**FR. Conceicao Rodrigues College of Engineering**

**Department of Computer Engineering**

**1.** **Addition of Two 8/16/32 bit numbers**

**1. Course, Subject & Experiment Details**

| **Academic Year** | **2023-24** | **Estimated Time** | **Experiment No. 1– 02 Hours** |
| --- | --- | --- | --- |
| **Course & Semester** | **S.E. (Comps)**  **– Sem. IV** | **Subject Name** | **Microprocessor** |
| **Chapter No.** | **2** | **Chapter Title** | **Instruction Set and Programming** |
| **Experiment Type** | **Software** | **Subject Code** | **CSC405** |

**Rubrics**

| **Timeline (2)** | **Practical Skill & Applied Knowledge**  **(2)** | **Output**  **(3)** | **Postlab**  **(3)** | **Total**  **(10)** | **Sign** |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

**2. Aim & Objective of Experiment**

**TO ADD TWO 8/16/32 BIT NUMBERS**

**Objective :** Program involves storing the two 8-bit no in memory locations and adding them taking into consideration the carry generated. The objective of this program is to give an overview of arithmetic instructions of 8086 for 8-bit operands

**3. Software Required**

TASM Assembler

**4** . **Brief Theoretical Description**

**Pre-Requisites:** 1. Instructions of microprocessor 8086

2. Addressing mode of microprocessor 8086.

3. Knowledge of TASM directories.

**Theory:** The addressing modes used in program are:

1) Direct addressing mode: in this mode address of operand is directly specified in the instruction. This address is offset address of the segment being indicated by an instruction.

E.g. MOV AL,[2000h]

EA = DS x 10H + 2000H

2) Register Addressing Mode: In this mode operand are specified using registers. Instructions are shorter but operations cannot be identified looking at instruction.

E.g. MOV CL, DL

3) Based Indexed Addressing Mode: The operand address is calculated using base register and index register.

E.g. MOV DX, [BX + SI] moves word from address pointed by BX + SI in data segment to DX.

EA = DS x 10H + BX + SI

4) Base indexed plus displacement: In this mode address of operand is calculated using base register , index register and displacement.

E.g. MOV CX, [BX+DI+10h]

This moves a word from address pointed by BX + DI +10h of segment to CX.

**5. Algorithm:** 1. Initialize the data segment.

2. Store two 8/16 -bit numbers in memory locations.

3. Move the 1st number in any one of the general purpose register.

4. Move the 2nd number in any other general purpose register.

5. Add the 2 numbers.

6. Store the result in memory location.

7. Check for carry flag. If carry flag is set then store '1' as

MSB of result.

8. Stop

# **6. Conclusion:**

1. 8 bit addition without carry:

.8086

.model small

.stack 100h

.data

a db 09h

b db 02h

.code

start:

mov ax,@data

mov ds,ax

mov al,a

mov bl,b

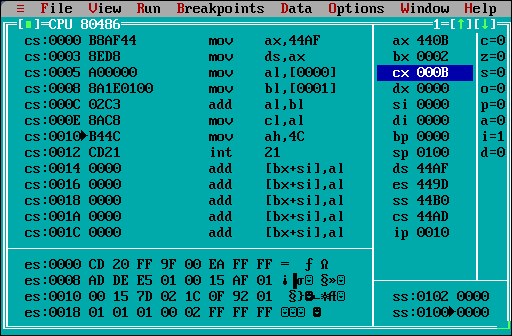
add al,bl

mov cl,al

mov ah,4ch

int 21h

end start



1. 8 bit addition with carry:

.8086

.model small

.stack 100h

.data

a db 99h

b db 88h

sum db ?

carry db ?

.code

start:

mov ax,@data

mov ds,ax

mov al,a

mov bl,b

add al,bl

jnc skip

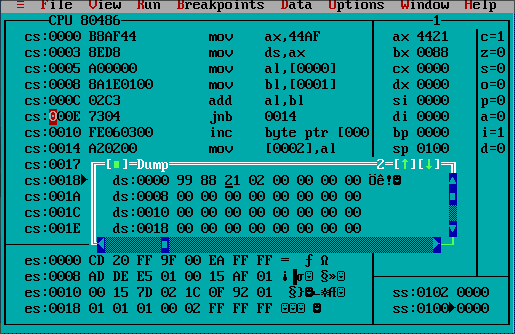
inc carry

skip:

mov sum, al

int 03h

end start



1. 16 bit addition with carry:

.8086

.model small

.data

A dw 1234H

B dw 4567H

sum dw ?

carry db ?

.code

start:

MOV AX, @data

MOV DS, AX

MOV AX, A

MOV BX, B

ADD AX, BX

JNC skip

INC carry

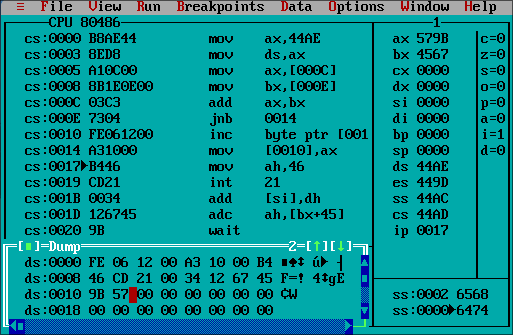
skip:

MOV sum, AX

MOV AH, 46H

INT 21H

end start



**Postlab:**

1. Write a program for addition of two 32 bit numbers ,execute and take the screen shots of the results.

.8086

.model small

.data

A dd 12345678H

B dd 87654321H

sumh dw ?

suml dw ?

.code

start:

MOV AX, @data

MOV DS, AX

LEA SI, A

MOV AX, [SI]

MOV BX, [SI+04H]

ADC AX, BX

MOV suml, AX

MOV AX, [SI+02H]

MOV BX, [SI+06H]

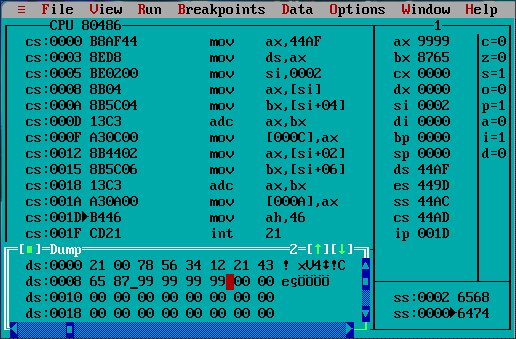
ADC AX, BX

MOV sumh, AX

MOV AH, 46H

INT 21H

end start



1. Write a program to Subtract two 16 bit numbers.

.8086

.model small

.data

A dw 4567H

B dw 1234H

diff dw ?

borrow db ?

.code

start:

MOV AX, @data

MOV DS, AX

MOV AX, A

MOV BX, B

SBB AX, BX

JNC skip

INC borrow

skip:

MOV diff, AX

MOV AH, 46H

INT 21H

end start

