool for designing software that requires high-performance computations. Assembly language is appropriate for applications that demand intensive number crunching due to its ability to directly access hardware resources and conduct complicated mathematical operations at rapid rates. Assembly language, while more difficult to understand and write than higher-level languages, is nonetheless a crucial ability for developers who need to optimise performance-critical code. Overall, assembly language with math calculations is a great tool for engineers who need to get the most performance from a particular system.

## *What I learned:*

While I was working on this task, I have learned how to use script also, I have worked on the if/else conditions. I clearly understand how to perform basic arithmetic calculations based on the third command line argument. At the same time, I have learned about the some terms and how to use them. For example “read”, “echo”, “let” etc. Currently I have clear understanding about the how to write script and I understand the basic structure.

# WEEK 4

## Introduction/Explanation:

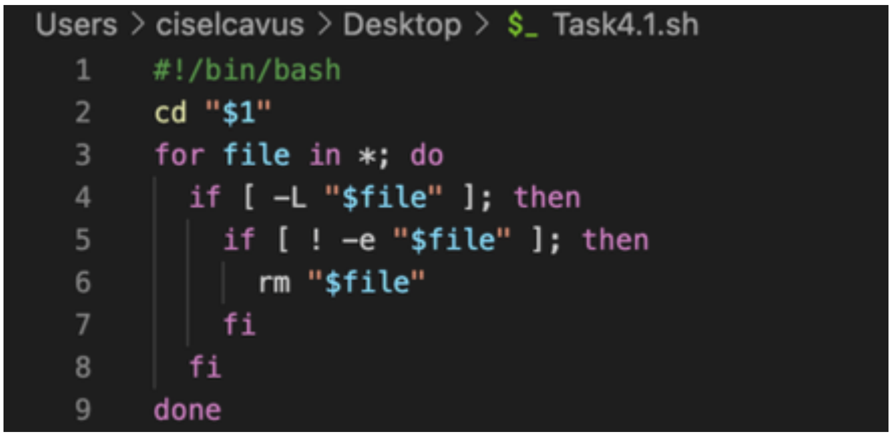
A file type called a symbolic link, commonly referred to as a soft link, serves as a pointer to another file or directory in the file system. It enables you to make a reference to a file or directory that is situated somewhere else than the symbolic link's origin. Symbolic links are widely employed to generate shortcuts or aliases for frequently visited files or directories that are housed in separate directories. Moreover, they can be utilised to support legacy programmes that need a certain file location.
A symbolic link points to a file by its path name rather than creating a direct link to it like a hard link does. As a result, if the original file is moved or deleted, the symbolic link will no longer work. In systems based on Unix, the ln command can be used to generate symbolic links. (Saverio Perugini et al., 2007)

## TASK4.1

I wrote this script required to take in one argument as input, which is going to be the name of a directory. The script then changes the current working directory to the directory specified by the input argument using the "cd" command. We have learned that in previous week. (Task4.1.sh)

The script then loops through each file in the current directory using a "for" loop with "*" as the pattern to match all files. I told the script to check for each file, the script first checks if the file is a symbolic link using the "-L" option in the "if" statement. If the file is a symbolic link, the script checks if the target file of the symbolic link exists using the "-e" option. If the target file does not exist, then the script removes the symbolic link using the "rm" command. Also as a reminder with ln -s command we can create a symbolic link on files as we requested on Task.

In summary, this script removes all broken symbolic links in the specified directory by checking if each file is a symbolic link and if its target file exists. If the target file does not exist, the symbolic link is going to removed. It has been great exercises to understand the lecture as well. (Symbolic Links)


Users > ciselcavus > Desktop > $_ Task4.1.sh

```bash
#!/bin/bash
cd "$1"
for file in *; do
  if [ -L "$file" ]; then
    if [ ! -e "$file" ]; then
      rm "$file"
    fi
  fi
done
```

Figure 16

## *TASK4.2*

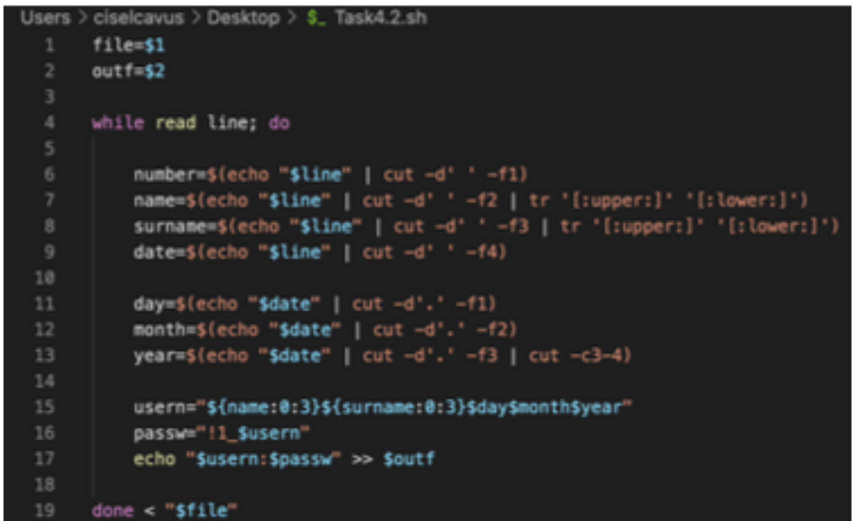
In this script we want to get 2 arguments as an input. The first argument is the name of a file containing a list of lines with the format "number name surname date", where "number" is a numerical value (int), we are getting the number because in the given example format the string is starting with numerical value, "name" and "surname" are going to be strings, and "date" is a date with the format "DD.MM.YYYY" in our script. The second argument is the name of the output file where the script will write the generated usernames and passwords.

The script reads each line of the input file as you can see the picture below, using a while loop, and for each line, it extracts the "number", "name", "surname", and "date" functions using the cut command with space and dot as delimiters. Then, it converts the "name" and "surname" to lowercase using the "tr" command we do this because in given format name and surname is starting with Capital letter. Sometimes it is called as error handling as well.

Next, the script extracts the day, month, and year from the "date" field using the cut command with the dot as the delimiter. The year is further reduced to the last two digits using the cut command with -c3-4 option as we requested to get the format as "DD.MM.YY(where YY is the last 2 digits in year).

Finally, the script generates a username by concatenating the first three characters of the lowercase "name", the first three characters of the lowercase "surname", and the day, month, and year extracted from the "date" field. It also generating a password by appending "!1_" to the username. This has been requested in the Task. The script then writes the generated username and password to the output file.

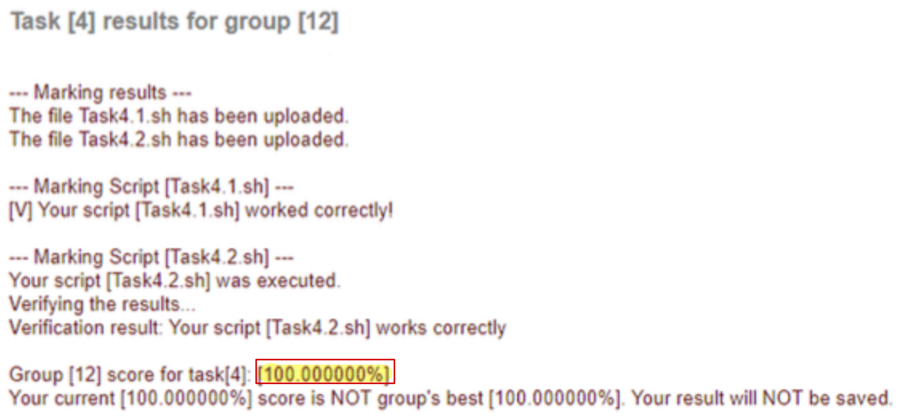
In summary, this script takes in a file containing a list of user information and generates a username and password for each user, which are then written to an output file.


Users > ciselcavus > Desktop > $_ Task4.2.sh

```
file=$1
outf=$2

while read line; do

    number=$(echo "$line" | cut -d' ' -f1)
    name=$(echo "$line" | cut -d' ' -f2 | tr '[:upper:]' '[:lower:]')
    surname=$(echo "$line" | cut -d' ' -f3 | tr '[:upper:]' '[:lower:]')
    date=$(echo "$line" | cut -d' ' -f4)

    day=$(echo "$date" | cut -d'.' -f1)
    month=$(echo "$date" | cut -d'.' -f2)
    year=$(echo "$date" | cut -d'.' -f3 | cut -c3-4)

    usern="${name:0:3}${surname:0:3}$day$month$year"
    passw="!1_$usern"
    echo "$usern:$passw" >> $outf

done < "$file"
```

Figure 17

RESULTS:



```
Task [4] results for group [12]

--- Marking results ---
The file Task4.1.sh has been uploaded.
The file Task4.2.sh has been uploaded.

--- Marking Script [Task4.1.sh] ---
[V] Your script [Task4.1.sh] worked correctly!

--- Marking Script [Task4.2.sh] ---
Your script [Task4.2.sh] was executed.
Verifying the results...
Verification result: Your script [Task4.2.sh] works correctly

Group [12] score for task[4]: [100.000000%]
Your current [100.000000%] score is NOT group's best [100.000000%]. Your result will NOT be saved.
```

*Figure 18*

## ***Conclusion:***

Symbolic links, also known as soft links, are a form of file that acts as a reference to another file or directory in the file system. They are widely used to create shortcuts or aliases for frequently accessed files or folders, as well as to support legacy programmes that need a certain file location. Symbolic links, as opposed to hard links, point to a file by its path name rather than forming a direct link to it. While symbolic links may be formed in Unix-based systems using the ln command, they will no longer function if the originating file is relocated or removed. Generally, symbolic links are a convenient technique to access files and directories in various parts of a file system.

## ***What I learned:***

According to lectures, I learned that soft links and symbolic links are shortcuts that guide the user to a different file or directory on the system. By avoiding duplicate files, this enables more effective file management and can free up disc space.

Contrarily, scripting refers to the use of computer languages to streamline and automate procedures. These can range from straightforward shell scripts to intricate applications that automate whole workflows.

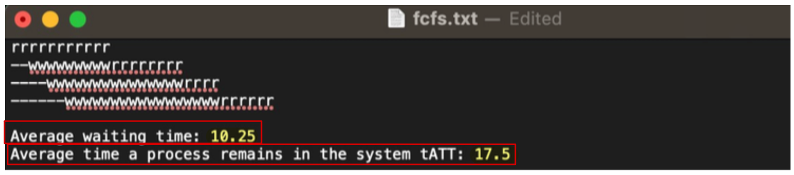
Scripting and symbolic links can help programmers and system administrators work faster and more efficiently by automating routine activities and streamlining file management. Developers can concentrate on the more crucial areas of their work and save time by employing these tools properly. Also, I learned the logic of the importance of Operating Systems part on it.

# *WEEK 5*

## *Introduction/Explanation:*

There are two types of famous CPU scheduling algorithms which is used in Computer Operating Systems. Which are FCFS (First Come, First Served) also known as FIFO (First In First Out) and SJN (Shortest Job next). The scheduler is executes the jobs to completion in arrival order. Also if we look at the early FCFS schedulers, the job did give up the CPU even when it was doing I/O(Input/Output).

## *TASK 5.1*



```
rrrrrrrrrrr
--wwwwwwwwwrrrrrrrr
----wwwwwwwwwwwwwwwrrrr
------wwwwwwwwwwwwwwwwwrrrrrr

Average waiting time: 10.25
Average time a process remains in the system tATT: 17.5
```

Figure 19

The processes for First Come First Served (FCFS) are carried out in the order they come. Process P1 arrives at time 0 and runs for 11 milliseconds until completion. Right after process P2 arrives at time 2 and has to wait for 11 milliseconds (the time P1 is running) before it can start executing. If we look at the Process P3 it is arriving at the time 4 and it should wait for 19 milliseconds (P1 and P2 are running in the time), it is before it starts executing. At the end process  P4  arriving at time 6 and should wait for 23 milliseconds (the time P1, P2 and P3 are running) before it can start executing.

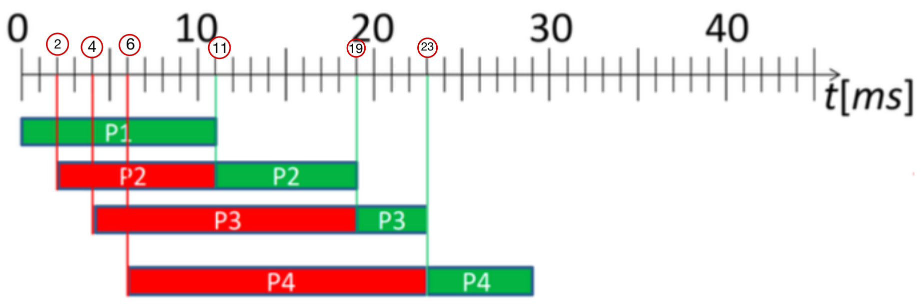


Figure 20

The average waiting time (tAWT) for FCFS is calculated as the sum of waiting times for all processes divided by the total number of processes:

tAWT = (0+9+15+17) / 4 = 41

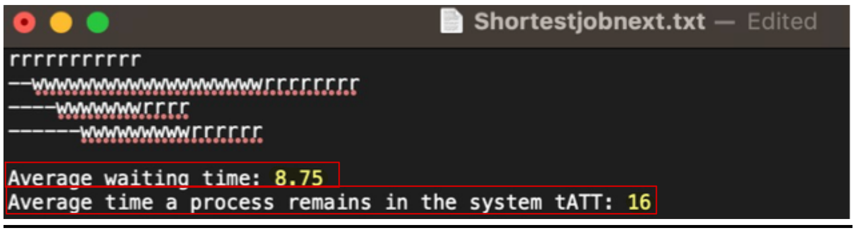
⇨ 41 / 4 = 10.25ms

The average time a process remains in the system (tATT) for FCFS is calculated as the sum of the completion times for all processes divided by the total number of processes:

tATT = (11+17+19+23) / 4 = 70

⇨ 70 / 4 = 17.5ms

## *TASK 5.2*



```
rrrrrrrrrrr
--wwwwwwwwwwwwwwwwwwwrrrrrrrr
----wwwwwwwrrrr
------wwwwwwwwwrrrrrr

Average waiting time: 8.75
Average time a process remains in the system tATT: 16
```

Figure 21

For Shortest Job Next (SJN), the processes are executed in order of increasing service time. The shortest service time is Process P3, so it runs first. After that, P2 starts running (which has shorter service time than P4). Once P2 is finished, P4 starts running, and finally, P1 runs last.

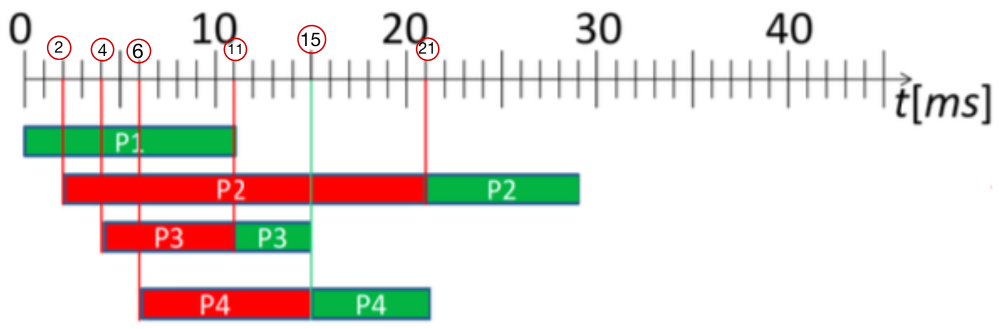


Figure 22

The average waiting time (tAWT) for SJN is calculated in the same way as for FCFS:

tAWT = (0+19+7+9) / 4 = 35

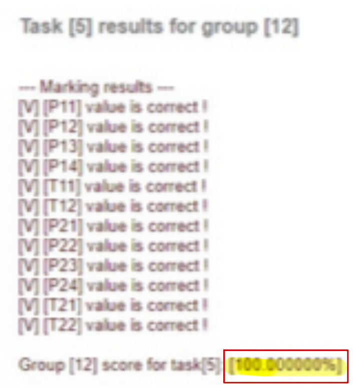
⇨ 35 / 4 = 8.75

The average time a process remains in the system (tATT) for SJN is also calculated in the same way as for FCFS:

tATT = (11+27+11+15) / 4 = 64

⇨ 64 / 4 = 16ms

RESULTS:



```
Task [5] results for group [12]

--- Marking results ---
[V] [P11] value is correct !
[V] [P12] value is correct !
[V] [P13] value is correct !
[V] [P14] value is correct !
[V] [T11] value is correct !
[V] [T12] value is correct !
[V] [P21] value is correct !
[V] [P22] value is correct !
[V] [P23] value is correct !
[V] [P24] value is correct !
[V] [T21] value is correct !
[V] [T22] value is correct !

Group [12] score for task[5] [100.000000%]
```

*Figure 23*

## *Conclusion:*

In conclusion, FCFS performs tasks in the order in which they arrive, regardless of their duration. SJN, on the other hand, arranges jobs depending on processing time, starting with the shortest. While both algorithms have merits and faults, SJN is usually seen to be more efficient and fair than FCFS since it reduces the average waiting time for jobs. Moreover, early versions of FCFS schedulers did not enable workloads to release the CPU even when conducting I/O activities, resulting in low CPU use. Generally, the scheduling algorithm used is determined by the system's unique requirements and the trade-off between efficiency and fairness.

## *What I learned:*

I gained knowledge of how various scheduling strategies affect a computer system's performance by studying scheduling algorithms. This information can be used to enhance scheduling procedures and raise a system's general effectiveness. Although one must take into account, a number of aspects when choosing an effective scheduling algorithm, such as the workload of the system and the nature of the processes being conducted, learning about FCFS and SJN can also aid in the development of critical thinking abilities.

As a result, studying FCFS and SJN helped me develop my critical thinking abilities, optimise scheduling practises, and gain significant insights into how computer systems work.

# *WEEK 6*

## *Introduction/Explanation:*

In a computer operating system, memory management refers to the process of effectively allocating, using, and deallocating memory resources. Any computer system needs memory because it is utilised to store data and instructions that are currently being carried out by the processor. Memory management includes a number of responsibilities, including deallocation releasing memory that is no longer required by a process and memory allocation reserving a piece of memory for a process or application. Memory protection, which guarantees that each process can only access memory areas that have been assigned to it, and memory swapping, which entails moving data from memory to disc to free up memory resources, are examples of duties that fall under the category of memory management. A computer system's best performance and stability depend on effective memory management. A process or application that continues to use memory resources without releasing them is known as a memory leak and affects the stability and performance of the system. Improper memory management can cause this. (Tofte  et al., 2002)

## *TASK 6.1*

I have started with the initial memory allocation:

M1=9k

M2=13k

M3=52k

M4=18k

M5=15k

Next, a process P1 requests 36k of memory and is allocated to the M3 gap (52k).

So the updated memory allocation is became;

M1=9k

M2=13k

M3=16k (used), 36k (P1), 36k (unused),

M4=18k

M5=15k

According to that I found following results;

**Best Fit:**

- P2=12k:-> This can fit into M2 or M5,

  Both of which have 13k and 15k respectively. I choose M2 since it is the smallest gap that can fit the process.

- P3=10k:-> This can fit into M5 (15k).
- P4=3k:-> This can fit into M1 (9k).
- P5=8k:-> This can fit into M4 (18k). The final memory allocation using
- *Best Fit is: P1-M3, P2-M2, P3-M5, P4-M1, P5-M4*

**Worst Fit:**

- P2=12k: This can fit into M4 or M5,

  Both of which have 18k and 15k respectively. I choose M4 since it is the largest gap.

- P3=10k: This can fit into M5 (15k).
- P4=3k: This can fit into M2 (13k).
- P5=8k: This can fit into M1 (9k). So the final memory allocation using
- *Worst Fit is: P1-M3, P2-M4, P3-M5, P4-M2, P5-M1*

**First Fit:**

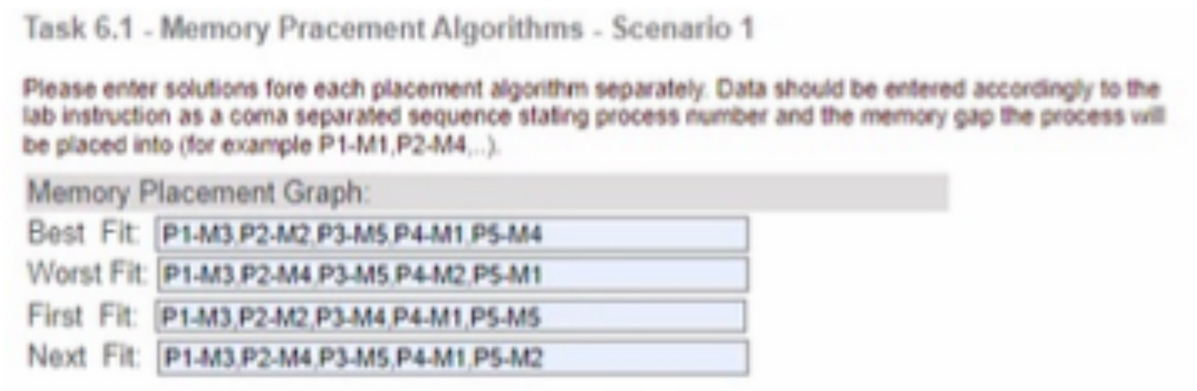
- P2=12k: This can fit into M2 (13k).
- P3=10k: This can fit into M4 (18k).
- P4=3k: This can fit into M1 (9k).
- P5=8k: This can fit into M5 (15k). So the final memory allocation using
- *First Fit is: P1-M3, P2-M2, P3-M4, P4-M1, P5-M5*

**Next Fit:**

- P2=12k: This can fit into M4 or M5,

  Both of which have 18k and 15k respectively. We choose M4 since it is the next gap after M3 in memory order.

- P3=10k: This can fit into M5 (15k).
- P4=3k: This can fit into M1 (9k).
- P5=8k: This can fit into M2 (13k). So the final memory allocation using
- *Next Fit is: P1-M3, P2-M4, P3-M5, P4-M1, P5-M2*



Task 6.1 - Memory Pracement Algorithms - Scenario 1

Please enter solutions fore each placement algorithm separately. Data should be entered accordingly to the lab instruction as a coma separated sequence stating process number and the memory gap the process will be placed into (for example P1-M1,P2-M4,...).

Memory Placement Graph:

| | |
|---|---|
| Best Fit: | P1-M3,P2-M2,P3-M5,P4-M1,P5-M4 |
| Worst Fit: | P1-M3,P2-M4,P3-M5,P4-M2,P5-M1 |
| First Fit: | P1-M3,P2-M2,P3-M4,P4-M1,P5-M5 |
| Next Fit: | P1-M3,P2-M4,P3-M5,P4-M1,P5-M2 |

*Figure 24*

## *TASK 6.2*

For the task 2, I did same process as well.

M1=9k

M2=13k

M3=52k

M4=18k

M5=15k

Next, a process P1 requests 36k of memory and is allocated to the M3 gap (52k).

So the updated memory allocation is:

M1=9k

M2=13k

M3=16k (used), 36k (P1), 36k (unused),

M4=18k

M5=15k

Best Fit:

- P2=15k: This can fit into M5 or the unused part of M3. I choose M5 since it is the smallest gap that can fit the process.
- P3=3k: This can fit into M1 (9k).
- P4=10k: This can fit into M2 (13k).
- P5=8k: This can fit into M4 (18k). The final memory allocation using
- Best Fit is: P1-M3, P2-M5, P3-M1, P4-M2, P5-M4

Worst Fit:

- P2=15k: This can fit into M5 (15k).
- P3=3k: This can fit into M5 or the unused part of M1. I choose M5 since it is the

  largest gap that can fit the process.

- P4=10k: This can fit into M2 (13k).
- P5=8k: This can fit into M1 (9k). So the final memory allocation using
- Worst Fit is: P1-M3, P2-M4, P3-M5, P4-M2, P5-M1

First Fit:

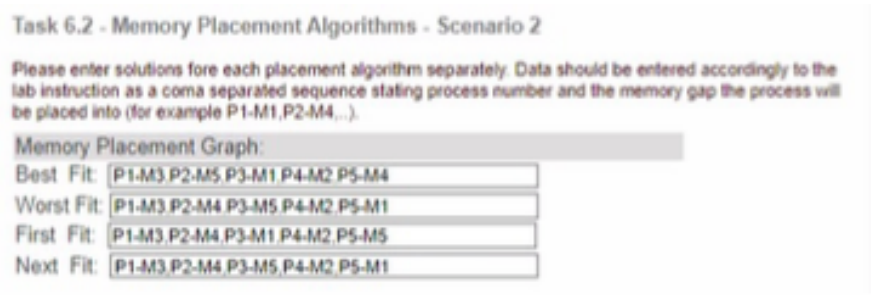
- P2=15k: This can fit into M4 (18k).
- P3=3k: This can fit into M1 (9k).
- P4=10k: This can fit into M2 (13k).
- P5=8k: This can fit into M5 (15k). So the final memory allocation using
- First Fit is: P1-M3, P2-M4, P3-M1, P4-M2, P5-M5

Next Fit:

- P2=15k: This can fit into M5 (15k).
- P3=3k: This can fit into M5 or the unused part of M1. We choose M5 since it is the

  next gap after M4 in memory order.

- P4=10k: This can fit into M2 (13k).
- P5=8k: This can fit into M1 (9k). So the final memory allocation using
- Next Fit is: P1-M3, P2-M5, P3-M5, P4-M2, P5-M1


Task 6.2 - Memory Placement Algorithms - Scenario 2

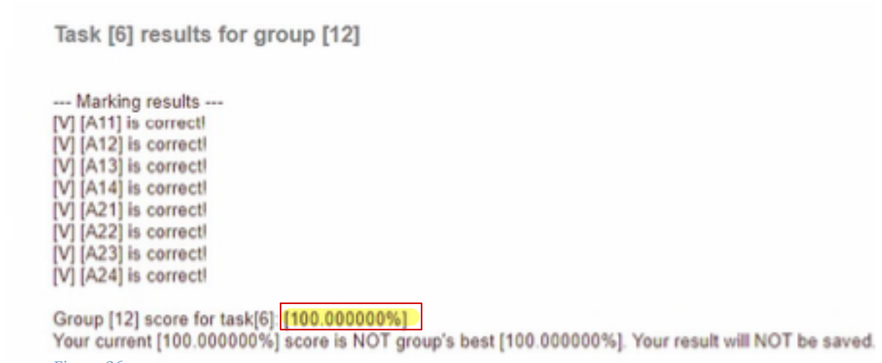
Please enter solutions fore each placement algorithm separately. Data should be entered accordingly to the lab instruction as a coma separated sequence stating process number and the memory gap the process will be placed into (for example P1-M1,P2-M4,...).

Memory Placement Graph:

| | |
|---|---|
| Best Fit: | P1-M3,P2-M5,P3-M1,P4-M2,P5-M4 |
| Worst Fit: | P1-M3,P2-M4,P3-M5,P4-M2,P5-M1 |
| First Fit: | P1-M3,P2-M4,P3-M1,P4-M2,P5-M5 |
| Next Fit: | P1-M3,P2-M4,P3-M5,P4-M2,P5-M1 |

Figure 25

RESULTS:



```
Task [6] results for group [12]

--- Marking results ---
[V] [A11] is correct!
[V] [A12] is correct!
[V] [A13] is correct!
[V] [A14] is correct!
[V] [A21] is correct!
[V] [A22] is correct!
[V] [A23] is correct!
[V] [A24] is correct!

Group [12] score for task[6]: [100.000000%]
Your current [100.000000%] score is NOT group's best [100.000000%]. Your result will NOT be saved.
```

*Figure 26*

## ***Conclusion:***

Memory management is an important part of a computer operating system that is in charge of assigning and managing memory resources. Memory management is crucial for ensuring that the system runs smoothly and efficiently. To manage memory resources while ensuring each application has access to the memory it requires, several approaches and algorithms including as paging, segmentation, and virtual memory are utilised. Overall, memory management is crucial to a computer system's stability, dependability, and performance. (Tofte  et al., 2002)

## ***What I learned:***

I have learned the value of making efficient and effective use of computer memory resources from my studies in memory management. Also, I understand how various memory allocation strategies affect a system's performance and stability. I have learned more about the difficulties of managing memory through my research, including fragmentation and allocation/deallocation overhead.

In addition, now I can clearly see how important memory management is to maintaining the stability and security of computer systems. Malicious actors may use memory-related vulnerabilities to execute malicious code or obtain unauthorised access. To create secure and dependable software systems, it is crucial to have a firm grasp of memory management principles and best practises.

Overall, my research on memory management has helped me better understand how computers function and how crucial effective resource management is. In order to create high-performance and safe software systems, I am eager to put this expertise to use in the future.

# *WEEK7*

## COMP1562 Operating Systems Week 7 Report

### Introduction:

Each computer user or system administrator must do regular disc management and maintenance on their system. To execute these tasks, many operating systems provide a plethora of tools and functions. The programmes fsck and scandisk are used in Linux and Windows systems, respectively, to inspect and repair discs. Similarly, Logical Volume Manager (LVM), Logical Disk Manager (LDM), and format handle disc management, whereas mkfs and format build file systems. Another device used to keep hard drives healthy is the Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) utility. This article will go through the qualities, similarities, and distinctions of these tools. (Bandel, 1997)

- ***fsck (Linux) vs scandisk (Windows):***

Tools for checking and fixing discs are called fsck and scandisk in Linux and Windows systems, respectively. They both carry out comparable tasks, yet they differ in some ways.
Linux uses the command-line tool fsck to scan and fix file system faults on storage devices. ext2, ext3, ext4, ReiserFS, and XFS are just a few of the many file systems that it may be used to inspect and fix. Fsck is a potent programme that can identify and fix a wide range of issues, including directory errors, corrupted inodes, and faulty sectors. It can also restore deleted files and directories. (Gjoen, 2002)

On the other hand, Windows systems employ the disc checking and repair utility scandisk. It is a graphical application that may be used from the command line or the File Explorer. Scandisk inspects the file system for mistakes like faulty sectors, files that are cross-linked, and directory issues, and fixes them. The user-friendly utility scandisk offers a straightforward user interface and detailed step-by-step instructions.

Despite the fact that fsck and scandisk perform the same functions, fsck is thought of as a more powerful and adaptable tool than scandisk. Unlike scandisk, which can only fix a few file systems, fsck additionally has the ability to recover deleted files and directories. Furthermore, fsck allows system administrators to select the repair options and, if necessary, skip particular steps, giving them additional control over the file system repair process. Nevertheless, scandisk does not provide a lot of control over the repair process and occasionally calls for the usage of other programmes to handle specific problem kinds. Fsck is a faster tool overall because it can finish the file system inspection and repair operation more quickly and effectively than scandisk. This is done so that the checks and fixes can be carried out more quickly and with less system resources thanks to fsck, a command-line software that doesn't need a graphical user interface. (Gjoen, 2002)

- ***mkfs (Linux) vs format (Windows):***

A new file system can be created on a storage device using the Linux command-line tool mkfs. ext2, ext3, ext4, ReiserFS, and XFS are just a few of the many file systems that may be made using it. With mkfs, users can personalise the formation of their file systems using a variety of choices and settings. The amount of inodes and blocks, as well as the file system's size and type, can all be customised by users. The disc formatting tool used in Windows computers, however, is called format. It is a graphical programme that is reachable via the command line or the File Explorer. FAT32, NTFS, and exFAT file systems can be created using format, which is used to build a new file system on a storage device. format is an approachable tool with an easy-to-use interface and detailed instructions for consumers to follow.

While mkfs and format both have similar roles to play, mkfs is regarded as a more potent and versatile tool than format. In addition to offering users more flexibility over the development of file systems, mkfs is able to create a wider variety of file systems. Furthermore, mkfs gives users more control over the file system settings and options, including the capacity to define the quantity of inodes and blocks, which can enhance the file system's performance. In contrast, format is a utility that is easier to use and more straightforward than mkfs. Users only need rudimentary knowledge of file systems and formatting, and it offers a step-by-step user interface. As a result, it is easier to use and more suitable for rapid formatting tasks or individuals with little experience. (Gjoen, 2002)

The ability of mkfs to correct file system faults in comparison to format's exclusive purpose of generating new file systems is another significant distinction between the two tools. So, unlike format, mkfs can be used to repair a file system if it gets corrupted or damaged.

- ***Logical Volume Manager (Linux) vs Logical Disk Manager (Windows).***

Users of Windows systems can control volumes on dynamic discs using LDM, a disc management tool. Similar to how LVM creates logical volumes, dynamic discs are a particular kind of disc that may be used in Windows to build volumes that span several drives. Users can make plain volumes, striped volumes, mirrored volumes, and RAID-5 volumes using LDM. LDM also gives customers the option to grow and shrink volumes instantly, without the need to restart the computer system. LDM can also be used to make snapshots, which are read-only copies of a disc at a certain moment. (Gjoen, 2002)

LDM's ability to integrate with Windows systems is one of its main benefits. LDM is a practical and well-liked disc management tool for Windows administrators because it is a built-in component of Windows and is offered with all releases of the operating system.
There are several significant variations between the two tools, even though LVM and LDM both offer comparable disc management functions. Their method of disc management is one of the most important variances. LVM is made to function with physical discs, whereas LDM is made to function with dynamic discs. ( Hummel, 2008)

Their degree of flexibility is a significant distinction between LVM and LDM. In addition to offering a variety of disc management functions, LVM is a very flexible technology that enables users to build logical volumes that span many drives. LVM is more flexible than LDM, however LDM is simpler to use and integrated with Windows systems. ( Hummel, 2008)

The ability to produce logical volume snapshots is one of the capabilities of LVM that is not offered by LDM. More file systems, such as ext2, ext3, ext4, and XFS, are supported with LVM than LDM.

- ***S.M.A.R.T. utility.***

For the purpose of preserving the functionality and health of hard drives on Linux and Windows, the S.M.A.R.T. programme is a useful tool. It can extend the life of drives, help with data recovery, and provide early indications of potential hazards. False positives and a low level of predictive power are some of its drawbacks. It is readily accessible on Linux, and third-party software can be used to access it on Windows.

A useful tool for keeping track of the health of hard drives and solid-state drives is the Self-Monitoring, Analysis, and Reporting Technology (SMART) programme. The main advantage of SMART is its capability to identify possible issues before they become serious, enabling users to take action to stop data loss. Furthermore, it offers current information on the drive's condition, assisting users in organising and carrying out maintenance procedures. Unfortunately, SMART is not perfect and occasionally misses impending failures. It can also produce false positives, which may worry users unnecessarily. Furthermore, some manufacturers only partially implement SMART, which limits its applicability under those circumstances. The SMART software is a useful tool for keeping an eye on and preserving the health of hard drives and SSDs generally, though. (Sivathanu et al., 2005)

## CONCULUSION:

Both the Linux and Windows operating systems have a number of tools and utilities that can assist with this task. While scandisk and format are straightforward and easy to use, more powerful and flexible applications, such as fsck and mkfs, provide you more control over file system repair and creation. Similarly, LDM and LVM are both useful disc management solutions; however, LVM provides greater flexibility in constructing logical volumes spanning many drives and making logical volume snapshots, whilst LDM is more tightly integrated with Windows systems. Finally, which tool to employ is determined by the unique requirements of the system being managed as well as the system administrator's preferences. System administrators can assure the continuing health and functionality of their computer systems by appropriately utilising these technologies.

## *TASK 7.1*

## *Introduction/Explanation:*

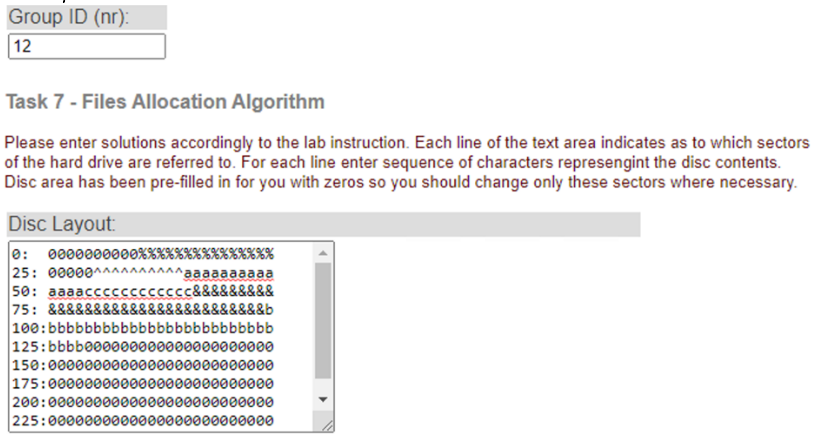
Operating systems utilize a particular kind of file allocation technique called contiguous file allocation to distribute storage space among files on a disc. Each file is given a contiguous block of storage space using this technique. This indicates that a file's allotted blocks are all situated close to one another, forming a single contiguous storage space. The operating system looks for a free block of contiguous space on the disc when a file is created that is big enough to hold the file. A directory or file allocation table, which the file system uses to keep track of the location of each file on the disc, stores the starting address and block length. (Zurich et al., 1991)

Contiguous file allocation has the advantage of being straightforward and efficient. Because file data may be read or written in a single sequential operation, the operating system just has to remember the beginning address and length of each file, which speeds up file access. The disadvantage of this method is that when files are added, updated, and removed over time, the disc may get fragmented. Locating continuous blocks of free space large enough to accommodate new files might be difficult since the available space may become fragmented when files are added or withdrawn from the disc. Since fragmentation can reduce the efficacy of the file system, disc defragmentation programmes may be required to increase disc performance. (Zurich et al., 1991)

This task is allocating 3 new files on a disk by using the contiguous file allocation algorithm. We have been given 3 Pre-allocates file which are;

File X: size 15B, content: '%', start block: 10
File Y: size 10B, content: '^', start block: 30
File Z: size 33B, content: '&', start block: 66

Also we have been given that File A size is 14B and content 'a', File B size is 30B and content 'b'. Finally File C size is 12B and content 'c'.


Group ID (nr):

12

Task 7 - Files Allocation Algorithm

Please enter solutions accordingly to the lab instruction. Each line of the text area indicates as to which sectors of the hard drive are referred to. For each line enter sequence of characters represengint the disc contents.
Disc area has been pre-filled in for you with zeros so you should change only these sectors where necessary.

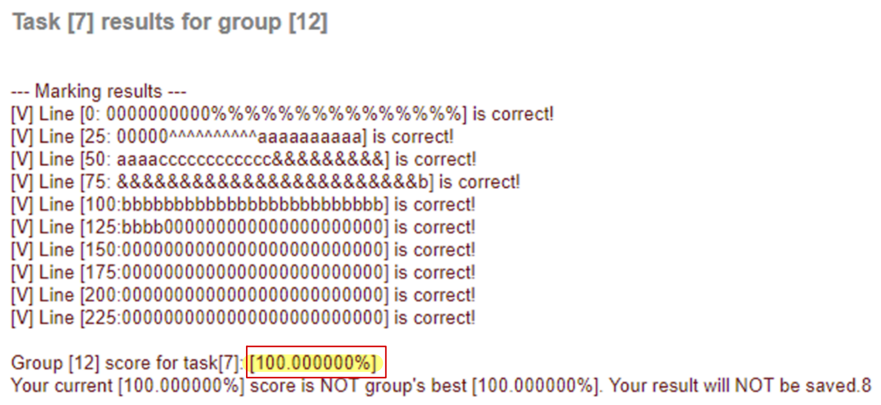
Disc Layout:

```
0:   0000000000%%%%%%%%%%%%%%%
25:  00000^^^^^^^^^^aaaaaaaaaa
50:  aaaacccccccccccc&&&&&&&&&
75:  &&&&&&&&&&&&&&&&&&&&&&&&b
100:bbbbbbbbbbbbbbbbbbbbbbbbb
125:bbbb000000000000000000000
150:0000000000000000000000000
175:0000000000000000000000000
200:0000000000000000000000000
225:0000000000000000000000000
```

*Figure 27*

So on the Disc Space, as we can see above Each line of the text area indicates as to which sectors of the hard drive are referred to. For ach line I provided sequence of characters representing the disc contents.

RESULT:



```
Task [7] results for group [12]


--- Marking results ---
[V] Line [0: 0000000000%%%%%%%%%%%%%%%] is correct!
[V] Line [25: 00000^^^^^^^^^^^aaaaaaaaa] is correct!
[V] Line [50: aaaacccccccccccc&&&&&&&&&] is correct!
[V] Line [75: &&&&&&&&&&&&&&&&&&&&&&&&b] is correct!
[V] Line [100:bbbbbbbbbbbbbbbbbbbbbbbbb] is correct!
[V] Line [125:bbbb000000000000000000000] is correct!
[V] Line [150:0000000000000000000000000] is correct!
[V] Line [175:0000000000000000000000000] is correct!
[V] Line [200:0000000000000000000000000] is correct!
[V] Line [225:0000000000000000000000000] is correct!

Group [12] score for task[7]: [100.000000%]
Your current [100.000000%] score is NOT group's best [100.000000%]. Your result will NOT be saved.8
```

*Figure 28*

## ***Conclusion:***

Finally, contiguous file allocation is a simple and effective method for sharing storage space across files on a disc. Because files are kept in a contiguous block of storage space, it streamlines file access. The disadvantage of this strategy is that when files are added, updated, and removed over time, the disc may get fragmented, causing performance concerns. This fragmentation can be resolved by rearranging the files on the drive and optimising the file system's effectiveness with disc defragmentation tools. Generally, the choice of file allocation technique is determined by the system's specific requirements and the trade-offs between simplicity and potential performance difficulties. (Zurich et al., 1991)

## ***What I learned:***

One of the strategies for allocating disc space for files is the contiguous file allocation algorithm, which I have learned about. Because the entire file is kept in a single continuous portion of the disc, this approach allocates files as contiguous blocks of disc space. I also learned how to represent disc space using a block diagram in this work, where each block is represented by a 0 if it is free space and by a character if it is a file. I saw an illustration of utilising this approach to allocate new files while taking into account the pre-existing files currently present on the disc. This challenge emphasises the significance of effective file allocation to guarantee optimal utilisation of disc space and prevent fragmentation.

# *WEEK 8*

## *Introduction/Explanation:*

One of the main duties of a Linux system administrator is to make sure the system runs efficiently and to solve any issues that might occur. To change system behaviour and configurations, this entails running the proper system commands and modifying system files. Defining system variables like PATH, limiting user logins, adding new users or groups, changing file permissions and access control lists, establishing network settings, managing system services, and many other jobs are examples of common tasks. As a result, Linux administration necessitates a thorough knowledge of system commands and file structures as well as the ability to identify and resolve intricate system problems.

ANSWERS FOR FOLLOWING QUESTIONS:

1. Which file you would use to redefine existing system variable PATH (provide full path)?

```
1.   /etc/bashrc
```

2. How would you redefine the **PATH** variable to add path to **/home/john/bin** directory?

```
2. export PATH=$PATH:/home/john/bin
```

3. Which file you would use to limit maximum number of logins for a user **mariusz** to 3 logins at a time (provide full path)?

```
3. /etc/security/limits.conf
```

4. How would you modify the above file to achieve the result (please show how the line you would add to the file would look like)?

```
4. mariusz hard maxlogins 3
```

5. You were requested to add to the system new user called **tatiana**. You want to execute the command adding the user to the system so that default shell for **tatiana** was be **/bin/sh** and her initial / primary group was **students**. How would the command look like?

```
5. useradd -s /bin/sh -g students Tatiana
```

6. One of the system users asked you to change his login name from **pm75** to **mariusz**. How would you achieve this (provide appropriate command)?

```
6. usermod -l mariusz pm75
```

7. You want to add user **mariusz** to ACL (Access Control List) of file **/usr/share/ccsm** so that this use had read permission (assume the user does NOT belong to the file group nor he is the file owner).

```
7. setfacl -m u:mariusz:r /usr/share/ccsm
```

8. You want to change current permissions for file **/usr/share/ccsm** to **rwxr--r--**. How the command allowing you to achieve the goal would look like.

```
8. chmod 744 /usr/share/ccsm
```

9. You want to check your file system supports ACL. Which command would tell you this?

```
9. ls -l
```

10. You want to create in the current directory symbolic link **ptr** to file **/usr/share/ccsm**. How the command allowing you to achieve this would look like?

```
10. ln -s /usr/share/ccsm ptr
```

11. One of the system users **clare** forgot her password. How the command initialising her password change would look like? Assume you execute the command as root (in other words: what command would root execute to change clare's password).

```
11. passwd clare
```

12. Which file you would modify to add a new DNS server (provide full path)?

```
12. /etc/resolv.conf
```

13. How would the entry (line) to the above file look like if the DNS server you wanted add to the system was **217.173.13.13**?

```
13. nameserver 217.173.13.13
```

14. You want to execute **/usr/share/ccsm** script **every Sunday at 2pm**. How the **crontab** entry allowing you to achieve this would look like?

```
14. 0 14 * * 7 /usr/share/ccsm
```

15. As system admin (root) you copied **/usr/share/ccsm** file to the home directory of user **ben**. Assuming user ben has his home directory in the default location for CentOS7.x system, show how you would make the file to be owned by **ben** and belong to system group **students**?

```
15. chown ben:students /home/ben/ccsm
```

16. You want immediately lock account for student **ben** (so that he was unable to login to the system) as it turned out he has fee hold status. How the command allowing you to achieve this would look like?

```
16. passwd -l ben
```

17. You want to check which process(es) running in the system is / are owned by user **ben**. How the command allowing you to achieve this may look like?

```
17. ps -u ben
```

18. Suppose you have just finished a completely fresh Linux installation. However, after first system boot it turned out that it by default takes you to the graphical user interface login prompt (meaning the system boots by default into **graphical.target**). Which command would list you all targets to choose from as an alternative to the **graphical.target**? (provide full command)?

```
18. systemctl list-units --type target
```

19. How you would change the default target to **multiuser.target** which allows multi user mode but with the graphical user interface not starting (provide full command)?

```
19. systemctl set-default multiuser.target
```

20. You want to change default location for the users' home directory from **/home** to **/home/users**. Which file you would use to make it a default setting (provide full path)?

```
20. /etc/default/useradd
```

21. How the entry (line) to the above file would look like.

```
21. HOME=/home/users
```

22. Assuming that your network interface name is **enp0s3**, you would like change default settings for this interface to enforce it to request IP address from a **DHCP** server. Which file you would use to force this (provide full path)?

```
22. /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

23. How the entry (line) in the above file would look like?

```
23. BOOTPROTO=dhcp
```

24. As normal / standard Linux system user **mariusz** you spotted that you r default shell is Korn shell **/bin/ksh**. You definitely prefer BASH **/bin/bash** shell over Korn shell (**/bin/ksh**) so you want it to get changed **permanently**. However, you do not want to engage system admin to achieve this as you are perfectly capable to do it on your own. Which file you would modify (provide full path assuming default location for the user's home directory)?

```
24. /home/mariusz/.bash_profile
```

25. How the line allowing you to change your default shell would look like.

```
25. export SHELL=/bin/bash
```

26. How would you interpret Heisenberg Uncertainty Principle in the context of wavelet analysis?

According to the Heisenberg Uncertainty Principle, it is difficult to correctly measure some pairs of physical variables at once, such as momentum and location. As a result, we can measure one of these values less precisely the more exactly we measure the first. The uncertainty principle is connected to the trade-off between time and frequency domain resolution in wavelet analysis. The wavelet transform breaks down a signal into various frequency components, and the frequency of the component being studied determines the wavelet transform's time resolution. The component's time duration in the signal will be shorter the higher its frequency, which will result in a reduced time resolution. (Bahri & Ashino, 2017)

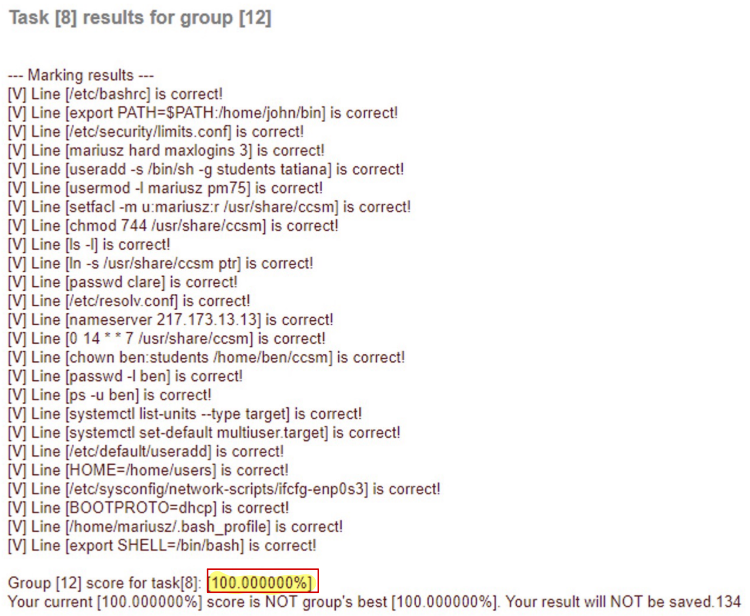
The Heisenberg Uncertainty Principle is analogous to this trade-off between time and frequency resolution in that it states that we may measure a signal's frequency with greater precision than its duration, and vice versa. This means that the Heisenberg Uncertainty Principle, a key characteristic of signal analysis, restricts our capacity to correctly quantify both the temporal and frequency characteristics of a signal at the same time.

In conclusion, wavelet analysis interprets the Heisenberg Uncertainty Principle as a restriction on the simultaneous measurement of time and frequency resolution of a signal. The idea emphasises a basic characteristic of signal analysis—the inherent trade-off between these two metrics. (Bahri & Ashino, 2017)

## ***What I learned:***

With this task I believe I clearly understand of the Linux operating system and its command line interface. Especially questions helped me a lot to understand of the logic. I have learned about modifying system variables, limiting user logins, adding and modifying user accounts, managing file permissions, creating symbolic links, and modifying system configuration files. While I was doing these questions, I had to do a lot of research about the commands and it helped me a lot to cover logic of it. I think these skills are essential for effective system administration and will enable me to manage Linux systems efficiently. Additionally, the last question suggests that I might have an interest in physics and the application of mathematical concepts in the analysis of signals and waveforms. With this questions I had to do huge research to understand the principles, because previously I didn't know it.

RESULT:



```
Task [8] results for group [12]

--- Marking results ---
[V] Line [/etc/bashrc] is correct!
[V] Line [export PATH=$PATH:/home/john/bin] is correct!
[V] Line [/etc/security/limits.conf] is correct!
[V] Line [mariusz hard maxlogins 3] is correct!
[V] Line [useradd -s /bin/sh -g students tatiana] is correct!
[V] Line [usermod -l mariusz pm75] is correct!
[V] Line [setfacl -m u:mariusz:r /usr/share/ccsm] is correct!
[V] Line [chmod 744 /usr/share/ccsm] is correct!
[V] Line [ls -l] is correct!
[V] Line [ln -s /usr/share/ccsm ptr] is correct!
[V] Line [passwd clare] is correct!
[V] Line [/etc/resolv.conf] is correct!
[V] Line [nameserver 217.173.13.13] is correct!
[V] Line [0 14 * * 7 /usr/share/ccsm] is correct!
[V] Line [chown ben:students /home/ben/ccsm] is correct!
[V] Line [passwd -l ben] is correct!
[V] Line [ps -u ben] is correct!
[V] Line [systemctl list-units --type target] is correct!
[V] Line [systemctl set-default multiuser.target] is correct!
[V] Line [/etc/default/useradd] is correct!
[V] Line [HOME=/home/users] is correct!
[V] Line [/etc/sysconfig/network-scripts/ifcfg-enp0s3] is correct!
[V] Line [BOOTPROTO=dhcp] is correct!
[V] Line [/home/mariusz/.bash_profile] is correct!
[V] Line [export SHELL=/bin/bash] is correct!

Group [12] score for task[8]: [100.000000%]
Your current [100.000000%] score is NOT group's best [100.000000%]. Your result will NOT be saved.134
```

*Figure 29*

## References:

Bahri, M. and Ashino, R. (2017) A variation on uncertainty principle and logarithmic uncertainty principle for continuous quaternion wavelet transforms, Abstract and Applied Analysis. Hindawi. Available at: https://www.hindawi.com/journals/aaa/2017/3795120/ (Accessed: March 13, 2023).

Bandel, D. (1997) Disk maintenance under linux (disk recovery).pdfDavid Bandel, docshare.tips. Available at: https://docshare.tips/disk-maintenance-under-linux-disk-recoverypdf_574b9ba1b6d87ff10f8b5074.html (Accessed: March 12, 2023).

Cray, R. (2018). The Book of Overclocking: Tweak Your PC to Unleash Its Power (1st ed.). No Starch Press.

Gregersen, E. (no date) Assembly language, Encyclopædia Britannica. Encyclopædia Britannica, inc. Available at: https://www.britannica.com/technology/assembly- language (Accessed: January 30, 2023).

Gjoen, S. (2002) Multi Disk System Tuning, HOWTO: Multi disk system tuning. Available at: https://tldp.org/HOWTO/Multi-Disk-HOWTO.html (Accessed: March 16, 2023).

Heuring, V. P. and Jordan, H. F. (1997) Computer Systems Design and Architecture - Clemson University, Computer Systems Design and Architecture. Available at: https://people.computing.clemson.edu/~mark/330/chap1.pdf (Accessed: January 21, 2023).

Hummel, H. (2008) Logical Volume Management for Linux on System z, IBM Logical Volume Management for Linux on System z. Available at: http://www.linuxvm.org/present/SHARE110/S9282hh.pdf (Accessed: March 15, 2023).

Hyuk Lee, J. et al. (2012) Pipelined CPU design with FPGA in teaching computer architecture | IEEE ..., Pipelined CPU Design With FPGA in Teaching Computer Architecture. Available at: https://ieeexplore.ieee.org/abstract/document/6093707/ (Accessed: January 19, 2023).

Keedy, M.L. (1966) What is a trapezoid?, NCTM Publications. National Council of Teachers of Mathematics. Available at: https://pubs.nctm.org/view/journals/mt/59/7/article-p646.xml (Accessed: January 30, 2023).

Li, R., Ren, X., Zhao, X., He, S., Stumm, M. and Yuan, D., 2022. ctFS: Replacing file indexing with hardware memory translation through contiguous file allocation for persistent memory. *ACM Transactions on Storage*, *18*(4), pp.1-24.

Patterson, D.A. and Hennessy, J.L., 2013. Computer Organization and Design MIPS Edition: The.

Perugini, S. et al. (2007) Symbolic links in the open directory project, Information Processing & Management. Pergamon. Available at: https://www.sciencedirect.com/science/article/pii/S0306457307001252?casa_token=qJIlqmZCWukAAAAA%3AVkjqLffnaGkc9MXRHcptW9MlsjLzK-u5r96d6Wl_rAF8mZbIYoOUhCwcno_9onM8J7vQtw_O-iI (Accessed: February 18, 2023).

Rodríguez, S., Pedraza, J.L., Dopico, A.G., Rosales, F. and Méndez, R., 2007. Computer-based management environment for an assembly language programming laboratory. *Computer Applications in Engineering Education*, *15*(1), pp.41-54.

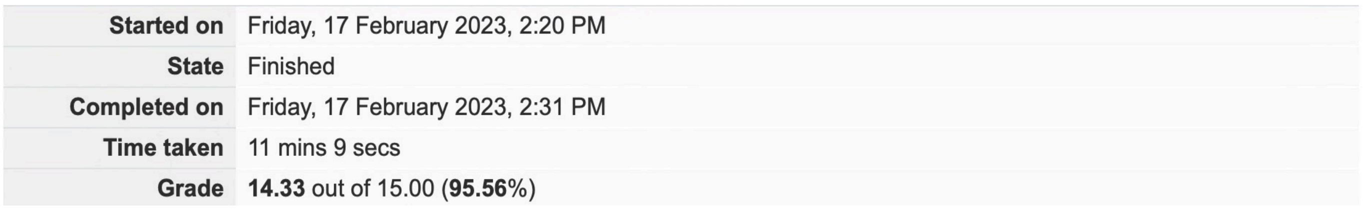
Russinovich, M., Solomon, D. A., & Ionescu, A. (2017). Windows Internals, Part 1: System architecture, processes, threads, memory management, and more (7th Edition). Microsoft Press. (Russinovich et al., 2017)

Tofte , M. and Pierre Talpin, J. (2002) *Region-based memory management*, *Information and Computation*. Academic Press. Available at: https://www.sciencedirect.com/science/article/pii/S0890540196926139 (Accessed: March 5, 2023).

Sivathanu, M., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H. and Jha, S., 2005, December. A Logic of File Systems. In *FAST* (Vol. 5, pp. 1-1).

Zurich, G.W.E.T.H. *et al.* (1991) *Dynamic File Allocation in disk arrays: Proceedings of the 1991 ACM SIGMOD international conference on management of data*, *ACM Conferences*. Available at: https://dl.acm.org/doi/10.1145/115790.115859 (Accessed: March 14, 2023).

QUIZ RESULT:



| | |
|---|---|
| **Started on** | Friday, 17 February 2023, 2:20 PM |
| **State** | Finished |
| **Completed on** | Friday, 17 February 2023, 2:31 PM |
| **Time taken** | 11 mins 9 secs |
| **Grade** | **14.33** out of 15.00 (**95.56**%) |