Experiment: -2

Name: Vibhuti Garachh	**Roll No. :** A-133
Date of Performance: 10/01/2024	**Date of Submission:** 17/01/2024

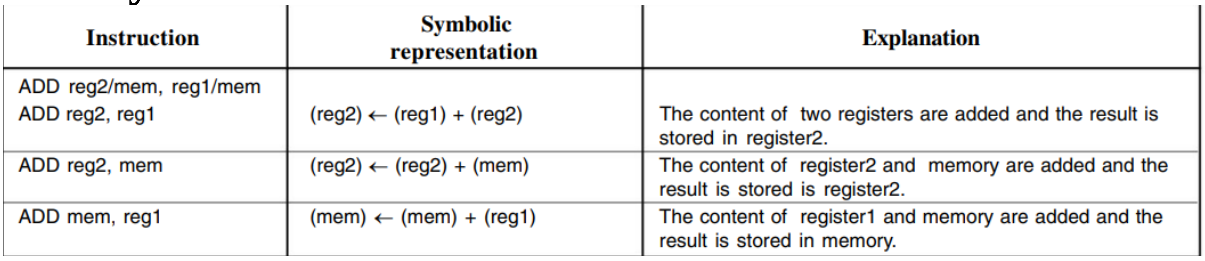
Aim: Study of Arithmetic Instructions - Addition, Division, Increment, Decrement

a. **Write a neatly commented 8086 ALP to add 5 numbers stored at location “numbers” in the data segment. Save the sum, carry count and the average of the numbers in the memory.**

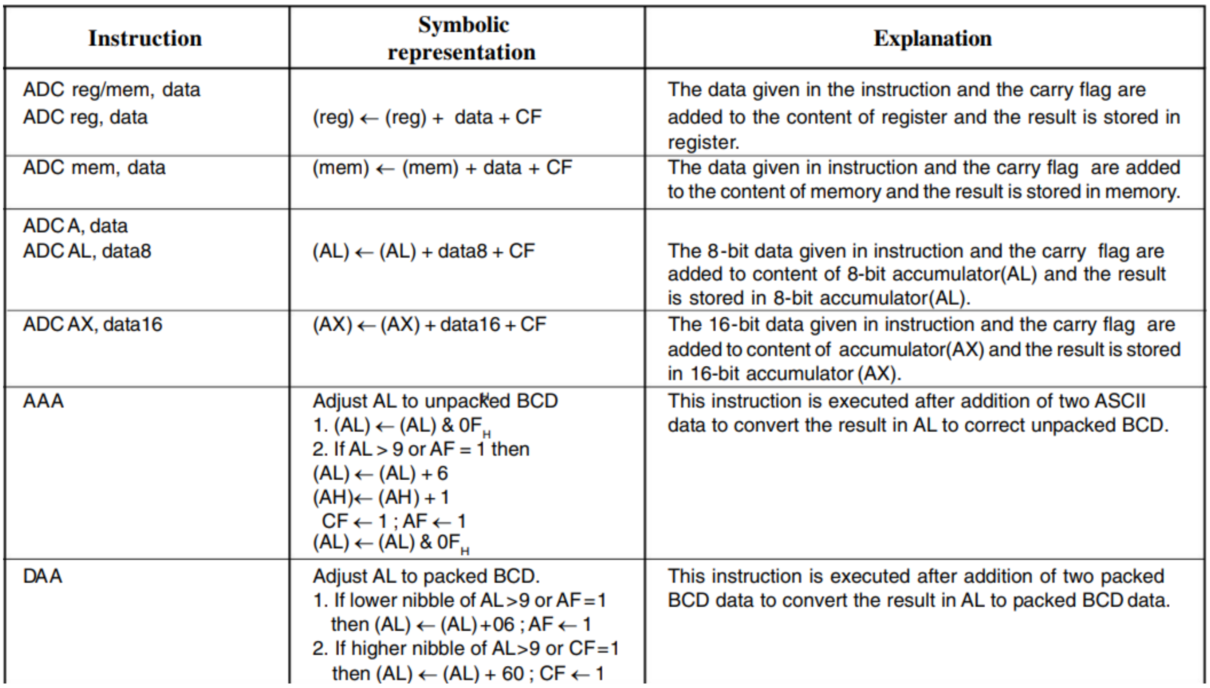
Apparatus Required: Personal Computer.

Software required: - emu8086.

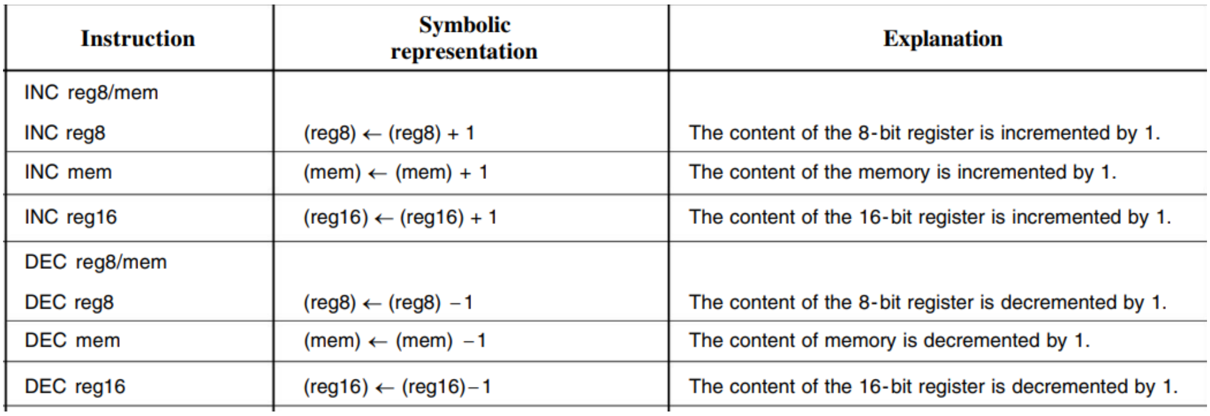
Theory:



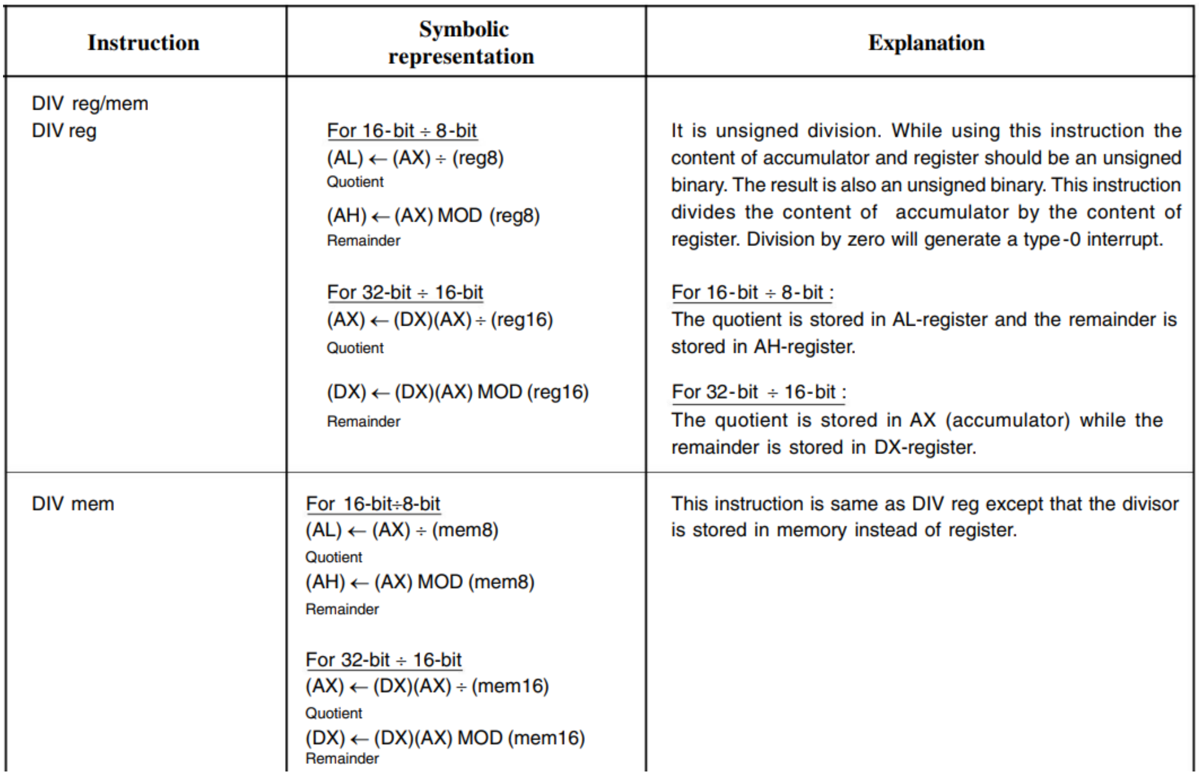
Instruction	Symbolic representation	Explanation
ADD reg2/mem, reg1/mem ADD reg2, reg1	(reg2) ← (reg1) + (reg2)	The content of two registers are added and the result is stored in register2.
ADD reg2, mem	(reg2) ← (reg2) + (mem)	The content of register2 and memory are added and the result is stored is register2.
ADD mem, reg1	(mem) ← (mem) + (reg1)	The content of register1 and memory are added and the result is stored in memory.



Instruction	Symbolic representation	Explanation
ADC reg/mem, data ADC reg, data	(reg) ← (reg) + data + CF	The data given in the instruction and the carry flag are added to the content of register and the result is stored in register.
ADC mem, data	(mem) ← (mem) + data + CF	The data given in instruction and the carry flag are added to the content of memory and the result is stored in memory.
ADC A, data ADC AL, data8	(AL) ← (AL) + data8 + CF	The 8-bit data given in instruction and the carry flag are added to content of 8-bit accumulator(AL) and the result is stored in 8-bit accumulator(AL).
ADC AX, data16	(AX) ← (AX) + data16 + CF	The 16-bit data given in instruction and the carry flag are added to content of accumulator(AX) and the result is stored in 16-bit accumulator (AX).
AAA	Adjust AL to unpacked BCD 1. (AL) ← (AL) & $0F_H$ 2. If AL > 9 or AF = 1 then (AL) ← (AL) + 6 (AH)← (AH) + 1 CF ← 1 ; AF ← 1 (AL) ← (AL) & $0F_H$	This instruction is executed after addition of two ASCII data to convert the result in AL to correct unpacked BCD.
DAA	Adjust AL to packed BCD. 1. If lower nibble of AL>9 or AF=1 then (AL) ← (AL)+06 ; AF ← 1 2. If higher nibble of AL>9 or CF=1 then (AL) ← (AL) + 60 ; CF ← 1	This instruction is executed after addition of two packed BCD data to convert the result in AL to packed BCD data.

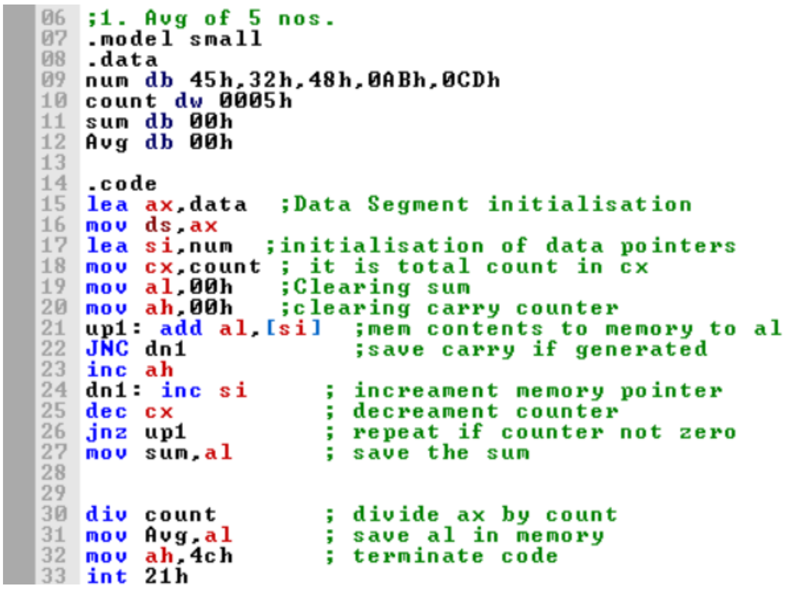


Instruction	Symbolic representation	Explanation
INC reg8/mem INC reg8	(reg8) ← (reg8) + 1	The content of the 8-bit register is incremented by 1.
INC mem	(mem) ← (mem) + 1	The content of the memory is incremented by 1.
INC reg16	(reg16) ← (reg16) + 1	The content of the 16-bit register is incremented by 1.
DEC reg8/mem DEC reg8	(reg8) ← (reg8) −1	The content of the 8-bit register is decremented by 1.
DEC mem	(mem) ← (mem) −1	The content of memory is decremented by 1.
DEC reg16	(reg16) ← (reg16)−1	The content of the 16-bit register is decremented by 1.



Instruction	Symbolic representation	Explanation
DIV reg/mem DIV reg	For 16-bit ÷ 8-bit (AL) ← (AX) ÷ (reg8) Quotient (AH) ← (AX) MOD (reg8) Remainder For 32-bit ÷ 16-bit (AX) ← (DX)(AX) ÷ (reg16) Quotient (DX) ← (DX)(AX) MOD (reg16) Remainder	It is unsigned division. While using this instruction the content of accumulator and register should be an unsigned binary. The result is also an unsigned binary. This instruction divides the content of accumulator by the content of register. Division by zero will generate a type-0 interrupt. For 16-bit ÷ 8-bit : The quotient is stored in AL-register and the remainder is stored in AH-register. For 32-bit ÷ 16-bit : The quotient is stored in AX (accumulator) while the remainder is stored in DX-register.
DIV mem	For 16-bit÷8-bit (AL) ← (AX) ÷ (mem8) Quotient (AH) ← (AX) MOD (mem8) Remainder For 32-bit ÷ 16-bit (AX) ← (DX)(AX) ÷ (mem16) Quotient (DX) ← (DX)(AX) MOD (mem16) Remainder	This instruction is same as DIV reg except that the divisor is stored in memory instead of register.

Code :

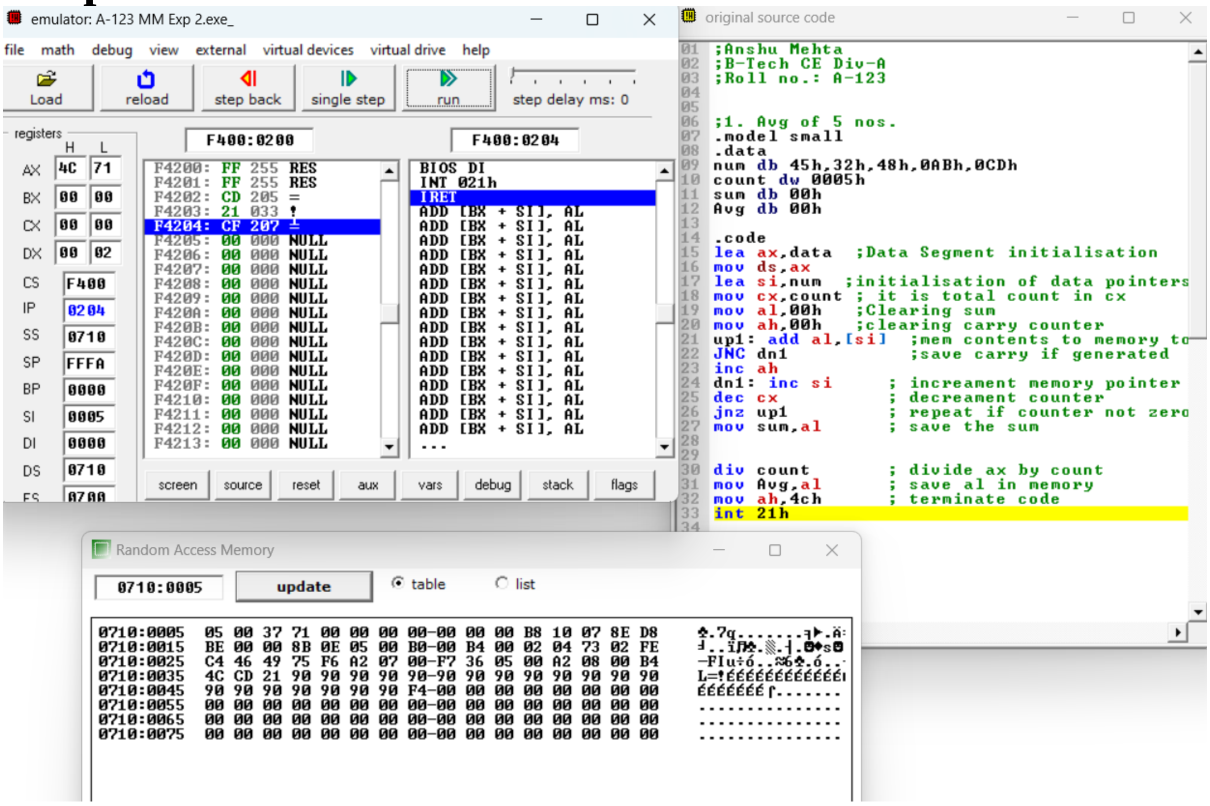


```
;1. Avg of 5 nos.
.model small
.data
num db 45h,32h,48h,0ABh,0CDh
count dw 0005h
sum db 00h
Avg db 00h

.code
lea ax,data  ;Data Segment initialisation
mov ds,ax
lea si,num  ;initialisation of data pointers
mov cx,count ; it is total count in cx
mov al,00h   ;Clearing sum
mov ah,00h   ;clearing carry counter
up1: add al,[si]  ;mem contents to memory to al
JNC dn1           ;save carry if generated
inc ah
dn1: inc si       ; increament memory pointer
dec cx            ; decreament counter
jnz up1           ; repeat if counter not zero
mov sum,al        ; save the sum


div count         ; divide ax by count
mov Avg,al        ; save al in memory
mov ah,4ch        ; terminate code
int 21h
```

Output:



```
     H  L
AX  4C 71
BX  00 00
CX  00 00
DX  00 02
CS  F400
IP  0204
SS  0710
SP  FFFA
BP  0000
SI  0005
DI  0000
DS  0710
ES  0700
```

```
F400:0200
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ±
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
;Anshu Mehta
;B-Tech CE Div-A
;Roll no.: A-123


```

```
;1. Avg of 5 nos.
.model small
.data
num db 45h,32h,48h,0ABh,0CDh
count dw 0005h
sum db 00h
Avg db 00h

.code
lea ax,data  ;Data Segment initialisation
mov ds,ax
lea si,num  ;initialisation of data pointers
mov cx,count ; it is total count in cx
mov al,00h   ;Clearing sum
mov ah,00h   ;clearing carry counter
up1: add al,[si]  ;mem contents to memory to
JNC dn1           ;save carry if generated
inc ah
dn1: inc si       ; increament memory pointer
dec cx            ; decreament counter
jnz up1           ; repeat if counter not zero
mov sum,al        ; save the sum


div count         ; divide ax by count
mov Avg,al        ; save al in memory
mov ah,4ch        ; terminate code
int 21h
```

```
0710:0005
0710:0005  05 00 37 71 00 00 00 00-00 00 00 B8 10 07 8E D8
0710:0015  BE 00 00 8B 0E 05 00 B0-00 B4 00 02 04 73 02 FE
0710:0025  C4 46 49 75 F6 A2 07 00-F7 36 05 00 A2 08 00 B4
0710:0035  4C CD 21 90 90 90 90 90-90 90 90 90 90 90 90 90
0710:0045  90 90 90 90 90 90 90 F4-00 00 00 00 00 00 00 00
0710:0055  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0710:0065  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0710:0075  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```