STEPS FOR PROGRAMMING:-

Changing Z to C drive and directory to 8086

z:\> mount c c:\

z:> c:

c:\> cd 8086

c:\8086>

Editor: This is used to open the editor for entering code

Step1: c:\8086> masm filename.asm

Step2: Linking: This is used to link the asm file to obj file

c:\8086> link filename.obj

Step3: The obj file is executed to generate the executable file

c:\8086> debug filename.exe

-g: This is used for entire program execution

-t: line by line execution

-u: unassemble the code

-q: To exit the debugger

-d<address of segment>:0

Eg: d 076b:0

Basic 8086 Program Structure:

Any 8086 program consists of two parts:

1. Directives (For the assembler)

2. Instruction and Values (For the compiler)

PART-A: EXPERIMENTS ON 8086 MICROPROCESSOR

Experiment No. 01: Programs for 16-bit arithmetic operations using Various Addressing Modes.

A) ADDITION OF TWO 8BIT NNUMBERS:

REGISTERS USED: AL,BL

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code
code segment
start:mov al,02h
mov bl,03h
add al,bl
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=FF05  BX=0003  CX=0007  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0006   NV UP EI PL NZ NA PE NC
076A:0006 CC            INT     3
```

B) SUBTRACTION OF TWO 8BIT NUMBERS:

REGISTERS USED: AL,BL

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code
code segment
start:mov al,05h
mov bl,03h
sub al,bl
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=FF02  BX=0003  CX=0007  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0006   NV UP EI PL NZ NA PO NC
076A:0006 CC            INT     3
```

C) MULTIPLICATION OF TWO 8BIT NUMBERS:

REGISTERS USED: AL,BL

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code
code segment
start:mov al,05h
mov bl,02h
mul bl
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=000A  BX=0002  CX=0007  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0006   NV UP EI PL NZ NA PO NC
076A:0006 CC            INT     3
```

D) DIVISION OF TWO 8BIT NUMBERS:

REGISTERS USED: AL,BL

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code
code segment
start:mov al,9h
mov bl,3h
div bl
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=0003  BX=0003  CX=0009  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 CC            INT     3
```

E)ADDITION OF TWO 16BIT NNUMBERS:

REGISTERS USED:AX,BX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code
code segment
start:mov ax,2345h
mov bx,5623h
add ax,bx
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=7968  BX=5623  CX=0009  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 CC              INT     3
```

F) SUBTRACTION OF TWO 16BIT NUMBERS:

REGISTERS USED:AX,BX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code
code segment
start:mov ax,7896h
mov bx,2345h
sub ax,bx
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=5551  BX=2345  CX=0009  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 CC            INT     3
```

G) MULTIPLICATION OF TWO 16BIT NUMBERS:

REGISTERS USED:AX,BX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code
code segment
start:mov ax,2345h
mov bx,3456h
mul  bx
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=DD2E  BX=3456  CX=0009  DX=0735  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0008   OV UP EI PL NZ NA PO CY
076A:0008 CC            INT     3
```

H) DIVISION OF TWO 16BIT NUMBERS:

REGISTERS USED:AX,BX

S**OFTWARE USED:** MASM.

```
assume cs:code
code segment
start:mov ax,0007h
mov bx,0003h
div bx
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=0002  BX=0003  CX=0009  DX=0001  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 CC              INT     3
```

ADDITION OF NUMBERS USING ADDRESSING MODES:

REGISTERS USED:AX,BX,SI,DI

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data
data segment
input dw 0F232h,4535h
output dw 0000h
data ends
code segment
start:mov ax,data
mov ds,ax
mov si,offset input
mov di,offset output
```

```
mov ax,[si]
add si,0002h
mov bx,[si]
add ax,bx
mov [di],ax
int 03h
code ends
end start
```

OUTPUT:

```
AX=3767  BX=4535  CX=0027  DX=0000  SP=0000  BP=0000  SI=0002  DI=0004
DS=076A  ES=075A  SS=0769  CS=076B  IP=0016   NV UP EI PL NZ NA PO CY
076B:0016 CC            INT     3
```

Experiment No. 02: Program for sorting an array for 8086.

A) SORTING AN ARRAY OF UNSIGNED INTEGERS IN ASCENDING ORDER

REGISTERS USED: AX,DS,SI,CX,DX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data
data segment
arr1 db 53h,25h,19h,02h,29h
count equ 04h
data ends
code segment
start:mov ax,data
mov ds,ax
mov dx,count
back0:mov cx,count
mov si,offset arr1
back1:mov al,[si]
cmp al,[si+1]
jb next
xchg [si+1],al
xchg [si],al
next:inc si
loop back1
dec dx
jnz back0
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=0729  BX=0000  CX=0000  DX=0000  SP=0000  BP=0000  SI=0004  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=0020   NV UP EI PL ZR NA PE CY
076B:0020 CC            INT     3



-d ds:0000
076A:0000  02 19 25 29 53 00 00 00-00 00 00 00 00 00 00 00
076A:0010  B8 6A 07 8E D8 BA 04 00-B9 04 00 BE 00 00 8A 04
076A:0020  3A 44 01 72 05 86 44 01-86 04 46 E2 F1 4A 75 E8
076A:0030  CC E9 11 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6
```

B) SORTING AN ARRAY OF UNSIGNED INTEGERS IN DESCENDING ORDER:

REGISTERS USED:AX,DS,SI,CX,DX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data
data segment
arr1 db 57h,26h,78h,98h,67h
count equ 04h
data ends
code segment
start:mov ax,data
mov ds,ax
mov dx,count
back0:mov cx,count
mov si,offset arr1
back1:mov al,[si]
cmp al,[si+1]
ja next
```

```
xchg [si+1],al
xchg [si],al
next:inc si
loop back1
dec dx
jnz back0
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=0757  BX=0000  CX=0000  DX=0000  SP=0000  BP=0000  SI=0004  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=0020   NV UP EI PL ZR NA PE NC
076B:0020 CC            INT     3


-d ds:0000
076A:0000  98 78 67 57 26 00 00 00-00 00 00 00 00 00 00 00
076A:0010  B8 6A 07 8E D8 BA 04 00-B9 04 00 BE 00 00 8A 04
076A:0020  3A 44 01 77 05 86 44 01-86 04 46 E2 F1 4A 75 E8
076A:0030  CC E9 11 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6
-
```

C) SORTING AN ARRAY USING DATAWORD:

REGISTERS USED: AX,DS,SI,CX,DX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data
data segment
arr1 dw 3242h,8567h,7865h,4536h,9867h
count equ 0004h
```

```
data ends
code segment
start:mov ax,data
mov ds,ax
mov dx,count
back0:mov cx,count
mov si,offset arr1
back1:mov ax,[si]
cmp ax,[si+2]
jb next
xchg [si+2],ax
xchg [si],ax
next:add si,0002h
loop back1
dec  dx
jnz back0
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=8567  BX=0000  CX=0000  DX=0000  SP=0000  BP=0000  SI=0008  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=0022   NV UP EI PL ZR NA PE NC
076B:0022 CC            INT     3


-d ds:0000
076A:0000  42 32 36 45 65 78 67 85-67 98 00 00 00 00 00 00
076A:0010  B8 6A 07 8E D8 BA 04 00-B9 04 00 BE 00 00 8B 04
076A:0020  3B 44 02 72 05 87 44 02-87 04 83 C6 02 E2 EF 4A
076A:0030  75 E6 CC 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6
```

D) SORTING AN ARRAY OF SIGNED INTEGERS IN ASCENDING ORDER:

REGISTERS USED: AX,DS,SI,CX,DX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data
data segment
arr1 db +53h,-25h,-19h,+02h
count equ 04h
data ends
code segment
start:mov ax,data
mov ds,ax
mov dx,count-1
back0:mov cx,dx
mov si,offset arr1
back1:mov al,[si]
cmp al,[si+1]
jl next
xchg [si+1],al
xchg [si],al
next:inc si
loop back1
dec dx
jnz back0
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=07DB  BX=0000  CX=0000  DX=0000  SP=0000  BP=0000  SI=0001  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=001F   NV UP EI PL ZR NA PE CY
076B:001F CC            INT     3
```

```
-d ds:0000
076A:0000  DB E7 02 53 00 00 00 00-00 00 00 00 00 00 00 00
076A:0010  B8 6A 07 8E D8 BA 03 00-8B CA BE 00 00 8A 04 3A
076A:0020  44 01 7C 05 86 44 01 86-04 46 E2 F1 4A 75 E9 CC
076A:0030  03 E9 11 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6
```

E) SORTING AN ARRAY OF SIGNED INTEGERS IN DESCENDING ORDER:

REGISTERS USED: AX,DS,SI,CX,DX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data
data segment
arr db +53h,-25h,-19h,+02h
count equ 04h
data ends
code segment
start:mov ax,data
mov ds,ax
mov dx,count-1
back0:mov cx,dx
mov si,offset arr
back1:mov al,[si]
cmp al,[si+1]
```

```
jg next
xchg [si+1],al
xchg [si],al
next:inc si
loop back1
dec dx
jnz back0
int 03h
code ends
end start
```

OUTPUT:

```
-g

AX=0753  BX=0000  CX=0000  DX=0000  SP=0000  BP=0000  SI=0001  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=001F   NV UP EI PL ZR NA PE NC
076B:001F CC              INT     3
```

```
-d ds:0000
076A:0000  53 02 E7 DB 00 00 00 00-00 00 00 00 00 00 00 00
076A:0010  B8 6A 07 8E D8 BA 03 00-8B CA BE 00 00 8A 04 3A
076A:0020  44 01 7F 05 86 44 01 86-04 46 E2 F1 4A 75 E9 CC
076A:0030  03 E9 11 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6
```

Experiment No. 03: Program for searching for a number or character in a string for 8086

A) PROGRAM TO FIND WHETHER A CHARACTER IS PRESENT IN STRING OR NOT

REGISTERS USED:AX,DS,SI,CX,DX

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data
data segment
str1 db "VNRVJIET$"
str2 db "character is found$"
str3 db "character is not found$"
data ends
code segment
start:mov ax,data
mov ds,ax
mov si,offset str1
mov cx,08h
mov al,"J"
back:cmp al,[si]
je last1
mov dx,offset str2
mov ah,09h
int 21h
jmp last2
last1:mov dx,offset str3
```

```
mov ah,09h

int 21h

last2:int 3h

code ends

end start
```

OUTPUT:

```
-g
character is found
AX=094A  BX=0000  CX=0008  DX=0009  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=076E  IP=0022   NV UP EI NG NZ NA PO CY
076E:0022 CC            INT     3
```

Experiment No. 04: Program for string manipulations for 8086

A) PROGRAM TO MOVE A SOURCE STRING TO DESTINATION

REGISTERS USED: AX,DS,SI,CX,ES,DI

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data,es:extra
data segment
str1 db "VNRVJIETECE$"
len1 dw($-str1)
data ends
extra segment
str db 10 dup(?)
extra ends
code segment
start:mov ax,data
mov ds,ax
mov ax,extra
mov es,ax
cld
mov si,offset str1
mov di,offset str
mov cx,len1
dec cx
rep movsb
int 3h
code ends
end start
```

OUTPUT:

```
-g

AX=076B  BX=0000  CX=0000  DX=0000  SP=0000  BP=0000  SI=000B  DI=000B
DS=076A  ES=076B  SS=0769  CS=076C  IP=0018   NV UP EI PL NZ NA PO NC
076C:0018 CC            INT     3


-d ds:0000
076A:0000  56 4E 52 56 4A 49 45 54-45 43 45 24 0C 00 00 00   VNRVJIETECE$....
076A:0010  56 4E 52 56 4A 49 45 54-45 43 45 00 00 00 00 00   VNRVJIETECE.....
076A:0020  B8 6A 07 8E D8 B8 6B 07-8E C0 FC BE 00 00 BF 00   .j....k.........
076A:0030  00 8B 0E 0C 00 49 F3 A4-CC 46 FC 8B 56 FE 05 0C   .....I...F..V...
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04   .RP..H...P.{....
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A   =..t.....^.&.G.*
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83   .@P.......RP..H.
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6   ..P....P..s.....
```

B) PROGRAM TO CHECK WHETHER A CHARACTER IS PRESENT IN STRING OR NOT

REGISTERS USED:AX,DS,ES,SI,CX,DI

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data,es:extra
data segment
str2 db "character is available$"
str3 db "character is not available$"
data ends
extra segment
str1 db "VNRVJIET$"
extra ends
code segment
start:mov ax,data
mov ds,ax
mov ax,extra
mov es,ax
```

```
mov di,offset str1
mov al,'E'
mov cx,0008h
CLD
REPNE SCASB
JZ next1
next2:mov dx,offset str3
mov ah,09h
int 21h
jmp next3
next1:mov dx,offset str2
mov ah,09h
int 21h
next3:int 03h
code ends
end start
```

OUTPUT:

```
-g
character is available
AX=0945  BX=0000  CX=0001  DX=0000  SP=0000  BP=0000  SI=0000  DI=0007
DS=076A  ES=076E  SS=0769  CS=076F  IP=0028   NV UP EI PL ZR NA PE NC
076F:0028 CC            INT     3
```

C) PROGRAM TO CONCATENATE TWO STRINGS

REGISTERS USED: AX,DS,ES,SI,CX,DI

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data,es:extra
data segment
str1 db "VNRVJIET$"
len1 dw($-str1)
str2 db "ELECTRONICS$"
len2 dw($-str2)
data ends
extra segment
str db 15 dup(?)
extra ends
code segment
start:mov ax,data
mov ds,ax
mov ax,extra
mov es,ax
cld
mov si,offset str1
mov di,offset str
mov cx,len1
dec cx
rep movsb
mov si,offset str2
mov cx,len2
rep movsb
int 3h
code ends
```

end start

OUTPUT:

```
-g

AX=076C  BX=0000  CX=0000  DX=0000  SP=0000  BP=0000  SI=0017  DI=0014
DS=076A  ES=076C  SS=0769  CS=076D  IP=0021   NV UP EI PL NZ NA PO NC
076D:0021 CC            INT     3
```

```
-d ds:0000
076A:0000  56 4E 52 56 4A 49 45 54-24 09 00 45 4C 45 43 54   VNRVJIET$..ELECT
076A:0010  52 4F 4E 49 43 53 24 0C-00 00 00 00 00 00 00 00   RONICS$.........
076A:0020  56 4E 52 56 4A 49 45 54-45 4C 45 43 54 52 4F 4E   VNRVJIETELECTRON
076A:0030  49 43 53 24 D8 B8 6C 07-8E C0 FC BE 00 00 BF 00   ICS$..l.........
```

D) PROGRAM TO REVERSE A STRING.

REGISTERS USED:AX,DS,SI,CX,DI,ES

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data,es:extra
data segment
str1 db "VNRVJIET$"
len1 dw($-str1)
data ends
extra segment
str db 00h
extra ends
code segment
start:mov ax,data
mov ds,ax
mov ax,extra
mov es,ax
```

```
mov si,offset str1
mov di,offset str
mov cx,len1
back:std
lod sb
cld
sto sb
loop back
int 3h
code ends
end start
```

OUTPUT:

```
-g

AX=0776  BX=0000  CX=0000  DX=0000  SP=0000  BP=0000  SI=FFFF  DI=0008
DS=076A  ES=076B  SS=0769  CS=076C  IP=001C   NV UP EI PL NZ NA PO NC
076C:001C CC              INT     3
-d ds:0000
076A:0000  76 6E 72 76 6A 69 65 74-24 00 00 00 00 00 00 00   vnrvjiet$.......
076A:0010  74 65 69 6A 76 72 6E 76-00 00 00 00 00 00 00 00   teijvrnv........
076A:0020  B8 6A 07 8E D8 B8 6B 07-8E C0 BE 00 00 83 C6 07   .j....k.........
076A:0030  BF 00 00 B9 08 00 FD AC-FC AA E2 FA CC FE 05 0C   ................
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04   .RP..H...P.{....
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A   =..t.....^.&.G.*
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83   .@P.......RP..H.
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6   ..P....P..s.....
```

E) PROGRAM TO COMPARE TWO STRINGS.

REGISTERS USED: AX,DS,SI,CX,DI,ES

S**OFTWARE USED:** MASM.

PROGRAM:

```
assume cs:code,ds:data
data segment
str3 db 'strings are equal$'
str4 db 'strings are not equal$'
str1 db 'VNRVJIET$'
data ends
extra segment
str2 db 'VNRVJEET$'
extra ends
code segment
start:mov ax,data
mov ds,ax
mov ax,extra
mov es,ax
mov di,offset str2
mov si,offset str1
mov  cx,0008h
cld
repe cmpsb
jz next1
next2:mov dx,offset str4
mov ah,09h
int 21h
jmp next3
next1:mov dx,offset str3
mov ah,09h
```

int 21h

next3:int 3h

code ends

end start

OUTPUT:

```
-g
strings are not equal
AX=096E  BX=0000  CX=0002  DX=0012  SP=0000  BP=0000  SI=002E  DI=0006
DS=076A  ES=076E  SS=0769  CS=076F  IP=0029   NV UP EI PL NZ NA PO NC
076F:0029 CC            INT     3
```

Experiment No. 05: Program to define and call a subroutine which calculates the average of three numbers

REGISTERS USED: AX,DS,SI,BL,DI

SOFTWARE USED: MASM.

PROGRAM

```
assume cs:code,ds:data

data segment

arr1 db 05h,06h,07h

avg db 03h

data ends

code segment

start:mov ax,data

mov ds,ax

mov si,offset arr1

mov di,offset avg

call average

int  03h

average:mov cx,02h

xor ax,ax

back:mov al,[si]

add  [si+1],al

jnc next

inc ah

next:inc si

loop back

mov al,[si]
```

```
mov bl,[di]

div bl

ret

code ends
```

OUTPUT:

```
-g

AX=0006  BX=0003  CX=0000  DX=0000  SP=0000  BP=0000  SI=0002  DI=0003
DS=076A  ES=075A  SS=0769  CS=076B  IP=000E   NV UP EI PL NZ NA PO NC
076B:000E CC              INT     3
```

INTERFACING MICROPROCESSOR WITH ADC, DAC, STEPPER MOTOR:

STEPS FOR EXCEUTION:

C:\masm 123>masm filename

C:\masm 123>link filename

C:\masm 123>promview

PROMVIEW SOFTWARE:

Select:

1) **File operations :** Select Format

Press 0-Binary (EXE)

2) **File operations:** Read

Filename.exe:

Default 0000

3) **RAM operation:** View

Starting address:starting address+200

:4200

Ending address :4300

Press ESC key

4) **File operation:** Select format

Press 1:intel hex

5) **File operation:** Write

RAM starting address:4200

RAM ending address:4230

Enter file name:ADC.Hex

Exit: Back to DOS

Are you sure(Y/N):Y

Are you saving file(Y/N):Y

C:\masm 123>edit filename. Hex

Post prom view steps:

C:\masm 123>tc serial 1

Press ‘S’ in kit keyword

UIK-86

Type-‘L’ OFFSET=0

Enter filename: adc.hex

Running the program:

The program can be run using two commands.

i)D4200: Unassembles and displays line wise code.

ii)G4200: It is used to run the program.

ADC INTERFACING:

In interfacing an ADC to the 8086 microprocessor in lab,we have a fixed set of port address.

PA: FF50H PC: FF54H

PB: FF52H CONTROL WORD REGISTER: FF56H

PORTA-Input port

PORTC Upper: SOC Signal

INTERFACING ADC WITH 8086:

REGISTERS USED:AL,DX,CX

S**OFTWARE USED:** MASM, promview

PROGRAM:

```
code segment
assume cs:code
org 2000h
mov dx,0ff56h
mov al,90h
```

```
out dx,al
mov dx,0ff54h
mov al,0ffh
out dx,al
mov al,00h
out dx,al
mov al,0ffh
out dx,al
call delay
call delay
mov dx,0ff50h
in al,dx
int 3h
delay:mov cx,0ffffh
loop1:nop
nop
dec cx
jnz loop1
ret
code ends
end
```

OUTPUT:

ANALOG I/P	DIGITAL O/P
1.2	42
1.7	5C
2.1	71
2.5	87

INTERFACING DAC WITH 8086:

i)PROGRAM TO GENERATE A SQUARE WAVEFORM:

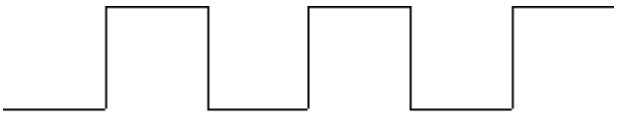
REGISTERS USED:AL,DX,CX

S**OFTWARE USED:** MASM,promview

PROGRAM:

```
code segment
assume cs:code
org 2000h
mov dx,0ff56h
mov  al,80h
out dx,al
mov dx,0ff52h
next:mov al,0ffh
        out dx,al
        call delay
        mov al,00h
        out dx,al
        call delay
        jmp next
delay:mov cx,07ffh
x:nop
   nop
   dec cx
   jnz x
ret
code ends
end start
```

OUTPUT:



Square wave

ii) PROGRAM TO GENERATE A RAMP WAVEFORM:

REGISTERS USED: AL,DX,CX

S**OFTWARE USED:** MASM,promview

PROGRAM:

```
code segment
assume cs:code
org 2000h
mov dx,0ff56h
mov  al,80h
out dx,al
mov dx,0ff52h
mov al,00h
next:out dx,al
        inc al
        jmp next
int 3h
code ends
end start
```

OUTPUT:

INTERFACING STEPPER MOTOR WITH 8086

REGISTERS USED:AL,DX,CX

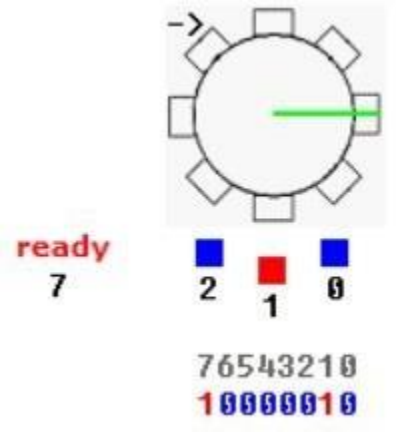
S**OFTWARE USED:** MASM,promview

PROGRAM:

```
mov dx,ff26h
mov al,80h
out dx,al
Step:mov dx,0ff20h
     mov al,80h
     out dx,al
     call delay
     mov al,A0
     out dx,al
     call delay
     mov al,E0
     out dx,al
     call delay
     mov al,C0
     out dx,al
     call delay
     jmp step
delay:mov bx,0010
last:mov al,ff
last1:nop
     nop
     nop
     dec al
     jnz last1
     dec bx
```

```
    jnz last
ret
```

OUTPUT:



PART B: EXPERIMENTS ON ARM DEVELOPMENT BOARDS:

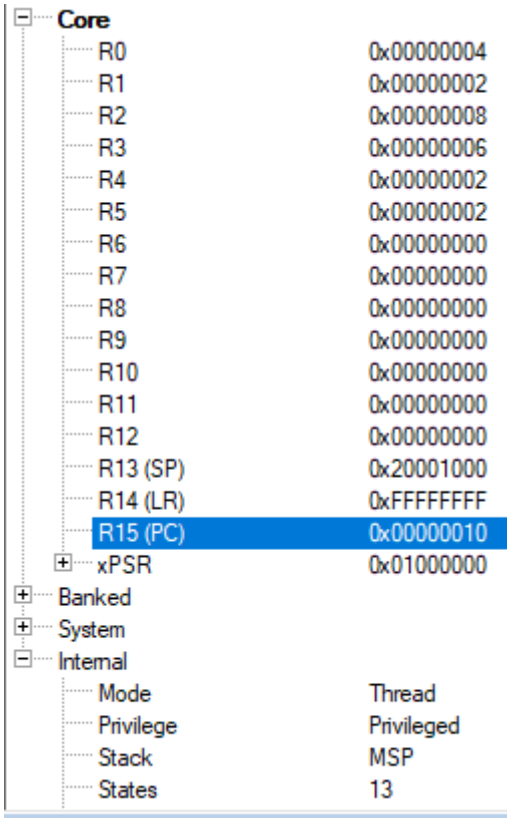
Experiment No. 01: Programs to perform arithmetic operations

PROGRAM:

```
;;; Directives
STACK_ADDR_123G EQU 0x20008000
        PRESERVE8
        THUMB
 ; Vector Table Mapped to Address 0 at Reset
; Linker requires __Vectors to be exported
       AREA    RESET, DATA, READONLY
        EXPORT  __Vectors
__Vectors
DCD  0x20001000  ; stack pointer value when stack is empty
                        ;The processor uses a full descending stack.
                        ;This means the stack pointer holds the address of the last
                        ;stacked item in memory. When the processor pushes a new item
                        ;onto the stack, it decrements the stack pointer and then
                        ;writes the item to the new memory location.

        DCD  Reset_Handler  ; reset vector
 ALIGN
; The program
; Linker requires Reset_Handler
        AREA    MYCODE, CODE, READONLY
        ENTRY
```

```
        EXPORT Reset_Handler

Reset_Handler

;;;;;;;;;User Code Starts from the next line;;;;;;;;;;;;

  MOV R0, #4

   MOV R1, #2

START

        ADD R3, R1, R0

        MUL R2,R1,R0

        SUB R4,R0,R1

        UDIV R5,R0,R1

        B  START

  END; End of the program
```

OUTPUT:



Experiment No. 02: Control ON/OFF of LEDs using switches involving delays.

PROGRAM:

```
#define SYSCTL_RCGCGPIO_R (*(( volatile unsigned long *)0x400FE608 ) )
#define GPIO_PORTF_DATA_R (*(( volatile unsigned long *)0x40025038 ) )
#define GPIO_PORTF_DIR_R (*(( volatile unsigned long *)0x40025400 ) )
#define GPIO_PORTF_DEN_R  (*(( volatile unsigned long *)0x4002551C ) )

#define  GPIO_PORTF_CLK_EN  0x20
#define  GPIO_PORTF_PIN1_EN  0x02
#define LED_ON1  0x02
#define GPIO_PORTF_PIN2_EN 0x04
#define LED_ON2                0x04
#define GPIO_PORTF_PIN3_EN 0x08
#define LED_ON3      0x08

#define DELAY_VALUE     1000000
unsigned long j=0;
void Delay(void){
        for (j=0; j<DELAY_VALUE ; j++);
}
int main ( void )
{
volatile unsigned long ulLoop ;
SYSCTL_RCGCGPIO_R |= GPIO_PORTF_CLK_EN ;
ulLoop = SYSCTL_RCGCGPIO_R;
GPIO_PORTF_DIR_R |= GPIO_PORTF_PIN1_EN ;
```

```
GPIO_PORTF_DEN_R |= GPIO_PORTF_PIN1_EN ;
GPIO_PORTF_DIR_R |= GPIO_PORTF_PIN2_EN ;
GPIO_PORTF_DEN_R |= GPIO_PORTF_PIN2_EN ;
GPIO_PORTF_DIR_R |= GPIO_PORTF_PIN3_EN ;
GPIO_PORTF_DEN_R |= GPIO_PORTF_PIN3_EN ;
while (1)
        {
                GPIO_PORTF_DATA_R &= LED_ON3;
                GPIO_PORTF_DATA_R &= LED_ON2;
                GPIO_PORTF_DATA_R |= LED_ON1;
                Delay();
                GPIO_PORTF_DATA_R &= LED_ON1;
                GPIO_PORTF_DATA_R &= LED_ON2;
                GPIO_PORTF_DATA_R |= LED_ON3;
                Delay();
                GPIO_PORTF_DATA_R &= LED_ON3;
                GPIO_PORTF_DATA_R &= LED_ON1;
                GPIO_PORTF_DATA_R |= LED_ON2;
                Delay();
                }
}
```

OUTPUT:

Experiment No. 03: Controlling an LED using switch by polling method/Interrupt method.

i) INTERRUPT METHOD:

PROGRAM:

```
#include "TM4C123.h"
int main(void)
{
SYSCTL->RCGCGPIO |= (1<<5);
   GPIOF->LOCK = 0x4C4F434B;
   GPIOF->CR = 0x01;
   GPIOF->LOCK = 0;
   GPIOF->DIR &= ~(1<<4)|~(1<<0);
   GPIOF->DIR |= (1<<3);
   GPIOF->DEN |= (1<<4)|(1<<3)|(1<<0);
   GPIOF->PUR |= (1<<4)|(1<<0);
   GPIOF->IS &= ~(1<<4)|~(1<<0);
   GPIOF->IBE &=~(1<<4)|~(1<<0);
   GPIOF->IEV &= ~(1<<4)|~(1<<0);
   GPIOF->ICR |= (1<<4)|(1<<0);
   GPIOF->IM  |= (1<<4)|(1<<0);

   NVIC->IP[30] = 3 << 5;
   NVIC->ISER[0] |= (1<<30);

   while(1)
   {
                        // do nothing and wait for the interrupt to occur
   }
}

void GPIOF_Handler(void)
{
  if (GPIOF->MIS & 0x10)
    {
      GPIOF->DATA |= (1<<3);
```

```
    GPIOF->ICR |= 0x10;
   }
  else if (GPIOF->MIS & 0x01
  {
   GPIOF->DATA &= ~0x08;
   GPIOF->ICR |= 0x01;
*/
  }
}
```

ii) POLLING METHOD:

PROGRAM:

```
#include "TM4C123GH6PM.h"
int main(void)
{
   unsigned int state;
  SYSCTL->RCGCGPIO |= 0x20;
  GPIOF->LOCK = 0x4C4F434B;
  GPIOF->CR = 0x01
  GPIOF->PUR |= 0x10;
  GPIOF->DIR  |=  0x02;
  GPIOF->DEN |= 0x12;
   while(1)
   {
      state = GPIOF->DATA & 0x10;
      GPIOF->DATA = (~state>>3);
   }
}
```

OUTPUT:

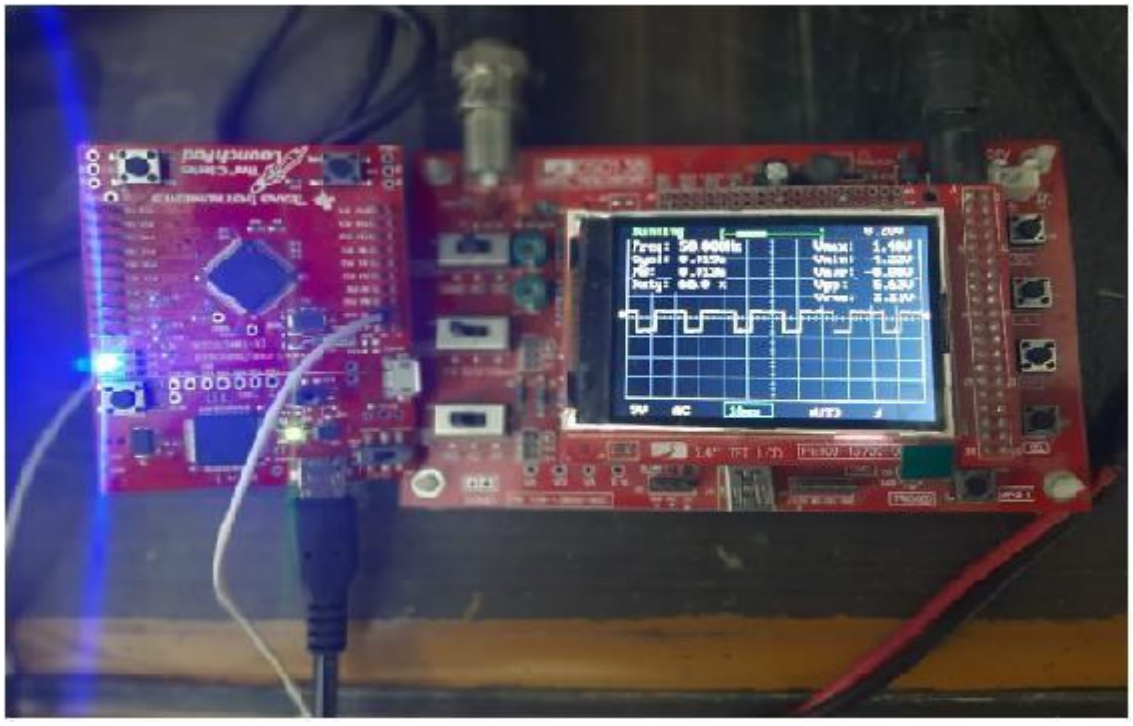
Experiment No. 04 : Implementation of PWM to change duty cycle.

PROGRAM:

```
#include "TM4C123GH6PM.h"

int main(void)

{

void Delay_ms(int n);

int duty_cycle = 4999;

SYSCTL->RCGCPWM |= 2;

SYSCTL->RCGCGPIO |= 0x20;

SYSCTL->RCC |= (1<<20);

SYSCTL->RCC |= 0x000E0000;

GPIOF->AFSEL |= (1<<2);

GPIOF->PCTL &= ~0x00000F00;

GPIOF->PCTL |= 0x00000500;

GPIOF->DEN |= (1<<2);

PWM1->_3_CTL &= ~(1<<0);

PWM1->_3_CTL &= ~(1<<1);

PWM1->_3_GENA = 0x0000008C;

PWM1->_3_LOAD = 5000;

PWM1->_3_CMPA = 4999;

PWM1->_3_CTL = 1;

PWM1->ENABLE = 0x40;

   while(1)

   {

      duty_cycle = duty_cycle - 10;

      if (duty_cycle <= 0)
```

```
            duty_cycle = 5000;
        PWM1->_3_CMPA = duty_cycle;
        Delay_ms(100);
    }
}
void Delay_ms(int time_ms){
    int i, j;
    for(i = 0 ; i < time_ms; i++)
        for(j = 0; j < 3180; j++)
            {}
}
void SystemInit(void)
{
    SCB->CPACR |= 0x00f00000;
}
```

OUTPUT:



Experiment No. 05: Communication through UART

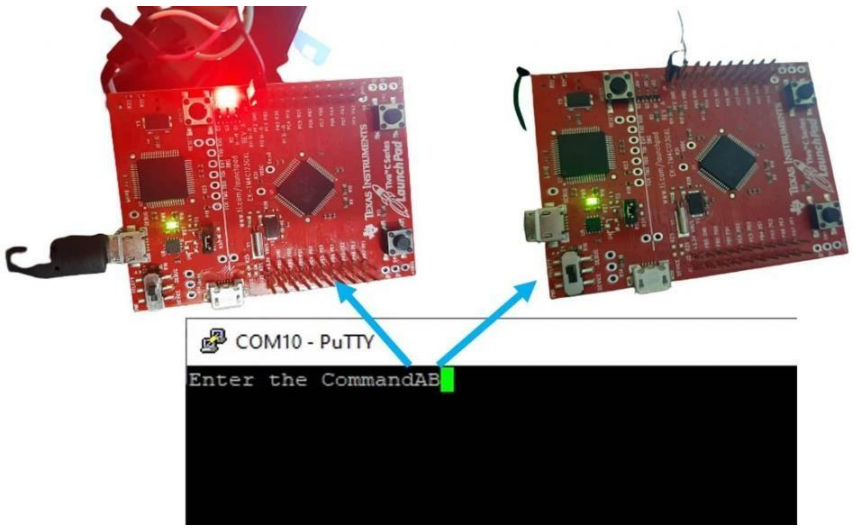
UART TRANSMITTER

PROGRAM:

```
#include "TM4C123.h"
void Delay(unsigned long counter);
char UART5_Receiver(void);
void UART5_Transmitter(char data);
int main(void)
{
  SYSCTL->RCGCUART |= 0x20;
   SYSCTL->RCGCGPIO |= 0x10;
   Delay(1);
   UART5->CTL = 0;
   UART5->IBRD = 104;
   UART5->FBRD = 11;
   UART5->CC = 0;
   UART5->LCRH = 0x60;
   UART5->CTL = 0x301;
   GPIOE->DEN = 0x30;
   GPIOE->AFSEL = 0x30;
   GPIOE->AMSEL = 0;
   GPIOE->PCTL = 0x00110000;
         Delay(1);

       while(1)
       {
             UART5_Transmitter('H');
          UART5_Transmitter('E');
          UART5_Transmitter('L');
          UART5_Transmitter('L');
```

```
            UART5_Transmitter('O');
            UART5_Transmitter('\n');
        }
}
void UART5_Transmitter(char data)
{
   while((UART5->FR & 0x20) != 0);
   UART5->DR = data;
}

void Delay(unsigned long counter)
{
        unsigned long i = 0;
        for(i=0; i< counter; i++);
}
```

OUTPUT:



UART RECEIVER

PROGRAM:

```
#include "TM4C123.h"
#include <stdint.h>
#include <stdlib.h>
void Delay(unsigned long counter);
char UART5_Receiver(void);
void UART5_Transmitter(unsigned char data);
void printstring(char *str);
int main(void)
{
   SYSCTL->RCGCUART |= 0x20;
   SYSCTL->RCGCGPIO |= 0x10;
   Delay(1);
   UART5->CTL = 0;
   UART5->IBRD = 104;
   UART5->FBRD = 11;
   UART5->CC = 0;
   UART5->LCRH = 0x60;
   UART5->CTL = 0x301;
   GPIOE->DEN = 0x30;
   GPIOE->AFSEL = 0x30;
   GPIOE->AMSEL = 0;
   GPIOE->PCTL = 0x00110000;
         Delay(1);
       printstring("Hello World \n");
       Delay(10);
       while(1)
       {
             char c = UART5_Receiver();
             UART5_Transmitter(c);
```

```
        }
}
char UART5_Receiver(void)
{
   char data;
         while((UART5->FR & (1<<4)) != 0);
   data = UART5->DR ;
   return (unsigned char) data;
}
void UART5_Transmitter(unsigned char data)
{
   while((UART5->FR & (1<<5)) != 0);
   UART5->DR = data;
}
void printstring(char *str)
{
  while(*str)
        {
              UART5_Transmitter(*(str++));
        }
}

void Delay(unsigned long counter)
{
        unsigned long i = 0;

        for(i=0; i< counter; i++);
}
void SystemInit(void)
{
   __disable_irq();
   SCB->CPACR |= 0x00F00000;
}
```

OUTPUT: