Student ID: 11987480

Module: 5004CEM Operating Systems and Security

Course: BSc. Computer Science

**Table of Contents**

[**Introduction to the Report** 3](#_Toc95739898)

[**Report Tasks** 3](#_Toc95739899)

[**Task 1 - ACL Approaches** 3](#_Toc95739900)

[**Task 2 – Assembly Code Game** 4](#_Toc95739901)

[**Commented Code for the Game** 4](#_Toc95739902)

[**Multiple Examples of Code Running** 4](#_Toc95739903)

[**Task 4 – TCP Festival Booking System** 5](#_Toc95739904)

[**Commented Code** 5](#_Toc95739905)

[**Multiple Examples of Code Running** 5](#_Toc95739906)

[**Conclusion and Reflection** 6](#_Toc95739907)

[**References** 6](#_Toc95739908)

# **Introduction to the Report**

This report covers three distinct tasks with unique learning outcomes. The first task involves a comparison between micro and macrokernels, examining their differences and benefits. The second task is the creation of a true/false game using assembly language, with a range of functionalities. Finally, the third task involves the development of a TCP client-server, which recommends different walks to the user and also helps them purchase books.

# **Report Tasks**

# **Task 1 - Kernels**

**MICRO AND MACRO KERNELS**

Any computer system's foundation is its operating system (OS), which is responsible for controlling hardware resources, running programs, and providing user interfaces. The kernel is a computer program at the core of a computer's operating system and generally has complete control over everything in the system**.**[**[1]**](#_References) It is considered as a central component that plays a very important role in interacting directly with the hardware and keeping it in usable state .There are two primary categories of OS kernels: microkernels and macrokernels.

A microkernel is a basic approach to kernel design as it keeps only the most crucial elements inside the kernel area. It performs mechanisms like low level addressing , memory management and basic inter-process communication (IPC).it tries to keep other functions like file systems , device drivers and network protocols in user space. Some of notable microkernels are Mach, Symbian OS , Amiga OS.

One of the main advantages of microkernels is their flexibility. Since most OS services are implemented in user space, it can easily be changed or modified while the kernel can remain unchanged . Microkernels are more reliable as it have fewer security vulnerabilities and bugs because of its limited code which is easier to maintain and audit . Every component runs in its own protected address space. In terms of the source code size, microkernels are often smaller than monolithic kernels. The MINIX 3 microkernel, for example, has only approximately 12,000 lines of code.[**[2]**](#_References)

While microkernels have several advantages they also have some disadvantages , it performs most of OS mechanisms in user space which leads to more inter process communication between user space and kernel causing more performance overheads which could cause problems while using high performance applications. They are considered to be less efficient and more complex because of the communication between user space and kernel and the need to ensure every component runs in its own address space which makes it more challenging to achieve high performance .

**MICROKERNAL**

Diagram

Description automatically generated

On the other hand macrokernels are conventional approach to OS kernel design .it contains almost every mechanism of operating system and everything is running in kernel space.it performs wide range of services including system calls , inter process communications , file systems , process scheduling etc. Linux and Windows use microkernels which have loadable modules and similar functionality.

Macrokernels are very efficient as most of components run in kernel space it is easier for user space applications to access all OS services directly without IPC overheads . Also there is no need to swap other programs between kernel and user space . it also have some cons as writing a microkernel is much more difficult because of its larger code and it is less reliable as problem in one part of kernal can cause other parts to stop working properly .

Diagram

Description automatically generated

The main difference between microkernels and macrokernels is their design philosophy and functionality. Microkernels follow a basic approach, keeping near minimum amount of components in the kernel space, while in macrokernels most of the services that are required to run hardware are in kernel space. Microkernels are more reliable and secure because of modular approach and smaller size. In 2018, a paper presented at the Asia-Pacific Systems Conference claimed that microkernels were demonstrably safer than monolithic kernels by investigating all published critical CVEs for the Linux kernel at the time. The study concluded that 40% of the issues could not occur at all in a formally verified microkernel, and only 4% of the issues would remain entirely unmitigated in such a system [[3]](#_References), while macrokernels are less reliable and large but aim to provide more efficient environment for user space applications. On most mainstream processors, obtaining a service is inherently more expensive in a microkernel-based system than a monolithic system[.[4]](#_References)The use of these kernels depends on the application as microkernels are often used for real life systems like automotive , medical systems etc as they are more flexible while macrokernels are suitable for general computing like servers , mobile devices and computers.

In conclusion, both the kernels have their own advantages and disadvantages in terms of design , functionality, and performance. Microkernels are more reliable and secure due to their modular approach and smaller size, while macrokernels aim to provide a more efficient environment for user space applications. The choice of kernel depends on the application.

|  |  |  |
| --- | --- | --- |
|  | **MICROKERNAL** | **MACROKERNAL** |
| **SIZE** | SMALLER CODE AS ONLY ESSENTIAL SERVICES IN KERNAL SPACE | LARGER CODE AS MOST OF MECHANISMS IN KERNEL SPACE |
| **EFFICIENCY** | LESS EFFICIENT BECAUSE OF MORE IPC OVERHEAD | MORE EFFICIENT |
| **SECURITY** | MORE RELIABLE AND SECURE BECAUSE OF SMALLER SIZE AND OWN ADDRESS SPACES | LESS SECURE AS IF ONE PART CRASHES , IT ALSO EFFECTS OTHER PARTS OF SYSTEM |
| **MODIFICATION** | EASIER TO CHANGE AND MODIFY THE USER SPACE WITHOUT CHANGING KERNAL SPACE | DIFFICULT TO MODIFY BECAUSE OF LARGER CODE AND MORE NUMBER OF MECHANUSMS |
| **EXAMPLES** | MACH , SYMBIAN OS | LINUX , WINDOWS |

# **Task 2 – Assembly Code Game**

## **Commented Code for the Game**

section .data

welcome db "Welcome to True False Game",10 ;Welcome message

welcomeLen equ $-welcome ;Calculate the length of the message

trueguide db "You have to type 'T' if your guess is True ",10 ; message for game instructions

trueguideLen equ $-trueguide ;Calculate the length of the message

falseguide db "You have to type 'F' if your guess is False ",10 ; message for game instructions

falseguideLen equ $-falseguide ;Calculate the length of the message

lifeinfo db "Each player has only 3 lifes",10 ; message for life available

lifeinfolen equ $-lifeinfo;Calculate the length of the message

correctmsg db "Player 1 : You guess correctly.", 10 ;Message that player 1 guess is correct

correctmsglen equ $-correctmsg ;Calculate the length of the message

correctmsg2 db "Player 2 : You guess correctly.", 10 ;Message that player guess is correct

correctmsg2len equ $-correctmsg2 ;Calculate the length of the message

wrongmsg db "Player 1 :Your guess is incorrect.", 10 ;Message that player 1 guess is incorrect

wrongmsglen equ $-wrongmsg ;Calculate the length of the message

wrongmsg2 db "Player 2 :Your guess is incorrect.", 10 ;Message that player 2 guess is incorrect

wrongmsg2len equ $-wrongmsg2 ;Calculate the length of the message

bothcmsg db "BOTH OF YOU ARE CORRECT" ; message if both are correct

bothcmsglen equ $-bothcmsg ;Calculate the length of the message

bothwmsg db "BOTH OF YOU ARE WRONG" ; message if both are wrong

bothwmsglen equ $-bothwmsg;Calculate the length of the message

player1scores db "Player 1 Your score is : "; Message to present player 1 score value

player1scoreslen equ $-player1scores;Calculate the length of the message

player2scores db "Player 2 Your score is : "; Message to present player 2 score value

player2scoreslen equ $-player2scores ;Calculate the length of the message

p1life db "Player 1 Life left : " ;message to show player 1 life

p1lifelen equ $-p1life;Calculate the length of the message

p2life db " Player 2 life left :";message to show player 2 life

p2lifelen equ $-p2life;Calculate the length of the message

player1msg db "Player 1 : " ;message for player 1 input

player1msglen equ $-player1msg ;Calculate the length of the message

player2msg db "Player 2 : ";message for player 2 input

player2msglen equ $-player2msg ;Calculate the length of the message

p1finish db "PLAYER 1 YOU HAVE NO LIFES LEFT ,PLAYER 2 YOU WON", ;player 1 win message

p1finishlen equ $-p1finish ;Calculate the length of the message

p2finish db "PLAYER 2 YOU HAVE NO LIFES LEFT ,PLAYER 1 YOU WON",10 ; player 2 win message

p2finishlen equ $-p2finish ;Calculate the length of the message

bothlostmsg db "BOTH OF YOU HAVE ZERO LIFES LEFT , GAME DRAW ",10 ; draw message

bothlostmsglen equ $-bothlostmsg ;Calculate the length of the message

continuemsg db "DO YOU WANT TO CONTINUE ? ",10; message to ask if player want to continue

continuemsglen equ $-continuemsg ;Calculate the length of the message

continueoption db "TYPE 'Y' TO CONTINUE PLAYING ELSE TYPE 'N' ",10 ; message to ask for continue

continueoptionlen equ $-continueoption ;Calculate the length of the message

uppercasemessage db "YOU HAVE TO GIVE INPUT IN CAPITAL (UPPERCASE) AND ONLY ONE LETTER " ,10

uppercasemessagelen equ $-uppercasemessage

fivecorrectp1msg db "CONGRATULATIONS PLAYER 1 YOU ANSWERED 5 QUESTIONS CORRECT IN A ROW , YOU WON A PRIZE"

fivecorrectp1msglen equ $- fivecorrectp1msg

fivecorrectp2msg db "CONGRATULATIONS PLAYER 2 YOU ANSWERED 5 QUESTIONS CORRECT IN A ROW , YOU WON A PRIZE "

fivecorrectp2msglen equ $- fivecorrectp2msg

outofmsg db " out of "

outofmsglen equ $-outofmsg

score1value db 0 ; score for player 1 initialized at 0

score2value db 0 ; score for player 2 initialized at 0

player1life db 3 ; life for player 1 initialized at 3

player2life db 3 ; life for player 2 initialized at 3

player1finish db 0 ; end life for player 1 initialized at 0

player2finish db 0 ; end life for player 2 initialized at 0

questioncountervalue db 0 ; counting question initialized at 0

questioncountervalue1 db 0 ; counting question initialized at 0

player1correctcount db 0

player2correctcount db 0

cr db 10 ; for creating a new line

; QUESTIONS

question1 db "The human brain is the most powerful computer in the world."

question1len equ $-question1

question2 db "The most popular operating system in the world is Linux."

question2len equ $-question2

question3 db "The longest river in the world is the Nile River."

question3len equ $-question3

question4 db "The King of the United Kingdom is also the head of state of 15 other countries."

question4len equ $-question4

question5 db "The first human to walk on the moon was Neil Armstrong, who landed on the moon on July 20, 1969."

question5len equ $-question5

question6 db "The average person has 30 senses."

question6len equ $-question6

question7 db "The sky is blue because of the way sunlight interacts with the atmosphere."

question7len equ $-question7

question8 db "The capital of the United States is New York City."

question8len equ $-question8

question9 db "The operating system provides a user interface, which allows users to interact with the computer. "

question9len equ $-question9

question10 db "The capital of Canada is Toronto. "

question10len equ $-question10

question11 db "The operating system is the most important program on a computer."

question11len equ $-question11

question12 db "The Great Wall of China is visible from space."

question12len equ $-question12

question13 db "The Eiffel Tower is the tallest building in the world."

question13len equ $-question13

question14 db "The human heart beats about 100,000 times per day."

question14len equ $-question14

question15 db "The operating system is responsible for managing memory."

question15len equ $-question15

global listquestions

listquestions:

dq question1

dq question2

dq question3

dq question4

dq question5

dq question6

dq question7

dq question8

dq question9

dq question10

dq question11

dq question12

dq question13

dq question14

dq question15;the pointer to the quotes are stored in 8 bytes

listquestionslen:

dq question1len

dq question2len

dq question3len

dq question4len

dq question5len

dq question6len

dq question7len

dq question8len

dq question9len

dq question10len

dq question11len

dq question12len

dq question13len

dq question14len

dq question15len; the pointer to the quote lengths are stored in 8 bytes

;ARRAY FOR ANSWERS

global listanswers

listanswers:

dq 'T'

dq 'F'

dq 'T'

dq 'T'

dq 'T'

dq 'F'

dq 'T'

dq 'F'

dq 'T'

dq 'F'; the pointer to the quote lengths are stored in 8 bytes

dq 'T'

dq 'F'

dq 'F'

dq 'T'

dq 'T'

letter:

dq 0 ; where we store each of the answers one at a time

section .bss

guess1 resq 1 ; variable to store the player 1 guess

guess2 resq 1 ; variable to store the player 2 guess

clean resq 1 ; variable to store keyboard buffer

score1 resb 1 ; variable to store player 1 score index

score2 resb 1 ; variable to store player 2 score index

life1 resb 1 ; variable to store player 1 life index

life2 resb 1 ; variable to store player 2 life index

askcontinue resb 1 ; variable to store keyboard buffer after asking to continue

questioncounter resb 1 ; variable to store no of question

questioncounter1 resb 1 ; variable to store no of question

p1correct resb 1

p2correct resb 1

section .text

global \_start ;must be declared for linker (ld)

\_start:

call displayWelcome ;call welcome message

call newLine

call displayguidefortrue ; call intro

call displayguideforfalse ; call intro

call newLine

call lifeguide ; function to tell about lifes

call newLine

call capitalinfo ; uppercase input only guide

call newLine

mov rax,15 ;number of questions and so possible answers

mov rbx,0 ;RBX will store the answers currently being guessed

mov rcx,listanswers

mov r8, listquestions ; point to first quote

Mov r9, listquestionslen ; point to the length of first quote

mov byte [player1life],3

mov byte [player2life],3

;Main function that calls other functions

mainloop:

mov rbx, [rcx] ; put the current letter being guessed in rbx

mov [letter],rbx ; move rbx into a variable letter that stores the current correct answer

push rax ; push rax on stack

push rcx ; push rax on stack

call newLine

call displayquestions ; call subroutine to display the question

call newLine

call player1input ; call subroutine to get the users guess and compare the guess with t

call player2input; new line like endl in C++

call bothfunction ; call to check if both player answers are same

call newLine

call player1index ; call player 1 score index

call questioncounterfunction;

call newLine

call player2index ; call player 2 score index

call questioncounterfunction1

call newLine

call newLine

call player1lifeindex ; call player 1 life index

call player2lifeindex ; call player 2 life index

call newLine

call continue ; call function to ask player if they want to continue

pop rcx ; get back from stack

pop rax ; get back from stack

add r8,8 ;move pointer to next questions. As 8 bits for each questiobs move on by 8

add r9,8

add rcx,8

dec rax

cmp rax,0 ;compare if rax equals 0

jne mainloop ;if not 0, then continue looping

mov rax, 15 ;reset the counter to 15 for questions

mov rcx, listanswers ;reset the answer pointer to the beginning

mov r8, listquestions ;reset the question pointer to the beginning

mov r9, listquestionslen ;reset the question length pointer to the beginning ;restart the loop from the beginning ;decrement counter by one so going down

jmp mainloop ;if counter not 0, then loop again

call done ; call subroutine to end program

;function to read the player 1 guess and do comparison with the answer

player1input:

call playermessage ; player 1 input message

mov eax, 3 ; read from keyboard

mov ebx, 2; stdin

mov ecx, guess1 ; move player 1 guess into ecx

mov edx, 1 ; As single letter using 1 byte

int 80h ; invoke the kernel to get the player 1 guess

mov eax, 3 ; read to clear the keyboard buffer

mov ebx, 2 ; stdin

mov ecx, clean ; Clear the key press from the user input so it does not messy up loop

mov edx, 1 ; As single character using 1 byte

int 80h

ret ; return to the main section

player2input: ; function to get player 2 input

call player2message

mov eax, 3 ; read from keyboard

mov ebx, 2; stdin

mov ecx, guess2 ; move user guess into ecx

mov edx, 1 ; As single letter using 1 byte

int 80h ; invoke the kernel to get the user's guess

mov eax, 3 ; read to clear the keyboard buffer

mov ebx, 2 ; stdin

mov ecx, clean ; Clear the key press from the user input so it does not messy up loop

mov edx, 1 ; As single character using 1 byte

int 80h

; if the guess is incorrect then go to Notsame function

call newLine

ret ; return to the main section

;Function to compare player 1 answer

compareanswers:

mov rax, [guess1] ; move player 1 guess by user into rax

cmp rax, [letter] ; compare correct answer with what in rax

je same ; if guess was correct jump to same function

call Notsame ; if the guess is incorrect then go to Notsame function

jmp compareanswers2

ret

;Function to compare player 2 answers

compareanswers2:

mov rdx, [guess2] ; move guess2 by player2 into rax

cmp rdx, [letter] ; compare correct answer with what in rax

je same2 ; if guess2 was correct jump to same function

call Notsame2

ret

;Function to check if users have same guess

bothfunction:

mov rax,[guess1]

cmp rax,[guess2] ; compare player 1 and player 2 guess

je comparetf ; if same then call comparetf function

call compareanswers

ret

;Function if users guess are same to check if true or false

comparetf:

mov rax, [guess1] ; move guess by player1 into rax

cmp rax, [letter] ; compare correct answer with what in rax

je bothcorrect ; if guess was correct jump to bothcorrect function

call bothwrong ; if the guess is incorrect then go to bothwrong function

ret

;Function to print message if both correct

bothcorrect:

mov ecx,bothcmsg ; both correct message

mov edx, bothcmsglen ; length of both correct message

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 80h ; invoke the kernel to display message

; increment both player score

inc byte [score1value] ; increment the value in the variable

mov eax, [score1value] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [score1], eax

;increment player 1 correct count to see how many answers they give correctly in row

inc byte [player1correctcount] ; increment the value in the variable

mov eax, [player1correctcount] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [p1correct], eax

inc byte [score2value] ; increment the value in the variable

mov eax, [score2value] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [score2], eax

;increment player 2 correct count to see how many answers they give correctly in row

inc byte [player2correctcount] ; increment the value in the variable

mov eax, [player2correctcount] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [p2correct], eax

call newLine

call checkprizefunction ; function to check if 5 correct answers in row

ret ; return to main function

bothwrong:

mov ecx,bothwmsg ; Not same message

mov edx, bothwmsglen ; length of same message

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 80h ; invoke the kernel to display message

; decrement each player life if both wrong

dec byte [player1life] ; decrement the value in the variable

mov eax, [player1life] ; move the decValue variable in eax

add eax, 48

mov [life1] , eax

dec byte [player2life] ; decrement the value in the variable

mov eax, [player2life] ; move the decValue variable in eax

add eax, 48

mov [life2], eax

;making both player correct count in a row to 0 if wrong answer given by multiplying by 0

mov eax,'0' ; put ASCII value 0 into eax register

sub eax, '0' ; subtract '0' from eax to create an integer

mov ecx,[player1correctcount]

sub ecx, '0' ; subtract '0' from eax to create an integer

mul ecx; mutliply the integers in eax and ecx and put the outcome in eax

add eax, '0' ; convert the result in eax to an ASCII representation of the integer

mov [p1correct], eax ; move the ASCII representation of the integer into a variable p1correct

mov eax,'0' ; put ASCII value 0 into eax register

sub eax, '0' ; subtract '0' from eax to create an integer

mov ecx,[player2correctcount]

sub ecx, '0' ; subtract '0' from eax to create an integer

mul ecx; mutliply the integers in eax and ecx and put the outcome in eax

add eax, '0' ; convert the result in eax to an ASCII representation of the integer

mov [p2correct], eax ; move the ASCII representation of the integer into p2correct

ret

; function to show message that answer was not correct answer

Notsame:

mov ecx,wrongmsg ; Not same message

mov edx, wrongmsglen ; length of same message

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 80h ; invoke the kernel to display message

dec byte [player1life] ; decrement the value in the variable

mov eax, [player1life] ; move the decValue variable in eax

add eax, 48

mov [life1] , eax

;making player 1 correct in a row to 0

mov eax,'0' ; put ASCII value 0 into eax register

sub eax, '0' ; subtract '0' from eax to create an integer

mov ecx,[player1correctcount]

sub ecx, '0' ; subtract '0' from eax to create an integer

mul ecx; mutliply the integers in eax and ecx and put the outcome in eax

add eax, '0' ; convert the result in eax to an ASCII representation of the integer

mov [p1correct], eax ; move the ASCII representation of the integer into a variable p1correct

ret ; return to main code

; function to show message answer was correct

same:

mov ecx,correctmsg ; same message

mov edx, correctmsglen ; length of same message

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 80h ; invoke the kernel to display message

inc byte [score1value] ; increment the value in the variable

mov eax, [score1value] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [score1], eax

;increasing player 1 correct in a row

inc byte [player1correctcount] ; increment the value in the variable

mov eax, [player1correctcount] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [p1correct], eax

call newLine

call checkprizefunction

call compareanswers2

ret ; return to main code

; function to show message answer was correct

same2:

mov ecx,correctmsg2 ; same message

mov edx, correctmsg2len ; length of same message

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 80h ; invoke the kernel to display message

inc byte [score2value] ; increment the value in the variable

mov eax, [score2value] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [score2], eax

inc byte [player2correctcount] ; increment the value in the variable

mov eax, [player2correctcount] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [p2correct], eax

call newLine

call checkprizefunction

ret ; return to main code

Notsame2:

mov ecx,wrongmsg2 ; same message

mov edx, wrongmsg2len ; length of same message

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 80h ; invoke the kernel to display message

dec byte [player2life] ; decrement the value in the variable

mov eax, [player2life] ; move the decValue variable in eax

add eax, 48

mov [life2], eax

;making player 2 correct count in a row to 0

mov eax,'0' ; put ASCII value 0 into eax register

sub eax, '0' ; subtract '0' from eax to create an integer

mov ecx,[player2correctcount]

sub ecx, '0' ; subtract '0' from eax to create an integer

mul ecx; mutliply the integers in eax and ecx and put the outcome in eax

add eax, '0' ; convert the result in eax to an ASCII representation of the integer

mov [p2correct], eax ; move the ASCII representation of the integer into a variable p2correct

ret ; return to main code

;Function for player 1 score index

player1index:

mov edx,player1scoreslen

mov ecx,player1scores

mov ebx,1

mov eax,4

int 80h

mov eax, [score1value] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [score1], eax

mov eax, 4 ; system call the screen

mov ebx, 1 ; standard out

mov ecx, score1; print value

mov edx, 1; One byte is size

int 80h ; interrupt

mov edx , outofmsglen

mov ecx,outofmsg

mov ebx,1

mov eax,4

int 80h

ret

;Function for player 2 score index

player2index:

mov edx,player2scoreslen

mov ecx,player2scores

mov ebx,1

mov eax,4

int 80h

mov eax, [score2value] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [score2], eax

mov eax, 4 ; system call the screen

mov ebx, 1 ; standard out

mov ecx, score2; print value

mov edx, 1; One byte is size

int 80h ; interrupt

mov edx , outofmsglen

mov ecx,outofmsg

mov ebx,1

mov eax,4

int 80h

ret

checkprizefunction:

cmp byte [p1correct] , '5'

je prize1

cmp byte [p2correct] , '5'

je prize2

ret

;function for player 1 life

player1lifeindex:

cmp byte [player1life], 0 ; check if player 1's life has become 0

jz player1done

cmp byte [player2life], 0 ; check if player 1's life has become 0

jz player2done

mov eax, [player1life] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [life1], eax

mov edx,p1lifelen

mov ecx,p1life

mov ebx,1

mov eax,4

int 80h

mov eax,4

mov ebx,1

mov ecx , life1

mov edx,1

int 80h

call newLine

ret

;function for player 2 life

player2lifeindex:

cmp byte [player2life], 0 ; check if player 1's life has become 0

jz player2done ;

cmp byte [player1life], 0 ; check if player 1's life has become 0

jz player1done ;

mov eax, [player2life] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [life2], eax

mov edx,p2lifelen

mov ecx,p2life

mov ebx,1

mov eax,4

int 80h

mov eax,4

mov ebx,1

mov ecx , life2

mov edx,1

int 80h

call newLine

ret

;function to display questions

displayquestions:

mov edx, [r9] ;message length content of register r9

mov ecx, [r8] ;message to write content of register r8

mov ebx, 1 ;file descriptor (stdout)

mov eax, 4 ;system call number (sys\_write)

int 0x80 ;invoke the kernel to display the quote

ret ; return to previous position in code

;Function to display welcome message

displayWelcome:

mov edx,welcomeLen ;message length

mov ecx, welcome ;message to write

mov ebx, 1 ;file descriptor (stdout)

mov eax, 4 ;system call number (sys\_write)

int 0x80 ;invoke the kernel to print the message

ret ; return to the main section

;Function to display guide for true

displayguidefortrue:

mov edx,trueguideLen ;message length

mov ecx, trueguide ;message to write

mov ebx, 1 ;file descriptor (stdout)

mov eax, 4 ;system call number (sys\_write)

int 0x80 ;invoke the kernel to print the message

ret ; return to the main section

;Function to display guide for false

displayguideforfalse:

mov edx,falseguideLen ;message length

mov ecx, falseguide ;message to write

mov ebx, 1 ;file descriptor (stdout)

mov eax, 4 ;system call number (sys\_write)

int 0x80 ;invoke the kernel to print the message

ret ; return to the main section

; Function to create a New line like endl in C++

newLine:

mov eax,4 ; Put 4 in eax register into which is system

;call for write (sys\_write)

mov ebx,1 ; Put 1 in ebx register which is the standard

; output to the screen

mov ecx, cr ; Put the newline value into ecx register

mov edx, 1 ; Put the length of the newline value into edx

; register

int 80h ; Call the kernel with interrupt to check the

; registers and perform the action of moving to

; the next line like endl in c++

ret ; return to previous position in code

;player 1 input message function

playermessage:

mov edx , player1msglen

mov ecx , player1msg

mov ebx,1

mov eax,4

int 0x80

ret

;player 2 input message function

player2message:

mov edx,player2msglen

mov ecx ,player2msg

mov ebx,1

mov eax , 4

int 0x80

ret

;PLAYER 2 WIN MESSAGE

player1done:

cmp byte [player2life],0

je gameover

mov edx,p1finishlen

mov ecx,p1finish

mov ebx,1

mov eax,4

int 80h

call newLine

call done

;PLAYER 1 WIN MESSAGE

player2done:

cmp byte [player1life],0

je gameover

mov edx,p2finishlen

mov ecx,p2finish

mov ebx,1

mov eax,4

int 80h

call newLine

call done

;GAME OVER DRAW FUNCTION

gameover:

mov ecx,bothlostmsg ; draw message

mov edx, bothlostmsglen ; length of draw messagr

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 80h ; invoke the kernel to display message

call newLine

call done ; end program

;FUNCTION FOR GETTING INPUT AND ASKING FOR CONTINUING GAME OR NOT

continue:

mov edx,continuemsglen

mov ecx,continuemsg

mov ebx,1

mov eax,4

int 80h

call newLine

mov edx,continueoptionlen

mov ecx ,continueoption

mov ebx,1

mov eax , 4

Int 80h

call newLine

mov eax, 3 ; read from keyboard

mov ebx, 2; stdin

mov ecx, askcontinue ; move user input into ecx

mov edx, 1 ; As single letter using 1 byte

int 80h ; invoke the kernel to get the user's input

cmp byte [askcontinue], 'N' ; compare input with 'N'

je done ; if input is equal to N jump to done function

mov eax, 3 ; read to clear the keyboard buffer

mov ebx, 2 ; stdin

mov ecx, askcontinue ; Clear the key press from the user input so it does not messy up loop

mov edx, 1 ; As single character using 1 byte

int 80h ; invoke the kernel to take the enter key press to clear the keyboard buffer ; return to main code

ret ; return to the main section

;INTRO FUNCTION FOR TELLING ABOUT LIFES AVAILABLE

lifeguide:

mov edx,lifeinfolen

mov ecx,lifeinfo

mov ebx,1

mov eax,4

int 80h

ret

;function to give input in uppercase only

capitalinfo:

mov edx , uppercasemessagelen

mov ecx , uppercasemessage

mov ebx,1

mov eax,4

int 80h

ret

;function to count questions that are given to player

questioncounterfunction:

inc byte [questioncountervalue] ; increment the value in the variable

mov eax, [questioncountervalue] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [questioncounter], eax

mov eax, 4 ; system call the screen

mov ebx, 1 ; standard out

mov ecx, questioncounter; print value

mov edx, 1; One byte is size

int 80h ; interrupt

ret

questioncounterfunction1:

inc byte [questioncountervalue1] ; increment the value in the variable

mov eax, [questioncountervalue1] ;move the incValue variable in eax

add eax, 48 ; convert the integer into ascii value to print

mov [questioncounter1], eax

mov eax, 4 ; system call the screen

mov ebx, 1 ; standard out

mov ecx, questioncounter1; print value

mov edx, 1; One byte is size

int 80h ; interrupt

ret

;function for five correct answers in row

prize1:

mov edx,fivecorrectp1msglen

mov ecx,fivecorrectp1msg

mov ebx,1

mov eax,4

int 80h

call newLine

ret

prize2:

mov edx,fivecorrectp2msglen

mov ecx,fivecorrectp2msg

mov ebx,1

mov eax,4

int 80h

call newLine

ret

;FUNCTION FOR ENDING PROGRAM

done:

mov eax, 1 ;system call number (sys\_exit)

int 0x80 ;invoke the kernel to end the program

## **Multiple Examples of Code Running**

1. **INTRODUCTION OF GAME AND WHEN BOTH PLAYER ENTER CORRECT ANSWER**

Text

Description automatically generated

1. **WHEN PLAYER 1 GUESS IS CORRECT BUT PLAYER 2 IS INCORRECT**

**Text

Description automatically generated**

1. **WHEN PLAYER 1 IS INCORRECT WHILE PLAYER 2 IS CORRECT**

Text

Description automatically generated

1. **WHEN BOTH PLAYER GUESS IS WRONG**

Text

Description automatically generated

1. **DRAW AND GAME OVER WHEN BOTH PLAYERS LIVES FINISH**

**Text

Description automatically generated**

1. **WHEN PLAYER 2 WIN AS PLAYER 1 LIVES FINISH**

**Text

Description automatically generated**

1. **IF USER TYPE N TO EXIT THE GAME**

**Text

Description automatically generated with medium confidence**

1. **WHEN PLAYER 1 GIVE 5 CORRECT ANSWERS IN A ROW**

**Text

Description automatically generated**

1. **WHEN PLAYER 2 GIVES FIVE CORRECT ANSWERS IN A ROW**

**Text

Description automatically generated**

1. **WHEN PLAYER 2 WIN AS PLAYER 1 LIVES FINISH**

**Text

Description automatically generated**

# **Task 4 – TCP Walks Recommendation and Book Sales System**

## **Commented Code**

**SERVER CODE**

import socket

import sys

*# A list of dictionaries to represent walks in different areas*

walks = [

    {

        'area': 'PeakDistrict',

        'book': 'More Peak District',

        'name': 'Hathersage',

        'distance': '7 miles',

        'difficulty': 'Easy',

        'page': '67'

    },

    {

        'area': 'PeakDistrict',

        'book': 'More Peak District',

        'name': 'Hope and Win Hill',

        'distance': '4.5 miles',

        'difficulty': 'Medium',

        'page': '18'

    },

    {

        'area': 'Lincolnshire',

        'book': 'Lincolnshire Wolds',

        'name': 'Thornton Abbey',

        'distance': '3.5 miles',

        'difficulty': 'Easy',

        'page': '20'

    },

    {

        'area': 'Lincolnshire',

        'book': 'Lincolnshire Wolds',

        'name': 'Tennyson County',

        'distance': '5 miles',

        'difficulty': 'Hard',

        'page': '28'

    },

    {

        'area': 'York',

        'book': 'Vale of York',

        'name': 'Cowlam and Cotham',

        'distance': '8 miles',

        'difficulty': 'Hard',

        'page': '64'

    },

    {

        'area': 'York',

        'book': 'Vale of York',

        'name': 'Fridaythorpe',

        'distance': '7 miles',

        'difficulty': 'Easy',

        'page': '42'

    },

    {

        'area': 'PeakDistrict',

        'book': 'Peak District',

        'name': 'Magpie Mine',

        'distance': '4.5 miles',

        'difficulty': 'Medium',

        'page': '20'

    },

    {

        'area': 'PeakDistrict',

        'book': 'Peak District',

        'name': 'Lord’s Seat',

        'distance': '5.5 miles',

        'difficulty': 'Easy',

        'page': '28'

    },

    {

        'area': 'NorthWales',

        'book': 'Snowdonia',

        'name': 'Around Aber',

        'distance': '4 miles',

        'difficulty': 'Hard',

        'page': '24'

    },

    {

        'area': 'NorthWales',

        'book': 'Snowdonia',

        'name': 'Yr Eifl',

        'distance': '3.5 miles',

        'difficulty': 'Medium',

        'page': '42'

    },

    {

        'area': 'Warwickshire',

        'book': 'Malvern and Warwickshire',

        'name': 'Edge Hill',

        'distance': '4 miles',

        'difficulty': 'Easy',

        'page': '28'

    },

    {

        'area': 'Warwickshire',

        'book': 'Malvern and Warwickshire',

        'name': 'Bidford-Upon-Avon',

        'distance': '8.5 miles',

        'difficulty': 'Medium',

        'page': '78'

    },

    {

        'area': 'Cheshire',

        'book': 'Cheshire',

        'name': 'Dane Valley',

        'distance': '5 miles',

        'difficulty': 'Easy',

        'page': '20'

    },

    {

        'area': 'Cheshire',

        'book': 'Cheshire',

        'name': 'Malpas',

        'distance': '8.5 miles',

        'difficulty': 'Medium',

        'page': '80'

    },

    {

        'area': 'Cheshire',

        'book': 'Cheshire',

        'name': 'Farndon',

        'distance': '6 miles',

        'difficulty': 'Hard',

        'page': '48'

    },

    {

        'area': 'Cheshire',

        'book': 'Cheshire',

        'name': 'Delamere Forest',

        'distance': '5.5 miles',

        'difficulty': 'Easy',

        'page': '30'

    }

]

*# A list of dictionaries to represent books*

books = [

    {'book\_name':"More Peak District" ,

    "book\_num": 101,

    "price": 12.99

    },

    {'book\_name':"Lincolnshire Wolds" ,

    'book\_num': 102 ,

    'price': 10.99

    },

    {'book\_name':"Vale of York",

    'book\_num': 103,

    'price': 11.99

    },

    {'book\_name':"Peak District",

    'book\_num': 104,

    'price': 12.99

    },

    {'book\_name':"Snowdonia",

    'book\_num': 105,

    'price': 13.99

    },

    {'book\_name':"Malvern and Warwickshire",

     'book\_num': 106,

     'price': 10.99

    },

    {

    'book\_name':"Cheshire",

    'book\_num': 107,

    'price': 12.99

    }

]

*#Function to search for walks using input from client*

def search(area, min\_len, max\_len, diff):

    result = ''

*# Loop through each walk in the list of walks*

    for walk in walks:

*# Check if the client input matches any walk in list*

        if walk['area'] == area and min\_len <= float(walk['distance'].split()[0]) <= max\_len and walk['difficulty'].lower() == diff.lower():

            result += walk['book'] + ', ' + walk['name'] + ', pg ' + walk['page'] + '\n'

    if result:

        return result

    else: *#IF NO WALK IS FOUND*

        return f"No walks found in {area} with length between {min\_len} and {max\_len} miles and difficulty {diff}."

*#Function to search for book in list of books*

def buy(book\_num, quantity):

    found\_book = None *#book to be searched is initialized at None*

    for book in books:

        if book['book\_num'] == int(book\_num): *#comparing client input with search input*

            found\_book = book

            break

    if not found\_book: *#if book does not exist*

        return f"Invalid book number: {book\_num}"

    elif int(quantity) > 10: *#to avoid selling more quantities*

        return "Maximum quantity per purchase is 10."

    else:

        price = found\_book['price'] \* int(quantity)

        if price > 75 : *# applying discount of 10% is price is more than 75*

            orignalprice = price

            price = price - (price \* 10/100)

            return f"Thank you for your purchase of {quantity} copies of {found\_book['book\_name']}. Total price: {price:.2f} \n You get 10% discount as yur bill amount is more than 75 Your previous amount was {orignalprice:.2f}"

        return f"Thank you for your purchase of {quantity} copies of {found\_book['book\_name']}. Total price: {price:.2f} "

host = 'localhost'

data\_payload = 2048

backlog = 5

port = 1234

*# Create a TCP socket*

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

*# Bind the socket to the port*

server\_address = (host, port)

print("Starting up server on %s port %s" % server\_address)

server.bind(server\_address)

*# Listen to clients, backlog argument specifies the max no. of queued connections*

server.listen(backlog)

client, address = server.accept()

print("Connection from:", address)

while True:

    data = client.recv(data\_payload)

    if not data:

        break

    received\_message = data.decode() *#decoding message from client*

    print(f"Received message: {received\_message}")

    if received\_message.startswith("search"): *# to check if user want to search*

        \_, area, min\_len, max\_len, diff = received\_message.split() *# splitting user input and storing them as different variable*

        response = search(area, float(min\_len), float(max\_len), diff)

        client.sendall(response.encode())

    elif received\_message.startswith("buy"): *# to check if user want to buy*

        \_, book\_num, quantity = received\_message.split() *# splittting user input to search for book*

        response = buy(book\_num, quantity)

        client.sendall(response.encode())

    else: *#if user input something else instead of seaarch or buy*

        client.sendall(b"Invalid command. Please enter 'search' or 'buy'.")

**CLIENT CODE**

import socket

host = 'localhost'

port = 1234

*# Create a TCP/IP socket*

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

*# Connect the socket to the server*

server\_address = (host, port)

print("Connecting to %s port %s" % server\_address)

client.connect(server\_address)

def search(area, min\_len, max\_len, diff): *# function to send search query to server*

    search\_query = f"search {area} {min\_len} {max\_len} {diff}"

    client.sendall(search\_query.encode())

    data = client.recv(1024)

    print(data.decode())

def buy(book\_num, quantity): *# function to send buy query to server*

    buy\_query = f"buy {book\_num} {quantity}"

    client.sendall(buy\_query.encode())

    data = client.recv(1024)

    print(data.decode())

customernames = [] *#list to store previous customer names*

while True:

    action = input("Do you want to search or buy? ")

    if action == 'search': *# if user input is search get all input regarding search*

        area = input("Area Where Want to Walk: ")

        min\_len = input("Minimum Length in Miles: ")

        max\_len = input("Maximum Length in Miles: ")

        diff = input("Level of Difficulty: ")

        search(area, min\_len, max\_len, diff) *# send all input to function search for encoding message to send to server*

    elif action == 'buy': *#if user input is buy then get input to search for book*

        customername = input("Please enter your name.\n")

        book\_num = input("Enter the book number you want to buy: ")

        quantity = input("Enter the quantity: ")

        bookquantity = int(quantity)

        if customername in customernames and bookquantity > 5 : *# to reduce reselling of books and checking if customer has placed order before*

            print("You have placed a order with us before , please wait some time or take less quantity ") *#to stop previous customer to purchase more than 5 books again*

        else:

             customernames.append(customername)

             buy(book\_num, quantity)

    else: *# if user type something else insteaf of search or buy*

        print("Invalid action. Please enter 'search' or 'buy'.")

## **Multiple Examples of Code Running**

1. **When client put buy query**

*Text

Description automatically generated*

1. **Message on server when client put buy query**

Text

Description automatically generated

s

1. **Example of search query**

**Text

Description automatically generated**

1. **Search query on server**

**Graphical user interface, text

Description automatically generated**

1. **When client input something else instead search or buy**

****

1. **If there is no particular walk matching search query**

**Text

Description automatically generated**

1. **When no book exist**

**Text

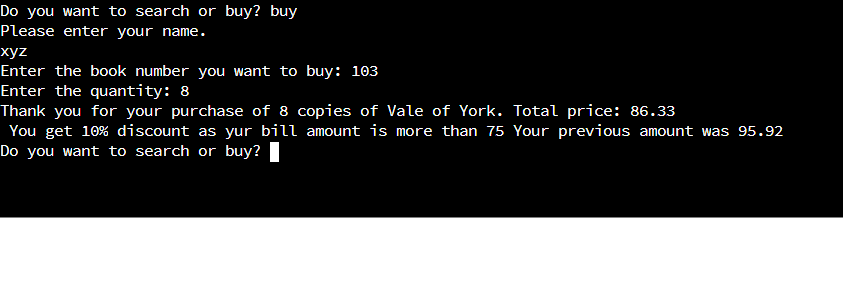
Description automatically generated**

1. **When previous customer tries to buy more than 5 books**

**Text

Description automatically generated**

1. **When 10% discount is applied automatically**

****

# **Conclusion and Reflection**

Through these tasks, I gained a deeper understanding of different concepts and technologies, such as kernel architecture, assembly language programming, and network communication protocols. Each task required a different set of skills and knowledge, and allowed for practical application of theoretical concepts. I explored different aspects of assembly code and also learned to make a proper tcp server . I learned about kernel architecture and how operating systems works.

# **References**

[[1](#_top)] linfo , Bellevue Linux Users Group. Archived from the original on 8 December 2006. Retrieved 15 September 2016 <https://web.archive.org/web/20061208185439/http://www.linfo.org/kernel.html>

[2]<https://wiki.minix3.org/doku.php?id=www:documentation:read-more> Retrieved 20 December 2016

[3]Liedtke, Jochen (September 1996). "Towards Real Microkernels". *Communications of the ACM*. **39** (9): 70–77. <https://dl.acm.org/doi/10.1145/234215.234473>

*[4]Proceedings of the 9th Asia-Pacific Workshop on Systems*. Jeju Island, Republic of Korea: Association for Computing Machinery. pp. 1–7. https://dl.acm.org/doi/10.1145/3265723.3265733