**Batch: B3 Roll No.: 121**

**Experiment / assignment / tutorial No.\_\_\_9\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:** Implement simple addition, subtraction, multiplication and division instructions using TASM. |

**AIM:** Implement simple addition, subtraction, multiplication and division instructions using TASM.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment: (Mentions the CO/CO’s attained)**

Understand the Central processing unit with addressing modes and working of control unit in depth.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

**1) Microprocessor architecture and applications with 8085: By Ramesh Gaonkar (Penram International Publication).**

**2) 8086/8088 family: Design Programming and Interfacing: By John Uffenbeck (Pearson Education).**

**Pre Lab/ Prior Concepts:**

**Assembler directives: These are statements that direct the assembler to do something**

**Definition:**

**Types of Assembler Directives:**

**ASSUME Directive** - The ASSUME directive is used to tell the assembler that the name of the logical segment should be used for a specified segment. The 8086 works directly with only 4 physical segments: a Code segment, a data segment, a stack segment, and an extra segment.

**Example:**

**ASUME CS:CODE** ;This tells the assembler that the logical segment named CODE contains the instruction statements for the program and should be treated as a code segment.

**ASUME DS:DATA** ;This tells the assembler that for any instruction which refers to a data in the data segment, data will found in the logical segment DATA

**Start:**

It is entry point of the program. without this program won’t run.

**END** - END directive is placed after the last statement of a program to tell the assembler that this is the end of the program module. The assembler will ignore any statement after an END directive. Carriage return is required after the END directive.

**ENDS** - This ENDS directive is used with name ofthe segment to indicate the end of that logic segment.

**Example:**

**CODE SEGMENT** ;

Hear it Start the logic

;segment containing code

; Some instructions statements to perform the logical

;operation

**CODE ENDS** ;End of segment named as;CODE

**Arithmetic instruction set:**

**ADD instruction:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mnemonic** | **Meaning** | **Format** | **Operation** | **Flags** |
|  |  |  |  | **Affected** |
|  |  |  |  |  |
| ADD | Addition | ADD D, S | (S) + (D)🡪(D) | All |
|  |  |  |  |  |
|  |  |  | Carry🡪(CF) |  |
|  |  |  |  |  |
|  |  |  |  |  |
| ADC | Add with | ADC D, S | (S) + (D) +(CF) | All |
|  | Carry |  | 🡪 (D) |  |
|  |  |  | Carry🡪(CF) |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Syntax: ADD destination,source**

**SUB instruction:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mnemonic** | **Meaning** | **Format** | **Operation** | **Flags Affected** |
|  |  |  |  |  |
| SUB | Subtract | SUB D, S | (D) - (S)🡪(D) | All |
|  |  |  |  |  |
|  |  |  | Borrow🡪(CF) |  |
|  |  |  |  |  |
|  |  |  |  |  |
| SBB | Subtract with | SBB D, S | (D) - (S) –(CF)🡪(D) | All |
|  | Borrow |  |  |  |
|  |  |  |  |  |

**MUL instruction:**

**Syntax: MUL source**

|  |  |  |  |
| --- | --- | --- | --- |
| **Multiplication** | **Multiplicand** | **Operand** | **Result** |
| **(MUL or IMUL)** |  | **(Multiplier)** |  |
|  |  |  |  |
| Byte \* Byte | AL | Register or | AX |
|  |  | Memory |  |
|  |  |  |  |
| Word \* Word | AX | Register or memory | DX :AX |

**DIV instruction:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Division** | **Dividend** | **Operand** | **Quotient : Remainder** |
| **(DIV or IDIV)** |  | **(Divisor)** |  |
|  |  |  |  |
| Word / Byte | AX | Register or memory | AL : AH |
|  |  |  |  |
| Dword / Word | DX:AX | Register or memory | AX : DX |
|  |  |  |  |

**The steps to execute a program in TASM are**

**ASSEMBLING AND EXECUTING THE ROGRAM**

1. **Writing an Assembly Language Program**

Assembly level programs generally abbreviated as ALP are written in text editor EDIT.

Type *EDIT* in front of the command prompt **(C:\TASM\BIN)** to open an untitled text file.

*EDIT<file name>*

After typing the program save the file with appropriate file name with an extension *.ASM*

Ex:Add.ASM

1. **Assembling an Assembly Language Program**

To assumble an ALP we needed executable file called MASM.EXE. Only if this file is in current working directory we can assemble the program. The command is

*TASM<filename.ASM>*

If the program is free from all syntactical errors, this command will give the **OBJEC**T file.In case of errors it list out the number of errors, warnings and kind of error.

**Note: No object file is created until all errors are rectified.**

1. **Linking**

After successful assembling of the program we have to link it to get **Executable file.**

The command is

*TLINK<File name.OBJ>*

This command results in <*Filename.exe>*which can be executed in front of the command prompt.

1. **Executing the Program**

Open the program in debugger by the command(note only exe files can be open)by the command.

*<Filename.exe>*

This will open the program in debugger screen where in you can view the assemble code with the CS and IP values at the left most side and the machine code. Register content,memory content also be viewed using ***TD***option of the debugger & to execute the program in single steps(F7)

**Algorithm for adding the two 8-bit numbers:**

DATA SEGMENT

NUM1 DW 1234H

NUM2 DW 1234H

RES DW ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

MOV AX,DATA

MOV DS,AX

MOV AX,NUM1

MOV BX,NUM2

ADD AX,BX

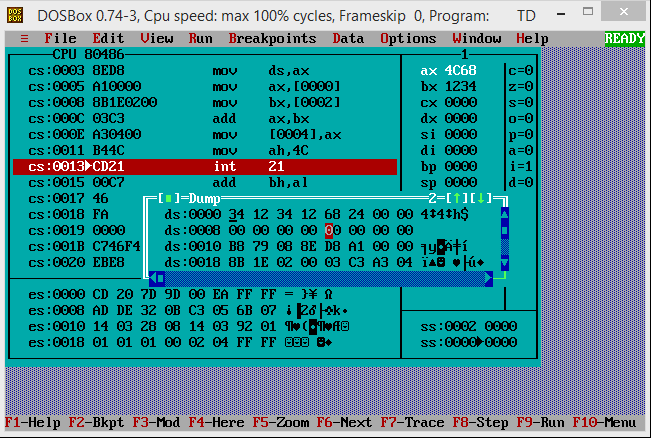
MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START



**Algorithm for subtracting the two 8 bit numbers:**

DATA SEGMENT

NUM1 DW 1255

NUM2 DW 28

RES DW ?

DATA ENDS

CODE SEGMENT

START:

ASSUME CS:CODE,DS:DATA

MOV AX,DATA

MOV DS,AX

MOV AX,NUM1

MOV BX,NUM2

SUB AX,BX

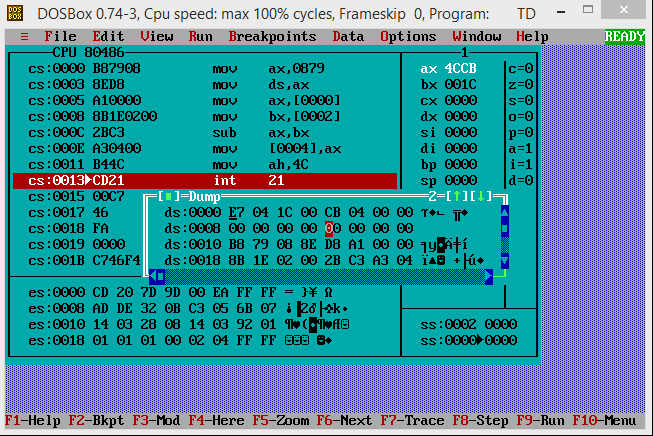
MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START



**Algorithm for multiplying the two 8 bit numbers:**

DATA SEGMENT

NUM1 DW 1234H

NUM2 DW 1234H

RES DW ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:MOV AX,DATA

MOV DS,AX

MOV AX,NUM1

MOV BX,NUM2

MUL BX

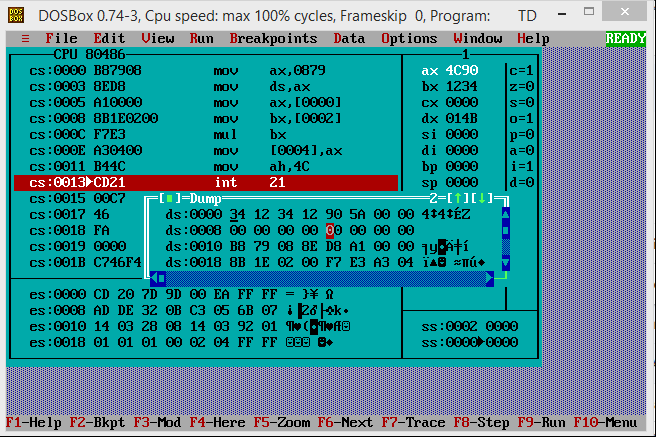
MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START



**Algorithm for dividing the two 8-bit numbers:**

DATA SEGMENT

NUM1 DW 1234H

NUM2 DW 1234H

RES DW ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:MOV AX,DATA

MOV DS,AX

MOV AX,NUM1

MOV BX,NUM2

DIV BX

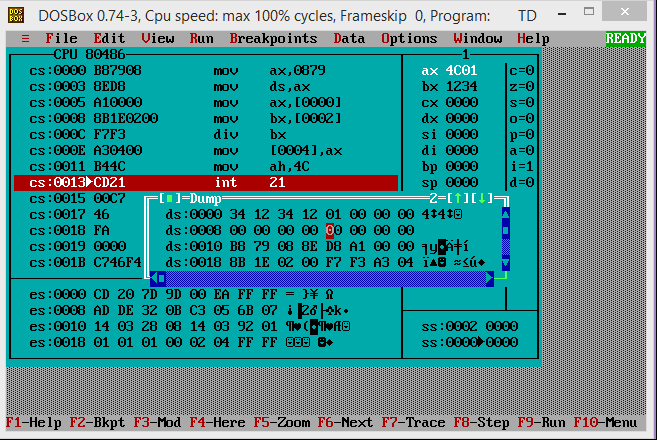
MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START



**Conclusion:**

Thus, in this experiment, the processes of Addition, Subtraction, Multiplication and Division using TASM was learnt and implemented.

**Post Lab Descriptive Questions (Add questions from examination point view)**

**Explain instructions ADC and SBB with example**

**Ans.**

1. **ADC:** The ADC (Add with Carry) instruction adds the destination operand (first operand), the source operand (second operand) and the 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. 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.
2. **SBB:** The SBB instruction adds the source operand (second operand) and the carry flag (CF), and subtracts the result from the destination operand (first operand). The result of the subtraction is stored in the destination operand. The destination operand can be a register or memory location; the source operand can be an immediate, a register, or a memory location. The state of the CF flag represents a borrow from a previous subtraction.

**Date: \_\_\_07-12-22\_\_\_\_ Signature of faculty in-charge**