Computer Organization and Architecture (EET2211)

LAB II: Analyze and Evaluate the Branching operation in the 8086 Microprocessor.

Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar

Branch: CSE		Section: 2241026	
S. No.	**Name**	**Registration No.**	**Signature**
6	Dinanath Dash	2241004161	

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Find the sum and average of N 16-bit numbers.
2. Count no. of 0's in an 8-bit number.
3. Move a block of 16-bit data from one location to other.
4. Multiplication of two 16-bit numbers without using MUL instruction in direct addressing mode.

II. PRE-LAB

Note: For each objective in prelab describe the following points:

- Write the pseudocode.

For Obj 1 →

i) Start
ii) Declare sum as a 16-bit integer variable and set it to 0.
iii) Declare count as a 16-bit integer variable and set it to 0.
iv) Declare N as a 16-bit integer variable, which represents the number of elements in the array.
v) Declare an array of 16-bit integer with a size of N.
vi) Get the value of N from the user
vii) Get the value of the array elements from the user and store them in the array.
viii) For i from 0 to N-1 a) Add the i-th element of the array to sum
b) Increment count by 1.
ix) Calculate the average by dividing sum by count.
x) Print the sum and the average
xi) Stop

For Obj. 2 →

i) Initialise a variable to store the number of 0's in the number.
ii) Initialise a variable to store the number of 1's in the number.
iii) Load the 8-bit number into the accumulator.
iv) Initialize a loop to go through all 8 bits of the numbers.
v) Rotate the number to the left to get the next bit.
vi) Check the carry flag. If the carry flag is set, it means that the bit that was shifted out was a 1, and so we increment the number of 1's.
vii) If the carry flag is clear, it means that the bit that was shifted out was a 0, and so we increment the number of 0's.

viii) Repeat the previous two steps until all 8 bits of the number have been processed.

ix) Print the number of 0's in the number.

For Obj. 3→

i) Start

ii) Set up the source and destination addresses.

a) MOV SI, source address

b) mov di, destination address

iii) Set up the number of words to be moved

a) mov cx, number of words

iv) Move the data

a) rep mov sm – This instruction moves a word from the source index register to th destination index register.

b) It repeats the previous instruction cx times, so it will move cx number of words.

c) The mov sw instruction increments both source and destination indices automatica

v) End

For Obj. 4→

i) Initialize the product to zero.

a) acc = 0

b) product = 0

ii) Multiply the multiplier by 2

a) multiplier = multiplier * 2

b) mov ax, multiplier * 2

c) shl ax, 1

b) mov multiplier, ax

iii) Check if the multiplier is greater than multiplicand

a) if (multiplier > multiplicand) then

iv) Add the multiplicand to the product

a) Product = product + multiplicand

b) add product, multiplicand

v) shift the multiplier to the right by one

a) multiplier = multiplier >> 1

b) sarc multiplier, 1

vi) Repeat steps 2 to 5 until the multiplier is zero

a) while (multiplier > 0) do

b) Loop
I) go to step 2
c) end while
vii) Return the product

- Write the assembly code with description (ex. Mov ax,3000h – ax<-3000h)
- Examine & analyze the input/output of assembly code.

III. LAB

Note: For each objective do the following job and assessment:

- Screen shots of the Assembly language program (ALP)

For Obj. 1:

```
;Dinanath Dash
;2241004161

;Find the sum and average of N 16-bit numbers

mov si, 2000h; data stored at si
mov cl, [si]; memory location of si stored at cl
mov ch, 00h; data stored at ch
mov bx, cx; data of cx copied at bx
mov  ax, 0000h; data stored at ax

12:
    inc si; increment si
    inc si; increment si
    add ax, [si]; memory location si stored at ax
    jnc l1; jump if no carry to l1
    inc ch; increment ch

l1:
    dec cl; decrement cl
    jnz l2; jump if no zero to l2
    inc si; increment si
    inc si; increment si
    mov [si], ax; data of ax stored at memory location si
    inc si; increment si
    inc si; increment si
    mov [si], ch; data of ch stored at memory location si
    mov dl, ch; data of ch copied to ch
    div bx; divide bx
    inc si; increment si
    inc si; increment si
    mov [si], ax; data of ax stored at memory location si
    inc si; increment si
    inc si; increment si
    mov [si], dx; data of dx stored at memory location si
hlt; execution halted
```

For Obj. 2:

```
;Dinanath Dash
;2241004161

;Count no. of 0's in an 8-bit number

mov bx, 2000h; value saved at bx
mov al, [bx]; memory location stored at al
mov ch, 00h; value saved at ch
mov cl, 08h; value saved at cl

12:
    shr al, 01h; right shift al to 1 bit
    jc l1; jump if carry to l1
    inc ch; increment ch
l1:
    dec cl; decrement cl
    jnz l2; jump if no zero to l2
    inc bx; increment bx
    mov [bx], ch; value of ch stored in memory location bx
    hlt; execution halted
```

For Obj. 3:

```
;Dinanath Dash
;2241004161

;Move a block of 16-bit data from one location to other

mov si, 1000h; value stored at si
mov di, 1010h; value stored at di
mov cx, 8; value stored at cx

l1:
    mov ax, [si]; value of memory location si stored at ax
    mov di, ax; value of ax stored at di
    add si, 2; adding 2 in si
    add di, 2; adding 2 in di
loop l1; looping l1
```

For Obj. 4:

```
;Dinanath Dash
;2241004161

;Multiplication of two 16-bit numbers without using MUL instruction in direct addressing mode

mov bx, [1000h]; value stored at bx by memory location
mov cx, [1002h]; value stored at cx by memory location
mov ax, 0000h; value stored at ax
mov dx, 0000h; value stored at dx

l1:
    add ax, bx; adding value of bx to ax
    jnc l2; jump if no carry to l2 loop
    inc dx; increment dx

l2:
    dec cx; decrement cx
    jnz l1; jump if no zero to l1
    mov [1004h], ax; value of ax stored at memory location
    mov [1006h], dx; value of dx stored at memory location
hlt; execution halted
```

- Observations (with screen shots)

For Obj. 1:

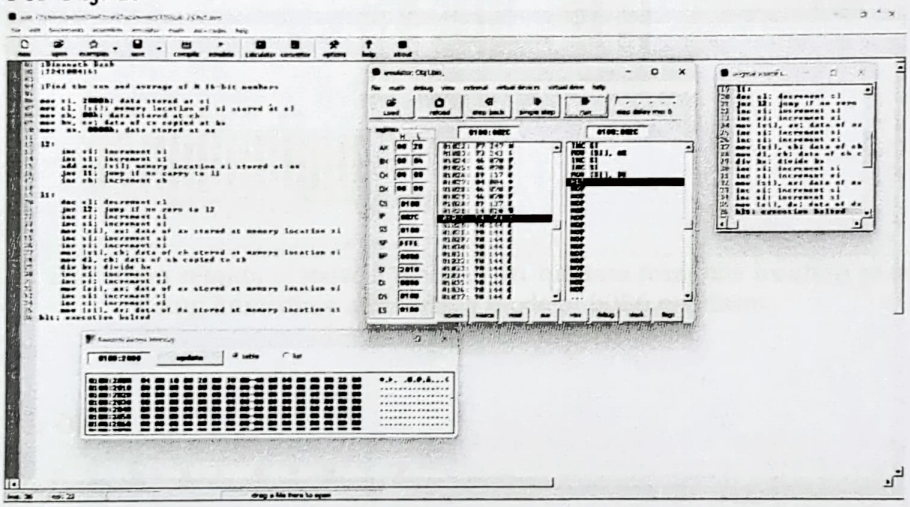

Fig. 1. Execution results of the sum and average of N 16-bit numbers using immediate addressing mode of 8086 emulator.

For Obj. 2:

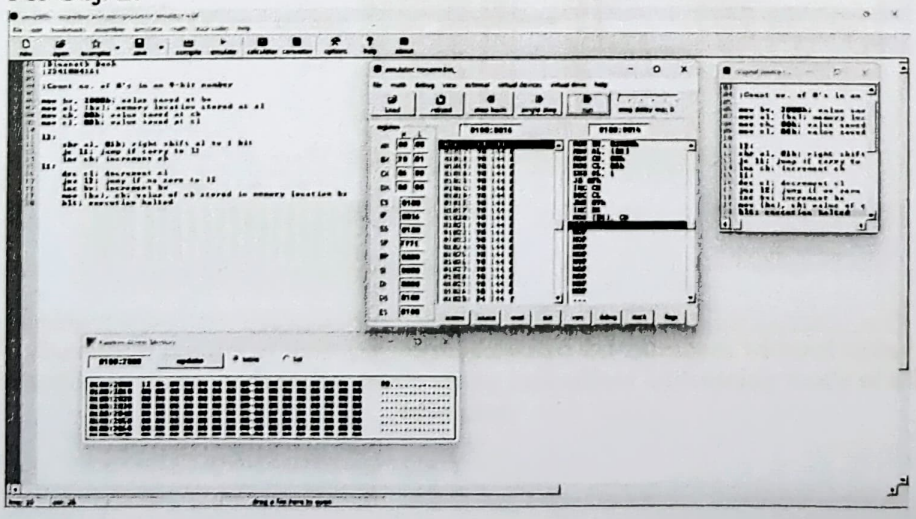

Fig. 2. Execution results of count no. of 0's in an 8-bit number using immediate addressing mode of 8086 emulator.

For Obj. 3:

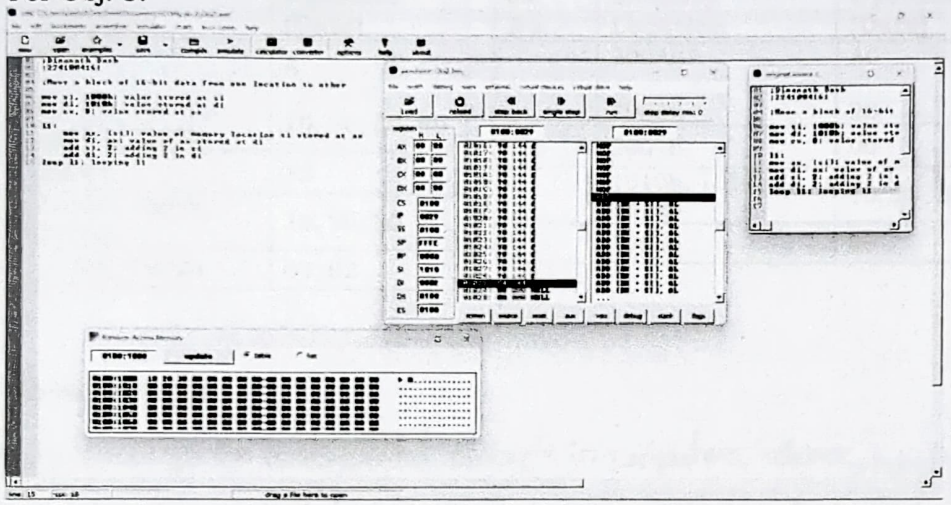

Fig. 3. Execution results of move a block of 16-bit data from one location to other using immediate addressing mode of 8086 emulator.

For Obj. 4:

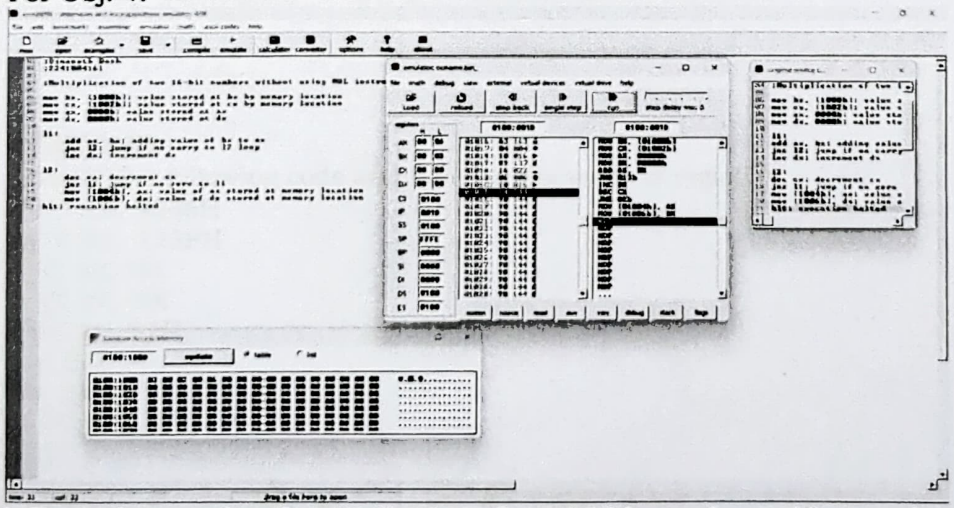

Fig. 4. Execution results of multiplication of two 16-bit numbers without using MUL instruction in direct addressing mode using immediate addressing mode of 8086 emulator.

From this result I have observed.....

Input:

Sl. No.	Memory Location	Operand (Data)
1	2000h 2002h, 2004h, 2006h, 2008h	5 10, 20, 30, 40
2	2000h	12
3	1000h, 1001h, 1003h	10, 20, 30
4	1000h, 1002h	03, 02

Output:

Sl. No.	Memory Location	Operand (Data)
1	200Ah 200Eh	A0 28
2	2001h	06
3	1010h, 1011h, 1012h	10, 20, 30
4	1006h	06

IV. CONCLUSION

These tasks encompass fundamental concepts in computere science, ranging from arithmetic operations and bitwise manipulation to memory management and algorithm design. By mastering these skills, programmers can efficiently handle data, optimise performance and develop robust software solutions across various domains.

V. POST LAB

1. Analyze the following code and find out the value of registers.

```
MOV AX, 4246H
MOV BX, 123FH
AND AX, BX
ADD AX, BX
ROR AX, 02H
INC BX
INC BX
MOV [BX], AX
HLT
```

Ans → i) ax is initialised with hexadecimal value 4246h.

ii) bx is initialised with the hexadecimal value 123Fh.

iii) ax is and with bx (binary and operations), and the result will be stored in ax.

iv) ax is incremented by the value of bx.

v) ax is rotated right 2 times.

vi) bx is incremented twice

vii) ax is finally moved to the memory location pointed by bx.

viii) execution halted.

2. Division of two 16-bit numbers without using DIV instruction in direct addressing mode.

Ans: Assembly Program-

```
;Dinanath Dash
;2241004161

mov bx, [1000h]; data stored at memory location bx
mov cx, [1002h]; data stored at memory location cx
mov dx, 0000h; value stored at dx
mov ax, 0000h; value stored at ax
l1:
    sub bx, cx; subtract cx from bx
    inc ax; increment cx, bx
    cmp cx, bx; compare cx and bx
    ja l2; jump if above l2
    jna l1; jump if not above l1
l2:
    mov dx, ax; value of ax stored in dx
hlt; execution halted
```

Observation-