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: CSIT		Section: 2241030	
S. No.	**Name**	**Registration No.**	**Signature**
9	Abhasha kumar swain	2241019580	Swain

Marks: 10 /10

Remarks:

28/3/24

Teacher's Signature

II. PRE-LAB

Objective-1 (Find the sum and average of N 16-bit numbers)

- **pseudocode**

→ First take the data (the numbers) as input you want to take

→ Then add the numbers one by one. the answer is store in AX (Accumulator).

→ Then divide the sum by the no of inputs. You get the Average.

- **Assembly code**

```
Mov AX, 0000H      AX ← 0000h
Mov DX, 0000H      DX ← 0000h
Mov SI, 5000H      SI ← 5000h
MOV CL, [SI]       CL ← [SI] (value of SI)
INC SI             SI ← SI + 01h

BACK: Mov BX, [SI]   BX ← [SI] (value of SI)
      ADD AX, BX     AX ← AX + BX

      JNC NEXT       Jump to next if no carry.
      INC DX         DX ← DX + 1

NEXT: INC SI         SI ← SI + 1
      INC SI         SI ← SI + 1
      DEC CL         CL ← CL - 1
```

JNZ BACK Jump to back if not zero (CL)

```
INC SI      SI ← SI+1
INC SI      SI ← SI+1
MOV [SI], AX   M[SI] ← AX

INC SI      SI ← SI+1
INC SI      SI ← SI+1
MOV [SI], DX   M[SI] ← DX

MOV BX, 0000H   BX ← 0000h
MOV BL, [5000h]   BL ← 5000h
DIV BX      AX ← AX/BX (Quotient)
INC SI      SI ← SI+1
INC SI      SI ← SI+1
MOV [SI], AX   M[SI] ← AX

INC SI      SI ← SI+1
INC SI      SI ← SI+1
MOV [SI], DX   M[SI] ← DX
```

Analysis File

Figure take input through memory before execute the code.

(ii) Transfer if from memory to BX one by one and Add it to AX.

(iii) At last we get sum of numbers in AX

Eg The numbers are taken

(i) 1234

(ii) 6145

(iii) 5612

Sum is $(1234)_{16} + (6145)_{16} + (5612)_{16} = C98B$

Average = $\frac{C98B}{3}$ = Quotient is → 432E h

Remainder is → 01 h

The store this to the memory location

Objective-2 [count no. of 0's in an 8-bit numbers]

pseudocode:→

(i) Right shift the 8bit number 1 by 1 then check the carry flag.

(ii) If the carry flag is 1 then the bit is 1 so don't increase the count that you store in another register.

(iii) When the carry is not generated the increase the count

(iv) After 8 times right shift (as the 8 bit numbers) we get our desired result.

Assembly code:-

```
Mov BX, 5000h    BX ← 5000h
Mov AL, [BX]     AL ← M[BX]
Mov CL, 00H      CL ← 00h
Mov CH, 08H      CH ← 08h
```

LOOP 2: SHR AL, 01H	Right shift AL by 01
JC LOOP1	Jump if carry to loop1
INC CL	CL ← CL+1
LOOP1: DEC CH	CH ← CH−1
JNZ LOOP2	Jump if not zero(ch)
INC BX	BX ← BX+1
MOV [BX], CL	CL ← M[BX]
HLT	

Analysis

i/p → FY [5000H]

↓

(1111 0100)

number of zero → 3

[5001H] → 03

1111 0100

L L L →

Right Shift all the bits.

Objective-3 (Move a block of 16 bit data from one location to another)

pseudocode

→ Take the data in the memory location that store in SI

→ INC the SI, the Also set the Destination address in DX

→ Mov the data from M[SI] to M[DX] through a general purpose register

Assembly code

```
; Mov AX, 2000H   } commented
; Mov DS
  Mov SI, 3000h      SI ← 3000h
  Mol CL, [SI]       CL ← M[SI]
  INC SI             SI ← SI+1
  Mov DI, 5000h      DI ← 5000h
LOOP1: Mov BX, [SI]  BX ← M[SI]
  Mov [DI], BX       [DI] ← BX
  INC SI             SI ← SI+1
  INC SI             SI ← SI+1
  INC DI             DI ← DI+1
  INC DI             DI ← DI+1
  DEC CL             CL ← CL-1
JNZ LOOP1            Jump if not zero (CL)
  HLT                TO LOOP1
```

Analysis

(i) we store 3000h in SI which is our source address where onwards we give the data.

(ii) we Move the data at M(SI) to register then register to the ~~address~~ destination address.

Input

[5001h] → 12
[5002h] → 34
[5003h] → 56
[5004h] → 78

output through BX

M[3000h] ← BX ← M[5001h]
M[3001h] ← BX ← M[5002h]
M[3002h] ← BX ← M[5003h]
M[3003h] ← BX ← M[5004h]

Objective-4 Multiplication of 16 bit numbers without using MUL instructions.

Pseudocode:→

→ Take 2 data as input on which you want to perform operation
→ Add the 1st data with itself for "2nd data" times.
→ Then the final result is the Multiplication of that two numbers.

Assembly code

```
MOV BX [1000h]   ; BX ← M[1000h]
MOV CX [1002h]     CX ← M[1002h]
MOV DX, 0000h      DX ← 0000h
MOV AX, 0000h      AX ← 0000h

BACK: ADD AX, BX   AX ← AX + BX
      JNC NEXT     Jump to Next if no carry
      INC DX       DX ← DX + 1h

NEXT: DEC CX       CX ← CX - 1h
      JNZ BACK     Jump to back if not zero (CX)
      MOV [0004h], AX   M[0004] ← AX
      MOV [1006h], DX   M[1006] ← DX

      HLT          END
```

Analysis

→ BX contain the first and CX contains the second data.

→ we add AX with BX, CX times.

→ the if in every addition carry generates the carry bit store in DX.

So the Result is

BX ← [1000h] = 5634

CX ← [1002h] = 000A

5634h × 000Ah = 035E08h

From this result I have observed...

INPUT:

Sl. No.	Memory Location	Operand (Data)
1	0100:1000	34h
2	0100:1001	56h
3	0100:1002	0Ah
4	0100:1003	00h

OUTPUT:

Sl. No.	Memory Location	Operand (Data)
1	0100:1004	08h
2	0100:1005	5Eh
3	0100:1006	03h
4	0100:1007	00h

IV. Conclusion:→

→ From the above objective we conclude that can perform various operation by the Assembly code.

→ use loops to perform same operation repeatedly by checking the flags.

→ Using ROR, ROL, RCL, RCR we manipulate the binary bits.

V POST LAB

1.

MOV AX, 4246H	AX ← 4246
MOV BX, 123FH	BX ← 123F
AND AX, BX	AX ← AX & BX
ADD AX, BX	AX ← AX + BX
ROR AX, 02H	
INC BX	BX ← BX+1
INC BX	BX ← BX+1
MOV [BX], AX	M[BX] ← AX
HLT	

(i)

AX ← 0100 0010 0100 0110
BX ← 0001 0010 0011 1111

(0000)0010 0000 0110

02 06

(ii) 0206 + 123F = 1445

0001 0100 0100 0101

(iii) Shift 2 bit

01000101 0001 0001

(iv) BX ← 4511

4 5 1 1

(v) BX ← 1241

(vi) M[1241] ← 4511 (Final result)

2.

```
MOV AX, [1000H]
MOV CX, [1002H]
MOV DX, 0000H
MOV BX, 0000H
NEXT: INC BX
      SUB AX, CX
      JNS NEXT

      DEC BX
      ADD AX, CX
      MOV DX, AX

HLT
```

Division without DIV

28/5/24