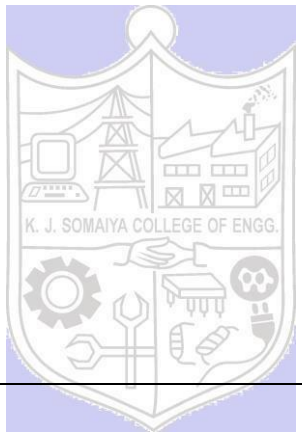

Experiment No. 8

Title: Use of 8086 Simulator

(Programs for arithmetic operations of 8086 using various addressing modes.)

Batch: B1 **Roll No.: 16010420133** **Experiment No.: 8**

Aim: Programs for arithmetic operations of 8086 using various addressing modes.

Resources needed: DosBox, 8086 Assembler-TASM

Theory:

Some addressing modes of the 8086 processor are:-

(i) Immediate Addressing e.g. MOV AX, 1616H
(ii) Register Addressing e.g. ADD AH, DH
(iii) Indirect Addressing e.g. MOV AX, [BX]
(iv) Implied Addressing e.g. XLAT
(v) Indexed Addressing e.g. INC [SI]
(vi) Indexed Addressing with displacement e.g. MOV AL, [BP+SI+10]

Write an assembly language program for given problem statement in lab session. Identify the usage of various addressing modes used in your program.

Algorithm for performing arithmetic operations

Step 1: Initialize data segment.
Step 2: Initialize the operands
Step 3: Declare the result variable Step
4: End of data segment.
Step 5: Initialize code segment.
Step 6: Initialize CS and DS register
Step 7: Load the operands in local general-purpose registers. (Such as AX , BX)
Step 8: Perform the arithmetic operation on operands in local general purpose registers.
Step 9: Load the result in declared variable Step
10: End of code segment.
Step 11: End of program.

Procedure:

a) Draw the flowchart/Algorithm/pseudocode for the problem given in Lab session.
b) Write as assembly language code and compile it using TASM.
c) Observe the memory contents affected while program execution.
d) Observe the output by observing the memory locations, registers and flags affected.

Observations and Results:

Understand basics of assembly language programming for microprocessor 8086.

Observe the memory locations and different registers affected while execution of the program.

Program for 16 Bit Addition Algorithm:

Step 1: Initialize data segment.
Step 2: Take two variable of size word i. e. 16 bits NUM1 and NUM2 and initialize them with some value
Step 3: Take other variable RESULT of size word which will store the result.
Step 4: End of data segment.
Step 5: Initialize code segment.
Step 6: Initialize CS and DS register
Step 7: Load the contents of NUM1 in Ax register.
Step 8: Load the contents of NUM2 in Bx register(for two register addition)
Step 9: Add the content of Ax and Bx and store the result in Ax. (For two register addition)
Add the content of Ax and data stored at the variable NUM2 and store the result in Ax (one register and memory operand)
Step 10: Load the Variable RESULT with contents of Ax Step
11: End of code segment.
Step 12: End of program.

CODE:

```
DATA SEGMENT
NUM1 DW 0005H
NUM2 DW 0002H RESULT
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 RESULT,AX
        MOV AX,4CH
        INT 21H CODE
ENDS
END START
END
```

OUTPUT:

LET NUM1 = 0005H NUM2 = 0002H

AX ← 0005H

BX ←0002H
ADD AX,BX
AX ← 0007H
RESULT 0007H

Program for 16-bit multiplication Algorithm:

Step 1: Initialize DATA segment.
Step 2: Take two variables NUM1 and NUM2 of size word i.e., 16-bit with some value.
Step 3: Take third variables RESULT of size double word to store the result. Step 4: End of DATA segment.
Step 5: Initialize CODE segment
Step 6: Initialize CS and DS registers.
Step 7: Load AX register with value of NUM1.
Step 8: Load BX register with value of NUM2 (for part 'a') / Load BX register with address of NUM2 (for part 'b')
Step 9: Initialize DX with 0000H.
Step 10: Multiply AX and BX and store result in AX and DX (for part 'a') / Multiply AX and data stored at address stored in BX and store result in AX and DX (for part 'b') Step 11: Load variable RESULT with AX and DX.
Step 12: End code segment.
Step 13: End of program.

CODE:

```
DATA SEGMENT
NUM1 DW 0005H
NUM2 DW 0002H RESULT
DD ?
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 [RESULT], DX
        MOV [RESULT+2], AX
        MOV AX,4CH
        INT 21H

CODE ENDS
END START
END
```

OUTPUT:

LET NUM1 = 0005H NUM2
= 0002H
AX ← 0005H
BX ←0002H
MUL BX AX
← 000AH
DX ← 0000H
RESULT 0000 000AH

Outcomes:

CO5- Understand the fundamental concepts of microprocessors.

Conclusion:

Understood basics of assembly language programming for microprocessor 8086 and wrote programs for arithmetic operations of 8086 using various addressing modes.

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

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Douglas Hall, Microprocessors and interfacing, McGraw Hill