**National University of Computer and Emerging Sciences**

Logo

Description automatically generated with medium confidence

**Lab Experiment 03**

**Computer Organization and Assembly Language Lab**

Department of Computer Science

FAST-NU, Lahore, Pakistan

**REGISTERS AND FLAGS**

# OBJECTIVES

∙ How to interpret the different types of registers available in the IAPX86 architecture. ∙ How to use the different types of registers and how to manipulate them in assembly language.

∙ How to interpret the function of flags and to use them effectively.

∙ How to perform arithmetic operations with registers.

∙ How to use the debugger for viewing the available registers and their function.

# THEORY

The 8086 chip uses registers for performing operations. Following is a brief detail of their functions:

**General registers**: There are four 16-bit general-purpose registers used primarily to contain operands for arithmetic and logical operations.

**Segment registers:** These special-purpose registers permit systems software designers to choose either a flat or segmented model of memory organization. These four registers determine, at any given time, which segments of memory are currently addressable.

**Status and instruction registers:** These special-purpose registers are used to record and alter certain aspects of the 8086 processor state.

**Flags Register:** FLAGS is a 16-bit register which should be interpreted as a collection of single bit flags rather than as a unit. The flags may be considered in two major groups: the status flags and the control flags.

**Status Flags:** The status flags of the FLAGS register allow the results of one instruction to influence later instructions. The arithmetic instructions use OF, SF, ZF, AF, PF, and CF. The SCAS (Scan String), CMPS (Compare String), and LOOP instructions use ZF to signal that their operations are complete. There are instructions to set, clear, and complement CF before execution of an arithmetic instruction.

**Control Flags:** The control flag DF of the EFLAGS register controls string instructions. Setting DF causes string instructions to auto-decrement; that is, to process strings from high addresses to low addresses. Clearing DF causes string instructions to autoincrement, or to process strings from low addresses to high addresses.

**Instruction Pointer:** The instruction pointer register (IP) contains the offset address, relative to the start of the current code segment, of the next sequential instruction to be executed. The

Page | 9

Lab Manual of Computer Organization and Architecture

instruction pointer is not directly visible to the programmer; it is controlled implicitly by control transfer instructions, interrupts, and exceptions.

Shape

Description automatically generated with medium confidence

**Figure 1 Control Flags**

Page | 10

Lab Manual of Computer Organization and Architecture

|  |  |  |
| --- | --- | --- |
| **Segment Registers** |  |  |
| CS | Code Segment | 16-bit number that points to the active codesegment |
| DS | Data Segment | 16-bit number that points to the active datasegment |
| SS | Stack Segment | 16-bit number that points to the active stacksegment |
| ES | Extra Segment | 16-bit number that points to the active extrasegment |
| **Pointer**  **Registers** |  |  |
| IP | Instruction Pointer | 16-bit number that points to the offset of the next instruction |
| SP | Stack Pointer | 16-bit number that points to the offset that the stack is using |
| BP | Base Pointer | used to pass data to and from the stack |
| **General**  **Purpose**  **Registers** |  |  |
| AX | Accumulator Register | mostly used for calculations and for input/output |
| BX | Base Register | Only register that can be used as an index |
| CX | Count Register | register used for the loop instruction |
| DX | Data Register | input/output and used for multiply and divide |
| **Index**  **Registers** |  |  |
| SI | Source Index | used by string operations as source |
| DI | Destination Index | used by string operations as destination |

## Q1

Give the value of the zero flag, the carry flag, the sign flag, and the overflow flag after each of the following instructions if AX is initialized with 0x1254 and BX is initialized with 0x0FFF. a. add ax, 0xEDAB b. add ax, bx c. add bx, 0xF001

## Q2

Write a program to calculate the square of number. Display the result in accumulator (AX).

**Hint:** Example 2.7

## Q3

In computing, endianness is the order or sequence of bytes of a word of digital data in computer memory. Endianness is primarily expressed as big-endian (BE) or little-endian (LE). A big-endian system stores the most significant byte of a word at the smallest memory address and the least significant byte at the largest. A little-endian system, in contrast, stores the least-significant byte at the smallest address.

Write a code so that you can prove wither the x88 architecture is big-endian or little-endian.