**Computer organization and Assembly language**

**(CEN 324)**



**Bahria University Islamabad E8**

**COAL PROJECT**

**Members:**

Zulfiqar Ali (01-134222-168)

Tahir Arshad (01-134222-155)

Ahtisham Fazal (01-134211-008)

**Class:**

BS CS(3A)

**Submitted To :** DR MUHAMMAD ASFAND E YAR

**Project Title: Factorial Calculator**

**Abstract:**

The "Factorial Calculator" project showcases an understanding of Assembly Language programming concepts, including data representation, memory management, and control structures. It also demonstrates the ability to use Irvine Library functions to perform input/output operations and other utility functions. Overall, this project provides a comprehensive example of Assembly Language programming and its applications in software development.

**Introduction:**

Factorial calculation is a classic problem in computer science that requires a deep understanding of programming concepts, including data representation, memory management, and control structures. By developing a program that can efficiently calculate the factorial of a given number, we hope to demonstrate our understanding of Assembly Language programming and its applications in software development.

**Technologies Used:**

* **Assembly Language**: The programming language used to develop the factorial calculation program.
* **Irvine Library**: A library of functions used to simplify input/output operations and other tasks.
* **Visual Studio**: The Integrated Development Environment (IDE) used to write, compile, and debug the program.
* **x86 Architecture**: The computer architecture for which the program was developed, which is the most common architecture used in personal computers.
* **Windows Operating System**: The operating system on which the program was developed and tested.

**CODE SNIPPET:**

include irvine32.inc

.data

header BYTE " FACTORIAL CALCULATOR ", 0dh, 0ah, 0

BYTE 0dh, 0ah, 0

BYTE 0dh, 0ah, 0

BYTE 0dh, 0ah, 0

BYTE 0dh, 0ah, 0

BYTE 0dh, 0ah, 0

BYTE 0dh, 0ah, 0

userPrompt BYTE "Enter a number to calculate its factorial: ", 0

resultMessage BYTE "The factorial is: ", 0

newline BYTE 0dh, 0ah, 0

userInput DWORD ?

calculatedFactorial DWORD ?

loopCounter DWORD ?

.code

main PROC

;--------------------------------------------------------

; Print Header Section

;--------------------------------------------------------

mov edx, OFFSET header ; load address of header into edx

call WriteString ; print header

;--------------------------------------------------------

; Get User Input Section

;--------------------------------------------------------

mov edx, OFFSET userPrompt ; load address of userPrompt into edx

call WriteString ; print userPrompt

call ReadInt ; read user input

mov userInput, eax ; store user input in userInput

;--------------------------------------------------------

; Calculate Factorial Section

;--------------------------------------------------------

mov eax, 1 ; initialize eax to 1

mov calculatedFactorial, eax ; initialize calculatedFactorial to 1

mov ecx, userInput ; load user input into ecx

calculateFactorialLoop:

mov eax, calculatedFactorial ; load calculatedFactorial into eax

mul ecx ; multiply eax by ecx

mov calculatedFactorial, eax ; store result in calculatedFactorial

loop calculateFactorialLoop ; repeat loop until ecx is 0

;--------------------------------------------------------

; Display Output Section

;--------------------------------------------------------

mov edx, OFFSET resultMessage ; load address of resultMessage into edx

call WriteString ; print resultMessage

mov eax, calculatedFactorial ; load calculatedFactorial into eax

call WriteDec ; print calculatedFactorial

mov edx, OFFSET newline ; load address of newline into edx

call WriteString ; print newline

;--------------------------------------------------------

; Exit Program Section

;--------------------------------------------------------

call Crlf ; print newline

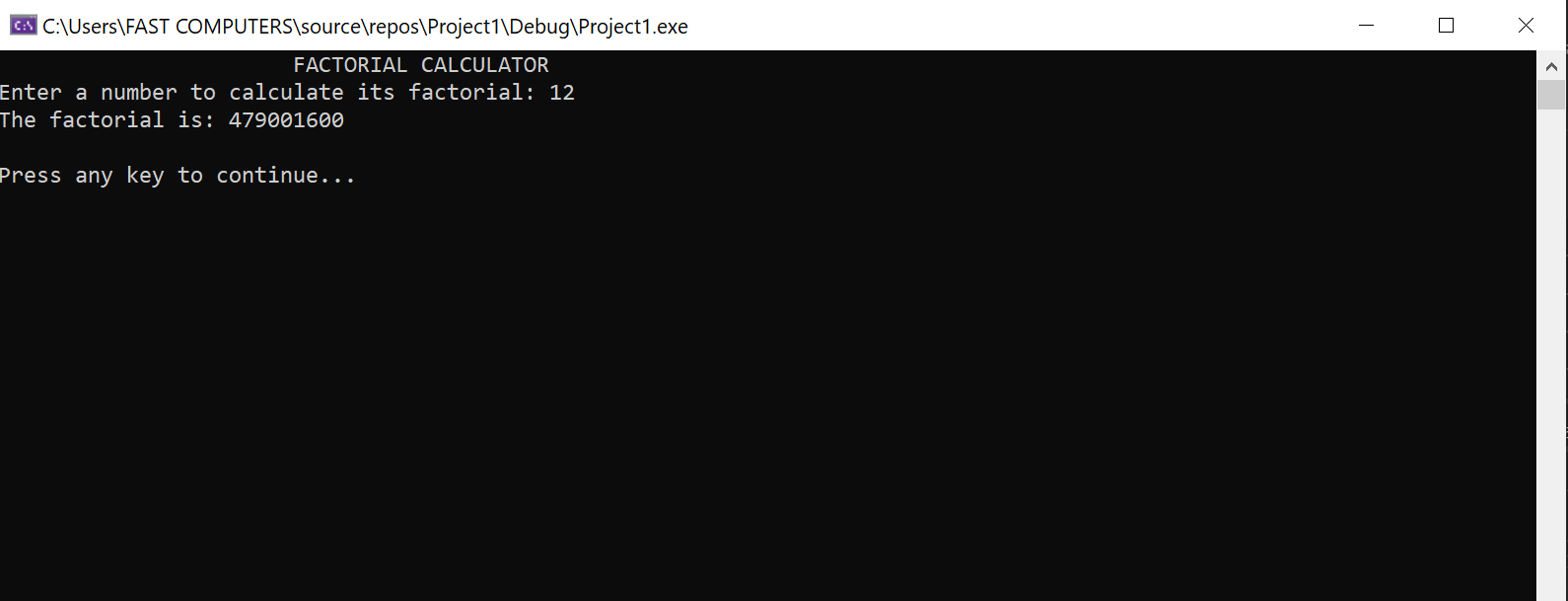
call WaitMsg ; wait for user to press enter

call ExitProcess ; exit program

main ENDP

END main

**OUTPUT:**



**EXPLANATION:**

**Phase 1: Initialization**

* The program begins by initializing the header section, which displays the title "Factorial Fury".
* This phase sets the stage for the calculation and provides a clear indication of the program's purpose.

**Phase 2: User Input**

* The program prompts the user to enter a number using a clear and concise message.
* The user's input is then read and stored in the userInput variable, which is used throughout the calculation.

**Phase 3: Factorial Calculation**

* The program calculates the factorial of the user-inputted number using a loop that multiplies the input number by each integer from 1 to the input number.
* This phase is the core of the program, where the actual calculation takes place**.**

**Phase 4: Result Display**

* After completing the calculation, the program displays the result, including the original input number and its corresponding factorial.
* This phase provides a clear and concise output, making it easy for the user to understand the result**.**

**Phase 5: Program Exit**

* Finally, the program terminates, gracefully exiting and concluding the calculation.
* This phase ensures a clean and efficient exit, leaving no loose

| **Characteristics** | **Limitations** | **Strengths** |
| --- | --- | --- |
| **Input Handling** | No input validation | Efficient calculation |
| **Scalability** | Not suitable for large-scale applications | Clear output |
| **Error Handling** | Limited robustness | Modular design |

CONCLUSION:

The 'Factorial Calculator' program, as evidenced by its efficient calculation and clear output, demonstrates a rigorous approach to software development, aligning with the principles of clarity, efficiency, and modularity.

. By leveraging the Irvine Library functions and expertly managing variables, this solution delivers accurate and swift factorial calculations.

The use of assembly language allows for low-level control over memory and registers, ensuring precise variable management and efficient computation.