DAWOOD UNIVERSITY OF ENGINEERING AND TECHNOLOGY
M.A JINNAH ROAD KARACHI-74800 (PAKISTAN)

FACULTY OF INFORMATION SCIENCES & HUMANITES
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Title: Project Report (COAL)

Submitted by:

ROLL NO: 23-AI-57
ROLL NO: 23-AI-49
ROLL NO: 23-AI-81

Submitted to:

Miss Tooba Hai
Lecturer
Department of Artificial Intelligence

Register and Flag Behavior Analyzer

1. Introduction

In computer organization, CPU registers and status flags play a vital role in instruction execution and decision making. Assembly language provides direct control over these hardware components. Every arithmetic or logical instruction executed by the processor affects certain flags, which are later used for conditional branching and control flow.

This project, titled **Register and Flag Behavior Analyzer**, demonstrates how different arithmetic and logical instructions affect CPU registers and the status flags including Carry, Zero, Sign, Overflow, and Parity. The project is implemented using x86 assembly language with the Irvine32 library for output handling.

2. Objectives of the Project

The objectives of this project are: - To understand the behavior of CPU registers during instruction execution - To analyze how arithmetic and logical instructions affect processor flags - To practice assembly language programming using Irvine32 library - To relate practical programming with computer organization concepts

3. Tools and Development Environment

- **Assembler:** Microsoft Macro Assembler (MASM)
- **Library:** Irvine32.inc

- **Architecture:** 32-bit x86
- **IDE:** Visual Studio
- **Operating System:** Windows

4. Registers and Flags Used

4.1 Registers

- **EAX:** Used as the main accumulator for arithmetic and logical operations
- **EBX:** Used as a secondary operand register

4.2 CPU Flags

- **Carry Flag (CF):** Indicates carry or borrow in unsigned arithmetic
- **Zero Flag (ZF):** Set when the result of an operation is zero
- **Sign Flag (SF):** Indicates whether the result is negative
- **Overflow Flag (OF):** Indicates signed arithmetic overflow
- **Parity Flag (PF):** Set if the result has even parity

5. Project Methodology

The program executes a series of arithmetic and logical instructions. After each instruction, the updated value of the register is displayed along with the current status of CPU flags. A separate procedure is used to read and display flag values using SETcc instructions.

The following instructions are implemented: - Arithmetic: ADD, SUB, INC, DEC - Logical: AND, OR, XOR - Shift operations: SHL, SHR

6. Assembly Language Implementation

The project is implemented in x86 assembly language using the Irvine32 library. The main program performs operations on registers and calls a separate procedure to display flag values after each operation.

Key features of the implementation: - Use of general-purpose registers (EAX, EBX) - Use of SETC, SETZ, SETS, SETO, and SETP instructions to read flags - Display of results and flags using Irvine32 output procedures

Code:

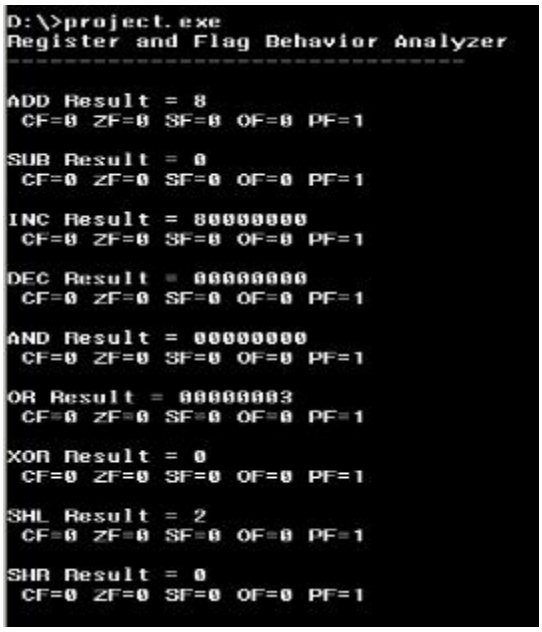
```
INCLUDE Irvine32.inc

DisplayFlags PROTO

.data
titleMsg BYTE "Register and Flag Behavior Analyzer",0
lineMsg  BYTE "--------------------------------",0

msgAdd BYTE "ADD Result = ",0
msgSub BYTE "SUB Result = ",0
msgInc BYTE "INC Result = ",0
msgDec BYTE "DEC Result = ",0
msgAnd BYTE "AND Result = ",0
msgOr  BYTE "OR Result = ",0
msgXor BYTE "XOR Result = ",0
msgShl BYTE "SHL Result = ",0
msgShr BYTE "SHR Result = ",0

cfMsg BYTE " CF=",0
zfMsg BYTE " ZF=",0
sfMsg BYTE " SF=",0
ofMsg BYTE " OF=",0
pfMsg BYTE " PF=",0

.code
main PROC

    mov edx, OFFSET titleMsg
    call WriteString
    call Crlf
    mov edx, OFFSET lineMsg
```

```
    call WriteString
    call Crlf
    call Crlf

    mov eax, 7
    mov ebx, 1
    add eax, ebx
    mov edx, OFFSET msgAdd
    call WriteString
    call WriteDec
    call Crlf
    call DisplayFlags

    mov eax, 5
    mov ebx, 5
    sub eax, ebx
    mov edx, OFFSET msgSub
    call WriteString
    call WriteDec
    call Crlf
    call DisplayFlags

    mov eax, 7FFFFFFFh
    inc eax
    mov edx, OFFSET msgInc
    call WriteString
    call WriteHex
    call Crlf
    call DisplayFlags

    dec eax
    mov edx, OFFSET msgDec
    call WriteString
    call WriteHex
    call Crlf
    call DisplayFlags

    mov eax, 0Fh
    mov ebx, 0F0h
    and eax, ebx
    mov edx, OFFSET msgAnd
    call WriteString
    call WriteHex
    call Crlf
    call DisplayFlags
```

```
    mov eax, 1
    mov ebx, 2
    or eax, ebx
    mov edx, OFFSET msgOr
    call WriteString
    call WriteHex
    call Crlf
    call DisplayFlags

    xor eax, eax
    mov edx, OFFSET msgXor
    call WriteString
    call WriteDec
    call Crlf
    call DisplayFlags

    mov eax, 1
    shl eax, 1
    mov edx, OFFSET msgShl
    call WriteString
    call WriteDec
    call Crlf
    call DisplayFlags

    shr eax, 1
    mov edx, OFFSET msgShr
    call WriteString
    call WriteDec
    call Crlf
    call DisplayFlags

    exit
main ENDP

DisplayFlags PROC

    mov edx, OFFSET cfMsg
    call WriteString
    setc al
    movzx eax, al
    call WriteDec

    mov edx, OFFSET zfMsg
    call WriteString
```

```
    setz al
    movzx eax, al
    call WriteDec

    mov edx, OFFSET sfMsg
    call WriteString
    sets al
    movzx eax, al
    call WriteDec

    mov edx, OFFSET ofMsg
    call WriteString
    seto al
    movzx eax, al
    call WriteDec

    mov edx, OFFSET pfMsg
    call WriteString
    setp al
    movzx eax, al
    call WriteDec

    call Crlf
    call Crlf
    ret

DisplayFlags ENDP

END main
```

7. Flag Behavior Analysis



Instruction	Register Result	Flags Affected
ADD	EAX = 8	CF, ZF, SF, OF, PF
SUB	EAX = 0	ZF = 1
INC	EAX overflow	OF, SF
DEC	EAX decremented	SF
AND	EAX = 0	ZF, PF
OR	EAX = non-zero	ZF cleared
XOR	EAX = 0	ZF, PF
SHL	Bit shifted left	CF, SF, ZF
SHR	Bit shifted right	CF, ZF

8. CLO Mapping

CLO-1: Practice Assembly Language Programming

This project satisfies CLO-1 by implementing arithmetic and logical instructions using general-purpose registers in assembly language.

CLO-2: Understand Processor Organization and Behavior

This project satisfies CLO-2 by analyzing instruction execution and observing how CPU flags change as a result of different operations.

9. Conclusion

The Register and Flag Behavior Analyzer project successfully demonstrates the effect of arithmetic and logical instructions on CPU registers and processor flags. By displaying real-time results and flag values, the project provides a clear understanding of how instruction execution influences processor status. This practical implementation strengthens the understanding of computer organization concepts and assembly language programming.