|  |
| --- |
| Experiment No. 7 |
| Implement 8068 assembly language program for performing various arithmetic operation |
| Date of Performance: |
| Date of Correction: |

**Aim:** To write and execute 8086 Assembly Language Programs (ALPs) that perform basic arithmetic operations such as addition, subtraction, multiplication, and division.

**Objective:** To understand how arithmetic operations are performed at the microprocessor level using 8086 assembly instructions.

**Theory:**

**Introduction to 8086 Microprocessor:**

The **Intel 8086** is a 16-bit microprocessor and a foundation for the x86 family of processors. It supports various arithmetic, logical, and control instructions that allow programmers to directly manage the CPU's registers, memory access, and I/O ports.

8086 has:

* 16-bit ALU (Arithmetic Logic Unit)
* 16-bit registers like AX, BX, CX, DX (general-purpose)
* Segment registers: CS, DS, SS, ES
* Instruction Pointer (IP), Flags register
* Supports both byte (8-bit) and word (16-bit) operations

**Arithmetic Instructions in 8086:**

The 8086 instruction set provides several **arithmetic instructions** that operate on registers or memory operands.

|  |  |
| --- | --- |
| **Instruction** | **Description** |
| ADD | Adds two operands |
| SUB | Subtracts source from destination |
| INC | Increments operand by 1 |
| DEC | Decrements operand by 1 |
| MUL | Unsigned multiplication |
| IMUL | Signed multiplication |
| DIV | Unsigned division |
| IDIV | Signed division |
| NEG | Changes sign of the operand (2’s complement) |

These instructions can work with:

* Register-to-register (e.g., ADD AX, BX)
* Immediate-to-register (e.g., SUB AX, 0005H)
* Memory-to-register or register-to-memory (e.g., ADD AL, [5000H])

**Arithmetic Operations Explained:**

1. **Addition (ADD)**: Adds values and updates flags like Carry (CF), Overflow (OF), Zero (ZF).
   * Example: ADD AX, BX → AX = AX + BX
2. **Subtraction (SUB)**: Subtracts source from destination.
   * Example: SUB AX, BX → AX = AX - BX
3. **Multiplication (MUL/IMUL)**: Multiplies accumulator with operand.
   * Example: MUL BX → AX = AL \* BX (for 8-bit); DX:AX = AX \* BX (for 16-bit)
4. **Division (DIV/IDIV)**: Divides accumulator by operand.
   * Result stored in AX (quotient) and DX (remainder) for 16-bit division.

**Why Learn Arithmetic in Assembly?**

* Provides **low-level control** over hardware.
* Helps understand **how compilers translate high-level arithmetic** into machine code.
* Essential for **embedded systems**, **OS development**, and **microcontroller programming**.
* Used in writing **bootloaders**, **interrupt handlers**, and **system utilities**.

**Tools Used:**

* **EMU8086** / **MASM** / **TASM** (assemblers and emulators for 8086)
* Simulators allow students to write, run, and debug ALPs.

**Solution:**

**Conclusion:** We learned how to write assembly programs for arithmetic operations and understood how low-level instructions work with registers in the 8086 microprocessor.