***Calculate the Area & Perimeter of the Circle, Square, and Rectangle***

****

***Computer Organization & Assembly Language Lab***

**Group Members**

***Ateeq Ur Rehman-221475***

***Salman Abdullah-221478***

***Submitted To: Ma’am Romana Maroof***

***Department of Computer Science***

# Description and Functionalities:

The given code is an assembly language program that allows the user to perform various calculations related to geometric shapes such as circles, squares, and rectangles. The program uses the Irvine32 library for input/output operations.

Here's a description of the code and its functionalities:

## Data Section:

* Various strings are declared to display messages and prompts to the user.
* Variables are declared to store user inputs, choices, and calculated results.
* Constants such as PI, TWO, and FOUR are defined.

## Display\_Name\_Function:

* This procedure displays the name of the department, subject, project title, teacher's name, and student's name.
* The information is displayed using the writestring function.
* After displaying the information, the screen is cleared.

## Choice\_Function:

* This procedure asks the user if they want to go to the main menu or exit the program.
* The user's choice is stored in the 'cho' variable.
* If the choice is 'y', the program jumps to the switch statement. Otherwise, it exits the program.

## main Procedure:

* The Display\_Name\_Function is called to display the information.
* The switch statement messages are displayed.
* The user is prompted to enter their choice, and the input is stored in the 'choice' variable.
* The switch statement compares the user's choice to different cases and performs corresponding calculations.
* Each case prompts the user for necessary inputs, performs the calculation, and displays the result.
* After each calculation, the Choice\_Function is called to ask the user if they want to continue or exit.
* If the user chooses to continue, the switch statement is displayed again.
* If the user chooses to exit, the program terminates.

Overall, the code provides a menu-driven interface for the user to calculate the area and perimeter of circles, squares, and rectangles. It includes input validation to handle invalid user inputs. The program loops until the user chooses to exit, allowing them to perform multiple calculations in a single run.

# Explanation of the algorithms and techniques used in the program:

The given program is written in Assembly language and it allows the user to calculate the area and perimeter of various shapes such as a circle, square, and rectangle. Let's go through the algorithms and techniques used in the program:

## Data Section:

* The program begins by declaring various data variables using the .data section. These variables store messages, input values, output results, and constants required for calculations.

## Display\_Name\_Function:

* This function is responsible for displaying the project information and details, including the department name, subject name, project title, teacher's name, and student's name.

## Choice\_Function:

* This function prompts the user to enter their choice to continue or exit the program. It reads a single character input and stores it in the cho variable.
* If the user enters 'y', it proceeds to the switch statement for shape selection. Otherwise, it exits the program.

## main PROC:

* This is the main procedure where the program execution starts.
* It calls the Display\_Name\_Function to display project information.
* Then it displays a menu of options using the switch statement, prompting the user to select a shape or exit.
* Based on the user's input, it jumps to the respective case to perform calculations for the selected shape.

## Switch Statements:

* The program uses a switch statement to handle user choices for different shapes.
* Each case represents a shape option (circle, square, rectangle) and performs specific calculations based on the user's input.

## Shape Calculations:

* For each shape, the program prompts the user to input the required values (radius, side length, length, width) using input or input\_msg messages.
* It reads the user's input using appropriate input functions (readfloat or readint) and stores the value in the respective variables.
* The program then performs the necessary calculations (perimeter, area) using the given formulas and stores the results in the respective variables.
* Finally, it displays the calculated results using appropriate output messages (output, output\_1, output\_msg1, output\_msg2, output\_msg3, output\_msg4).

## Error Handling:

* The program includes error handling for invalid inputs. If the user enters an invalid value, it displays a default error message (default\_msg) and prompts the user to enter the correct input.

## Looping and Program Termination:

* After performing calculations and displaying results, the program calls the Choice\_Function to ask the user if they want to continue or exit.
* If the user chooses to continue, it loops back to the switch statement menu.
* If the user chooses to exit, the program jumps to the done label and terminates.

Overall, the program uses input/output operations, conditional branching, looping, and switch statements to provide a user-friendly interface for calculating the area and perimeter of different shapes. It ensures error handling for invalid inputs and allows the user to continue or exit the program as desired.

Note: The program uses other Irvine32 library functions such as WriteString, ReadString, exit, sizeof, Crlf, and ret to handle input/output operations, program termination, and control flow.

# CODE:

## Input:

; Include the Irvine32 library

include Irvine32.inc

; Data section to declare variables

.data

; Take the name from user

Department\_Name byte "------------------------------------------Department BS Cyber Security-------------------------------------",0

Subject\_Name byte "-----------------------------------Computer Organization & Assembly Language-------------------------------",0

Project\_Title byte "------------------------Calculation of Area and Perimeter of Circle, Square, Rectangle---------------------",0

Teacher\_Name byte "Submitted To: Mam Maryam Malik",0

Student\_Name byte "Submitted By: 211077 Huzaifa-Ur-Rehman BSCYS-4B",0

; Switch statement

; Declare a message to show when an invalid input is given

default\_msg byte "Invalid Input! Please Enter The Correct Input. ",0

; Declare messages to show options to the user

meter\_1 byte " meter (m)",0

meter\_2 byte " meter square (m^2)",0

message\_1 byte "Select the Option that You Want to perform.",0

message\_2 byte "1. Calculate the perimeter of the Circle.",0

message\_3 byte "2. Calculate the Area of the Circle.",0

message\_4 byte "3. Calculate the perimeter of the Square.",0

message\_5 byte "4. Calculate the Area of the Square.",0

message\_6 byte "5. Calculate the perimeter of the Rectangle.",0

message\_7 byte "6. Calculate the Area of the Rectangle.",0

message\_8 byte "7. Exit.",0

message\_9 byte "Choice-> ",0

; Declare a variable to store user's choice

choice dword ?

; Declare a message to ask user if they want to continue or not

again\_choice byte "Do You Want to go to the main menu (y/n): ",0

; Declare a variable to store user's choice for continuing or not

cho byte ?

; Circle Part

; Declare a message to get input from user

input byte "Enter the radius of the circle: ",0

; Declare a variable to store user's input

val real4 ?

; Declare messages to show output

output byte "The Area of circle is: ",0

area2 real4 ?

output\_1 byte "The perimeter of circle is: ",0

perimeter2 real4 ?

; Declare PI and TWO constants

PI real4 3.1416

TWO real4 2.0

; Square Part

; Declare a message to get input from user

input\_msg byte "Enter the side length of square: ",0

; Declare a variable to store user's input

side real4 ?

; Declare messages to show output

output\_msg1 byte "The perimeter of the Square is: ",0

perimeter real4 ?

output\_msg2 byte "The Area of the Square is: ",0

area real4 ?

; Declare FOUR constant

FOUR real4 4.0

; Rectangle part

; Declare messages to get inputs from user

input\_msg1 byte "Enter the length of Rectangle: ",0

length1 real4 ?

input\_msg2 byte "Enter the width of Rectangle: ",0

width1 real4 ?

; Declare messages to show output

output\_msg3 byte "The perimeter of the Rectangle is: ",0

perimeter1 real4 ?

output\_msg4 byte "The Area of the Rectangle is: ",0

area1 real4 ?

; Declare an integer variable to store the user's input as a menu choice

inte1 dword ?

.code

Display\_Name\_Function PROC

mov edx, offset Department\_Name

call writestring

call crlf

mov edx, offset Subject\_Name

call writestring

call crlf

mov edx, offset Project\_Title

call writestring

call crlf

mov edx, offset Teacher\_Name

call writestring

call crlf

mov edx, offset Student\_Name

call writestring

call crlf

call WaitMsg

call Clrscr

ret

Display\_Name\_Function ENDP

Choice\_Function PROC

mov edx, offset again\_choice ; Prompts the user to enter a choice

call writestring

mov edx, 0 ; clear edx to read a single character

call readchar ; read a single character from standard input

call crlf

call WaitMsg

call Clrscr

mov cho, al ; store the character in cho

cmp cho, 'y'

je switch\_Statement

jmp done

ret ; Returns control to the calling code.

Choice\_Function ENDP

main PROC

call Display\_Name\_Function

; Switch Statements Messages

switch\_Statement:: ; Global Label

mov edx, offset message\_1

call writestring

call crlf

mov edx, offset message\_2

call writestring

call crlf

mov edx, offset message\_3

call writestring

call crlf

mov edx, offset message\_4

call writestring

call crlf

mov edx, offset message\_5

call writestring

call crlf

mov edx, offset message\_6

call writestring

call crlf

mov edx, offset message\_7

call writestring

call crlf

mov edx, offset message\_8

call writestring

call crlf

mov edx, offset message\_9

call writestring

call readint

mov choice, eax

; Switch Statements Start

; Compares 'choice' to the values 1 through 7 using 'cmp' instruction

mov ebx, choice

cmp ebx,1

je case\_1

cmp ebx,2

je case\_2

cmp ebx,3

je case\_3

cmp ebx,4

je case\_4

cmp ebx,5

je case\_5

cmp ebx,6

je case\_6

cmp ebx,7

je case\_7

jne default

case\_1:

; Input the radius from the user

intput\_Jump:

mov edx, offset input

call writestring

call readfloat

fstp val ; pops the top value from the FPU stack and stores it in val

fld val ; loads the floating-point value onto the top of the FPU stack.

fistp inte1 ; converts the floating-point value to an integer

cmp inte1, 1

jge input\_1

mov edx, offset default\_msg

call writestring

call crlf

jmp intput\_Jump

input\_1:

; calculate perimeter

fld TWO

fld PI

fmul

fstp perimeter2

fld perimeter2

fld val

fmul

fstp perimeter2

fld perimeter2

; Output of Perimeter

mov edx, offset output\_1

call writestring

call writefloat

mov edx, offset meter\_1

call writestring

call crlf

call Choice\_Function

case\_2:

; Input the radius from the user

intput\_Jump1:

mov edx, offset input

call writestring

call readfloat

fstp val

fld val

fistp inte1

cmp inte1, 1

jge input\_2

mov edx, offset default\_msg

call writestring

call crlf

jmp intput\_Jump1

input\_2:

; calculate area

fld val

fld val

fmul

fstp area2 ; pops the top value from the FPU stack and stores it in val

fld area2 ; loads the floating-point value onto the top of the FPU stack.

fld PI

fmul

fstp area2

fld area2

; Output of Area

mov edx, offset output

call writestring

call writefloat

mov edx, offset meter\_2

call writestring

call crlf

call Choice\_Function

case\_3:

; Take the side as the Input From User

input\_Jump2:

mov edx, offset input\_msg

call writestring

call readfloat

fstp side

fld side

fistp inte1

cmp inte1, 1

jge input\_3

mov edx, offset default\_msg

call writestring

call crlf

jmp input\_Jump2

input\_3:

; Calculate Perimeter

fld FOUR

fld side

fmul

;call writefloat

fstp perimeter

fld perimeter

; Result Primeter

mov edx, offset output\_msg1

call writestring

call writefloat

mov edx, offset meter\_1

call writestring

call Crlf

call Choice\_Function

case\_4:

; Take the side as the Input From User

intput\_Jump3:

mov edx, offset input\_msg

call writestring

call readfloat

fstp side

fld side

fistp inte1

cmp inte1, 1

jge input\_4

mov edx, offset default\_msg

call writestring

call crlf

jmp intput\_Jump3

input\_4:

; Calculate Area

fld side

fld side

fmul

fstp area

fld area

; Result Area

mov edx, offset output\_msg2

call WriteString

call writefloat

mov edx, offset meter\_2

call writestring

call Crlf

call Choice\_Function

case\_5:

; Take the length as the Input From User

input\_Jump4:

mov edx, offset input\_msg1

call writestring

call readfloat

fstp length1

fld length1

fistp inte1

cmp inte1, 1

jge input\_Jump5

mov edx, offset default\_msg

call writestring

call crlf

jmp input\_Jump4

input\_Jump5:

; Take the width as the Input From User

mov edx, offset input\_msg2

call writestring

call readfloat

fstp width1

fld width1

fistp inte1

cmp inte1, 1

jge input\_5

mov edx, offset default\_msg

call writestring

call crlf

jmp input\_Jump5

input\_5:

; Calculate Perimeter

fld length1

fld width1

fadd

fstp perimeter1

fld perimeter1

fld TWO

fmul

fstp perimeter1

fld perimeter1

; Result Primeter

mov edx, offset output\_msg3

call writestring

call writefloat

mov edx, offset meter\_1

call writestring

call Crlf

call Choice\_Function

case\_6:

; Take the length as the Input From User

input\_Jump6:

mov edx, offset input\_msg1

call writestring

call readfloat

fstp length1

fld length1

fistp inte1

cmp inte1, 1

jge input\_Jump7

mov edx, offset default\_msg

call writestring

call crlf

jmp input\_Jump6

input\_Jump7:

; Take the width as the Input From User

mov edx, offset input\_msg2

call writestring

call readfloat

fstp width1

fld width1

fistp inte1

cmp inte1, 1

jge input\_6

mov edx, offset default\_msg

call writestring

call crlf

jmp input\_Jump7

input\_6:

; Calculate Area

fld length1

fld width1

fmul

fstp area1

fld area1

; Result Area

mov edx, offset output\_msg4

call WriteString

call writefloat

mov edx, offset meter\_2

call writestring

call Crlf

call Choice\_Function

case\_7:

jmp done

default:

call WaitMsg

call Clrscr

mov edx, offset default\_msg

call writestring

call crlf

jmp switch\_statement

done::

; Exit the program

exit

main ENDP

END main

# Output:





