**EXPERIMENT 5**

**Aim:**Program for 16 bit BCD addition

**LO:** 3

**LO STATEMENT:** Build a program on a microprocessor using arithmetic & logical instruction set of 8086.

**Software and Hardware Requirements:** TASM Software

**Theory:**

**1.** **MOV** **Instruction:**

The MOV instruction is the most important command in the 8086 because it moves data from one location to another. It also has the widest variety of parameters; so the assembler programmer can use MOV effectively, the rest of the commands are easier to understand. MOV copies the data in the source to the destination. The data can be either a byte or a word. Sometimes this has to be explicitly stated when the assembler cannot determine from the operands whether a byte or word is being referenced.

**Syntax:**

Move Destination, Source

**Example:**

MOV Ax, Bx

**2.** **ADD** **Instructions:**

The ADD instructions are used for performing simple addition of binary data in byte, word and doubleword size, i.e., for adding or subtracting 8-bit, 16-bit or 32-bit operands, respectively.

The ADD/SUB instruction can take place between −

o Register to register o Memory to register o Register to memory

o Register to constant data o Memory to constant data

**Syntax:**

ADD destination, source

**Example:**

ADD AX,BX

**3.** **DAA** **Instruction:**

The DAA (Decimal Adjust after Addition) instruction allows addition of numbers represented in 8-bit packed BCD code. It is used immediately after normal addition instruction operating on BCD codes. This instruction assumes the AL register as the source and the destination, and hence it requires no operand.

**Syntax:**

DAA

**Example:**

ADD AL,BL

DAA

**4.** **ADC** **Instruction:**

Adds the destination operand (first operand), the source operand (second operand), andthe carry (CF) flag and stores the result in the destination operand.

The destination operand can be a register or a memory location; the source operand can be an immediate, a register, or a memory location. (However, two memory operands cannot be used in one instruction.)

The state of the CF flag represents a carry from a previous addition When an immediate value is used as an operand, it is sign-extended to the length of the destination operand format. The ADC instruction does not distinguish between signed or unsigned operands. Instead, the processor evaluates the result for both data types and sets the OF and CF

flags to indicate a carry in the signed or unsigned result, respectively. The SF flag indicates the sign of the signed result. The ADC instruction is usually executed as part of a multibyte or multiword addition in which an ADD instruction is followed by an ADC instruction.

**Example:**

ADC AX,BX

**Code:**

Assume CS: Code, DS: Data

Data Segment

n1 dw 1274H

n2 dw 5608H

ans dw 01 dup(?)

Data ends

Code Segment

Start: Mov Ax,Data

Mov Ds,Ax

Mov Ax,n1

Mov Bx,n2

Add AL,BL

DAA

Mov CL,AL

Mov AL,AH

Adc AL,BH

DAA

Mov CH,AL

Mov ans,Cx

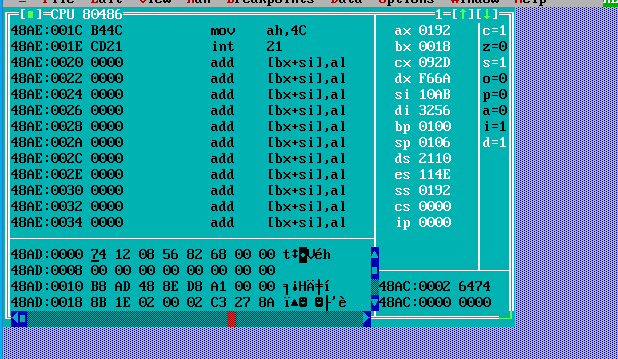
Mov AH,4CH

INT 21H

Code ends

end Start

**Output:**



**Conclusion:**

From this experiment we have learned how to run a 64 bit BCD addition using TASM software.