MASM EXPERIMENTS

EXPERIMENT 8

8 bit addition and multiplication using MASM

AIM
To implement 8 bit addition and multiplication

INPUT
numbers to add

OUTPUT
sum and product

ADDITION

ALGORITHM

1) Load data segment starting address to DS
2) Get the first number and store it in AL
3) Move the value to BL
4) Get the second number and store it in AL
5) ADD BL and AL and store it in AL
6) Correct the Value and convert it into BCD using AAA
7) Move value in AX to BX
8) Print the value
9) Stop

PROGRAM

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
   M1 DB 10,13,"Enter first number:$"
   M2 DB 10,13,"Enter second number:$"
   M3 DB 10,13,"Sum: $"
DATA ENDS
PRTMSG MACRO MESSAGE
   LEA DX, MESSAGE
   MOV AH,09
   INT 21H
   ENDM
GETDCM MACRO
   MOV AH, 01
   INT  21H
   SUB AL, 30H
   ENDM
CODE  SEGMENT
   START: MOV AX, DATA
      MOV DS, AX
      PRTMSG M1
      GETDCM
      MOV BL, AL
      PRTMSG M2
      GETDCM
      ADD AL, BL
      MOV AH, 00H
      AAA
      MOV BX, AX
      PRTMSG M3
      MOV DL, BH
      ADD DL, 30H
      MOV AH, 02
      INT  21H
      MOV DL, BL
      ADD Dl, 30H
      INT  21H
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

<u>MULTIPICATION</u>

<u>ALGORITHM</u>

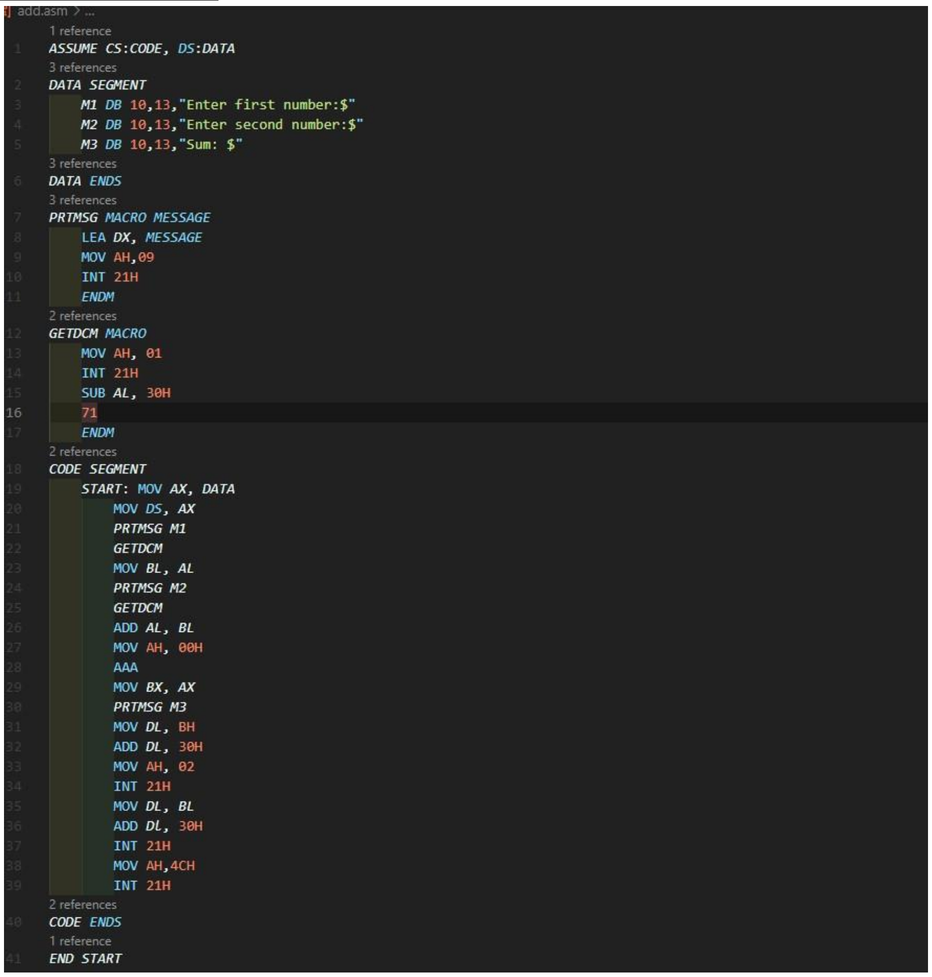
1) Load data segment starting address to DS
2) Get the first number and store it in AL
3) Move the value to BL
4) Get the second number and store it in AL
5) MUL BL and AL and store it in AX
6) Correct the Value and convert it into BCD using AAM
7) Move value in AX to BX
8) Print the value
9) Stop

PROGRAM

```
DATA SEGMENT
M1 DB 13,10,"ENTER 2 NUMBERS $"
M2 DB 13,10,"PRODUCT IS $"
DATA ENDS


PRTMSG MACRO MESSAGE
    LEA DX, MESSAGE
    MOV AH,09
    INT 21H
    ENDM
GETDCM MACRO
    MOV AH, 01
    INT 21H
    SUB AL, 30H
    ENDM

CODE SEGMENT
ASSUME CS:CODE , DS:DATA
 START:MOV AX,DATA
        MOV DS,AX
        PRTMSG M1
        GETDCM
        MOV BL,AL
        GETDCM
        MOV AH,00H
        MUL BL
        AAM
        MOV BX,AX
                        PRTMSG M2
        MOV DL,BH
        OR DL,30H
        MOV AH,02H
        INT  21H
        MOV DL,BL
        OR DL,30H
        INT  21H
        MOV AH,4CH
        INT 21H

CODE ENDS
END START
```

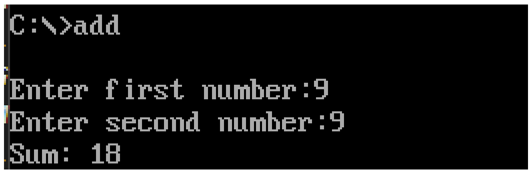
SAMPLE CODE AND OUTPUT

ADDITION



```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
    M1 DB 10,13,"Enter first number:$"
    M2 DB 10,13,"Enter second number:$"
    M3 DB 10,13,"Sum: $"
DATA ENDS
PRTMSG MACRO MESSAGE
    LEA DX, MESSAGE
    MOV AH,09
    INT 21H
    ENDM
GETDCM MACRO
    MOV AH, 01
    INT 21H
    SUB AL, 30H
    71
    ENDM
CODE SEGMENT
    START: MOV AX, DATA
        MOV DS, AX
        PRTMSG M1
        GETDCM
        MOV BL, AL
        PRTMSG M2
        GETDCM
        ADD AL, BL
        MOV AH, 00H
        AAA
        MOV BX, AX
        PRTMSG M3
        MOV DL, BH
        ADD DL, 30H
        MOV AH, 02
        INT 21H
        MOV DL, BL
        ADD DL, 30H
        INT 21H
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

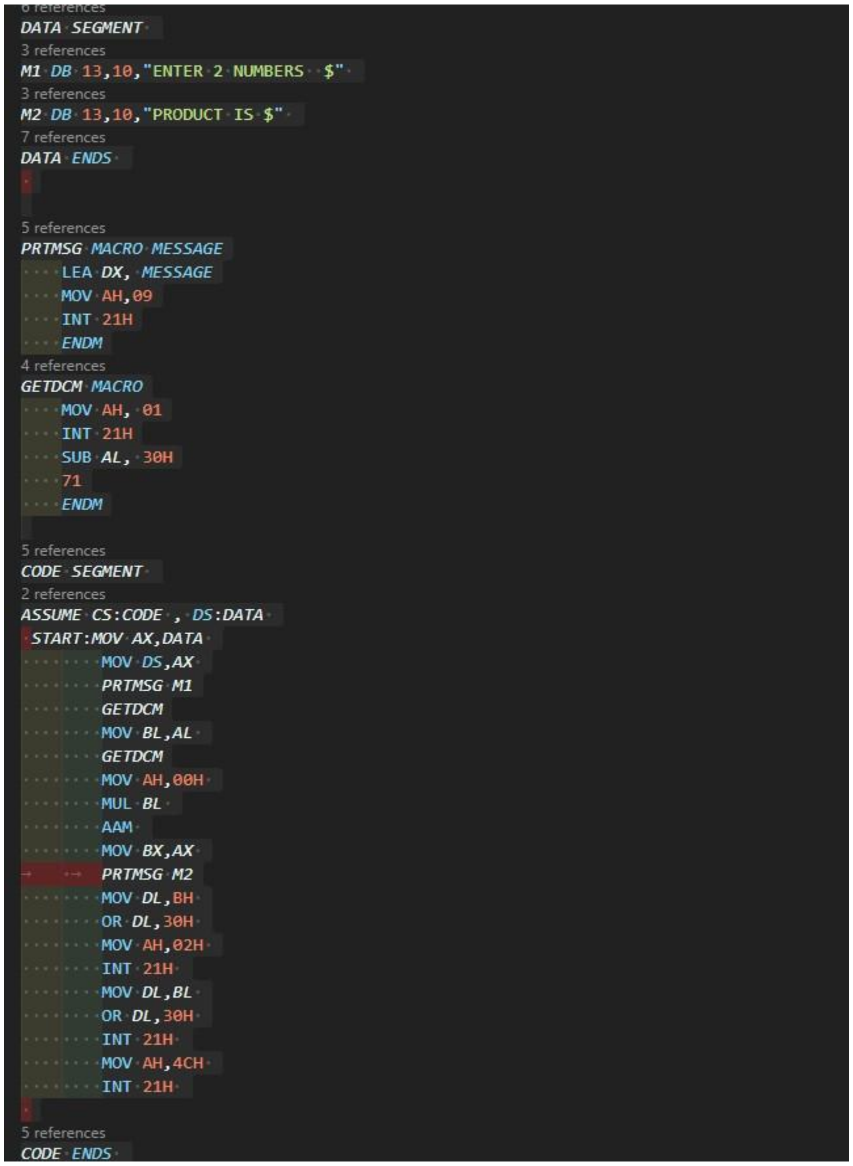
OUTPUT



```
C:\>add

Enter first number:9
Enter second number:9
Sum: 18
```

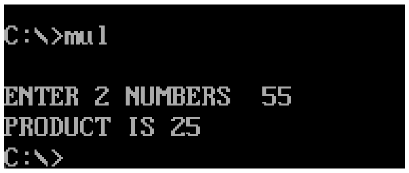
RESULT

MULTIPLICATION



```
6 references
DATA SEGMENT
3 references
M1 DB 13,10,"ENTER 2 NUMBERS  $"
3 references
M2 DB 13,10,"PRODUCT IS $"
7 references
DATA ENDS


5 references
PRTMSG MACRO MESSAGE
    LEA DX, MESSAGE
    MOV AH,09
    INT 21H
    ENDM
4 references
GETDCM MACRO
    MOV AH, 01
    INT 21H
    SUB AL, 30H
    71
    ENDM

5 references
CODE SEGMENT
2 references
ASSUME CS:CODE , DS:DATA
 START:MOV AX,DATA
        MOV DS,AX
        PRTMSG M1
        GETDCM
        MOV BL,AL
        GETDCM
        MOV AH,00H
        MUL BL
        AAM
        MOV BX,AX
        PRTMSG M2
        MOV DL,BH
        OR DL,30H
        MOV AH,02H
        INT 21H
        MOV DL,BL
        OR DL,30H
        INT 21H
        MOV AH,4CH
        INT 21H

5 references
CODE ENDS
```

OUTPUT



```
C:\>mul

ENTER 2 NUMBERS  55
PRODUCT IS 25
C:\>
```

RESULT

EXPERIMENT 9

Check number is ODD/EVEN using MASM

AIM

Write a program to check whether the given number is ODD/EVEN using MASM

INPUT

A number

OUTPUT

ODD or EVEN

ALGORITHM

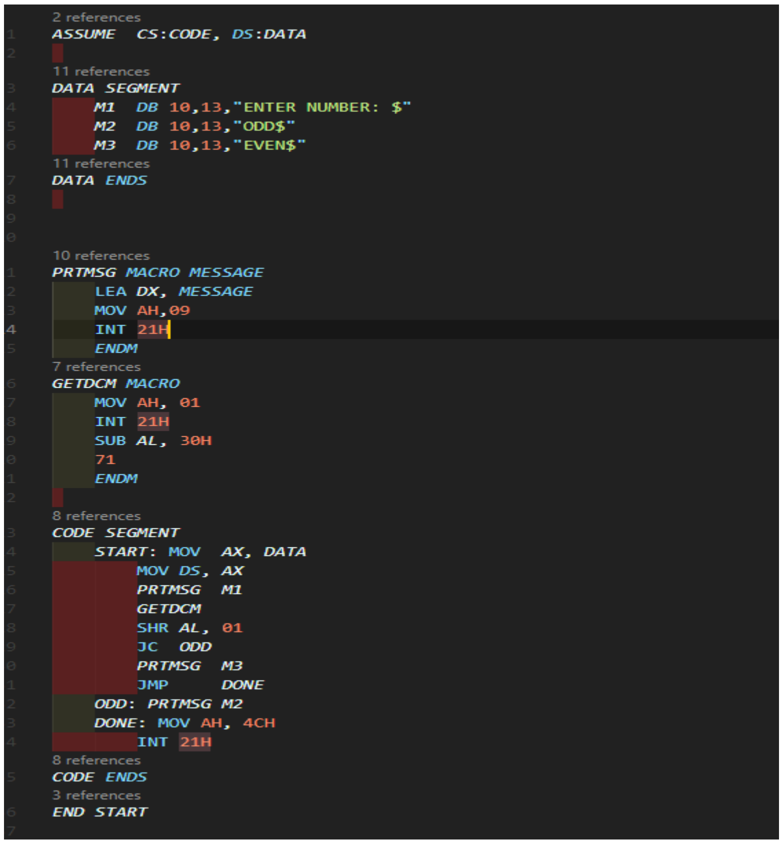
1) Load data segment starting address to DS
2) Get the first number and store it in AL
3) Do Shift right with carry in AL
4) If carry is present jump to ODD label
5) If not present Print "EVEN"
6) Jump to Stop
7) ODD: print "ODD"
8) Stop

PROGRAM

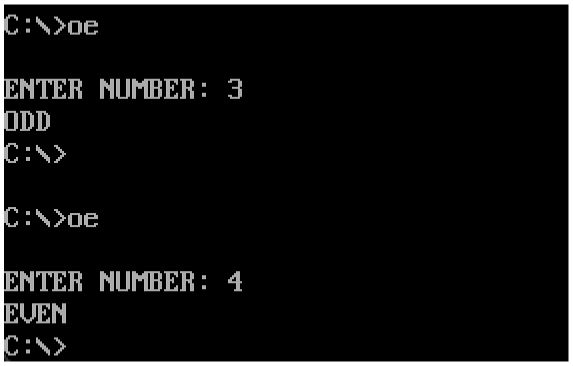
```
ASSUME          CS:CODE, DS:DATA

DATA SEGMENT
        M1      DB 10,13,"ENTER NUMBER: $"
        M2      DB 10,13,"ODD$"
        M3      DB 10,13,"EVEN$"
DATA ENDS
PRTMSG MACRO MESSAGE
   LEA DX, MESSAGE
   MOV AH,09
   INT 21H
   ENDM
```

```
GETDCM MACRO
    MOV AH, 01
    INT 21H
    SUB AL, 30H
    ENDM

CODE SEGMENT
        START: MOV    AX, DATA
               MOV DS, AX
               PRTMSG        M1
               GETDCM
               SHR AL, 01
               JC       ODD
               PRTMSG        M3
               JMP      DONE
        ODD: PRTMSG M2
        DONE: MOV AH, 4CH
               INT 21H
CODE ENDS
END START
```

SAMPLE CODE AND OUTPUT



```
2 references
ASSUME  CS:CODE, DS:DATA

11 references
DATA SEGMENT
    M1  DB 10,13,"ENTER NUMBER: $"
    M2  DB 10,13,"ODD$"
    M3  DB 10,13,"EVEN$"
11 references
DATA ENDS



10 references
PRTMSG MACRO MESSAGE
    LEA DX, MESSAGE
    MOV AH,09
    INT 21H
    ENDM
7 references
GETDCM MACRO
    MOV AH, 01
    INT 21H
    SUB AL, 30H
    71
    ENDM

8 references
CODE SEGMENT
    START: MOV  AX, DATA
        MOV DS, AX
        PRTMSG  M1
        GETDCM
        SHR AL, 01
        JC  ODD
        PRTMSG  M3
        JMP     DONE
    ODD: PRTMSG M2
    DONE: MOV AH, 4CH
        INT 21H
8 references
CODE ENDS
3 references
END START
```



```
C:\>oe

ENTER NUMBER: 3
ODD
C:\>

C:\>oe

ENTER NUMBER: 4
EVEN
C:\>
```

RESULT

EXPERIMENT 10

16 bit addition and multiplication using MASM

AIM

Write a program to implement 16 bit addition and multiplication using MASM

INPUT

Two numbers

OUTPUT

SUM and Product

ADDITION
ALGORITHM

```
1)  Load data segment starting address to DS
2)  Get the first number and store it in AX
3)  Move it to BX
4)  Get second number and store it in AX
5)  Move it to CX
6)  Add CL add BL
7)  Copy BL to AL
8)   Covert to BCD using AAA
9)  Store destination address in SI
10) Move AL to Destination
11)ADD AH to BH
12)Add CH to BH
13)Mov BH to AL
14)Covert to BDC format using AAA
15)Increment SI
16)Copy AL to address pointed by SI
17)Print SI
18)Stop
```

PROGRAM

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
          M1        DB 10,13,"ENTER FIRST NUMBER: $"
          M2        DB 10,13,"ENTER SECOND NUMBER: $"
```

```
       M3       DB 10,13,"SUM: $"
  SUM DB 03
DATA ENDS

PRTMSG MACRO MESSAGE
  LEA DX, MESSAGE
  MOV AH,09
  INT 21H
  ENDM
GETDCM MACRO
  MOV AH, 01
  INT 21H
  SUB AL, 30H
  ENDM

PRTDCM          MACRO
  MOV DL,[SI]
  ADD  DL, 30H
  MOV  AH, 02
  INT     21H
  ENDM

CODE SEGMENT
       START: MOV       AX, DATA
              MOV DS, AX
              PRTMSG         M1
              GETDCM
         MOV BH, AL
    GETDCM
              MOV BL, AL
              PRTMSG         M2
              GETDCM
         MOV CH, AL
    GETDCM
              MOV CL, AL
              ADD BL, CL
    MOV AL, BL
         MOV AH, 00
         AAA
         LEA SI, SUM
         MOV [SI], AL
         ADD  BH,  AH
         ADD  BH,  CH
         MOV  AL,  BH
         MOV  AH,  00
         AAA
    INC SI
         MOV [SI], AL
         INC SI
         MOV [SI], AH
    PRTMSG     M3
         PRTDCM
    DEC SI
    PRTDCM
    DEC SI
              PRTDCM
              MOV AH, 4CH
```

```
                INT 21H
CODE ENDS
END START
```

MULTIPLICATION

ALGORITHM

1) Load data segment starting address to DS
2) Get the first number and store it in AX
3) Get the second number and store in CX
4) Copy destination address to SI
5) Multiply CL and BL
6) Store it in destination
7) Multiply CL and store it in DX
8) Add data in destination with DL
9) Multiply BL with AL
10) Add the value in AL with DL
11) Store the value in destination
12) Display the value
13) Stop

PROGRAM

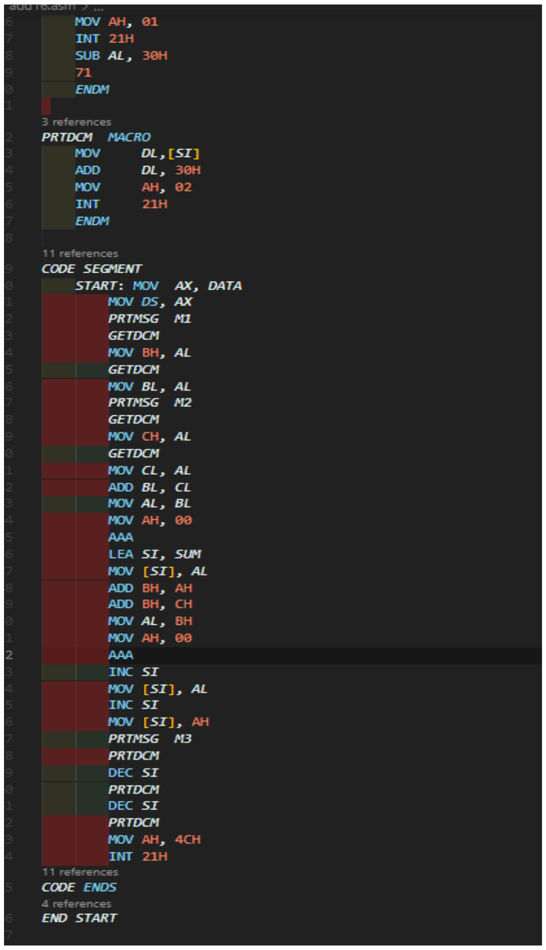
```
ASSUME          CS:CODE, DS:DATA

DATA SEGMENT
        M1      DB 10, 13, "ENTER FIRST NUMBER: $"
        M2      DB 10, 13, "ENTER SECOND NUMBER: $"
        M3      DB 10, 13, "PRODUCT: $"
        PROD DB 4 DUP(00H)

DATA ENDS

PRTMSG          MACRO           MESSAGE
   LEA DX, MESSAGE
   MOV AH, 09
   INT 21H
   ENDM

GETDCM          MACRO
   MOV AH, 01
   INT   21H
   SUB AL, 30H
   ENDM

PRTDCM          MACRO
```

```
   MOV  DL, [SI]
   ADD DL, 30H
   MOV AH, 02
         INT       21H
   ENDM

CODE SEGMENT
                  START:         MOV AX, DATA
                           MOV DS, AX
                           PRTMSG        M1
                           GETDCM
                     MOV BH, AL
         GETDCM
                           MOV BL, AL
                           PRTMSG        M2
                           GETDCM
                     MOV CH, AL
         GETDCM
                           MOV CL, AL
                           LEA SI, PROD
                           MOV AH, 00H
                           MUL BL
                           AAM
                           MOV [SI], AL
                           INC      SI
                           MOV [SI], AH
                     MOV AH, 00H
                     MOV AL, BH
                     MUL CL
                     AAM
                     MOV DX, AX
                           ADD DL, [SI]
                           MOV AH, 00H
                           MOV AL, CH
                           MUL BL
                           AAM
                           ADD DX, AX
                           MOV AL, DL
                           MOV AH, 00H
                           AAM
                           ADD DH, AH
                           MOV DL, DH
                           MOV DH, 00H
                           MOV [SI], AL
                           INC      SI
                           MOV AH, 00H
                           MOV AL, BH
                           MUL CH
                           AAM
                           ADD DX, AX
                           MOV AL, DL
                           MOV AH, 00H
                           AAM
```

```
                    MOV [SI], AL
               INC  SI
     ADD DH, AH
     MOV AL, DH
          MOV [SI], AL
     PRTMSG         M3
                    PRTDCM
                    DEC      SI
                    PRTDCM
                    DEC      SI
                    PRTDCM
                    DEC      SI
                    PRTDCM
                    MOV AH, 4CH
                    INT       21H

CODE ENDS
END START
```

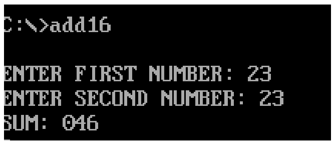
SAMPLE CODE AND OUTPUT

ADDITION



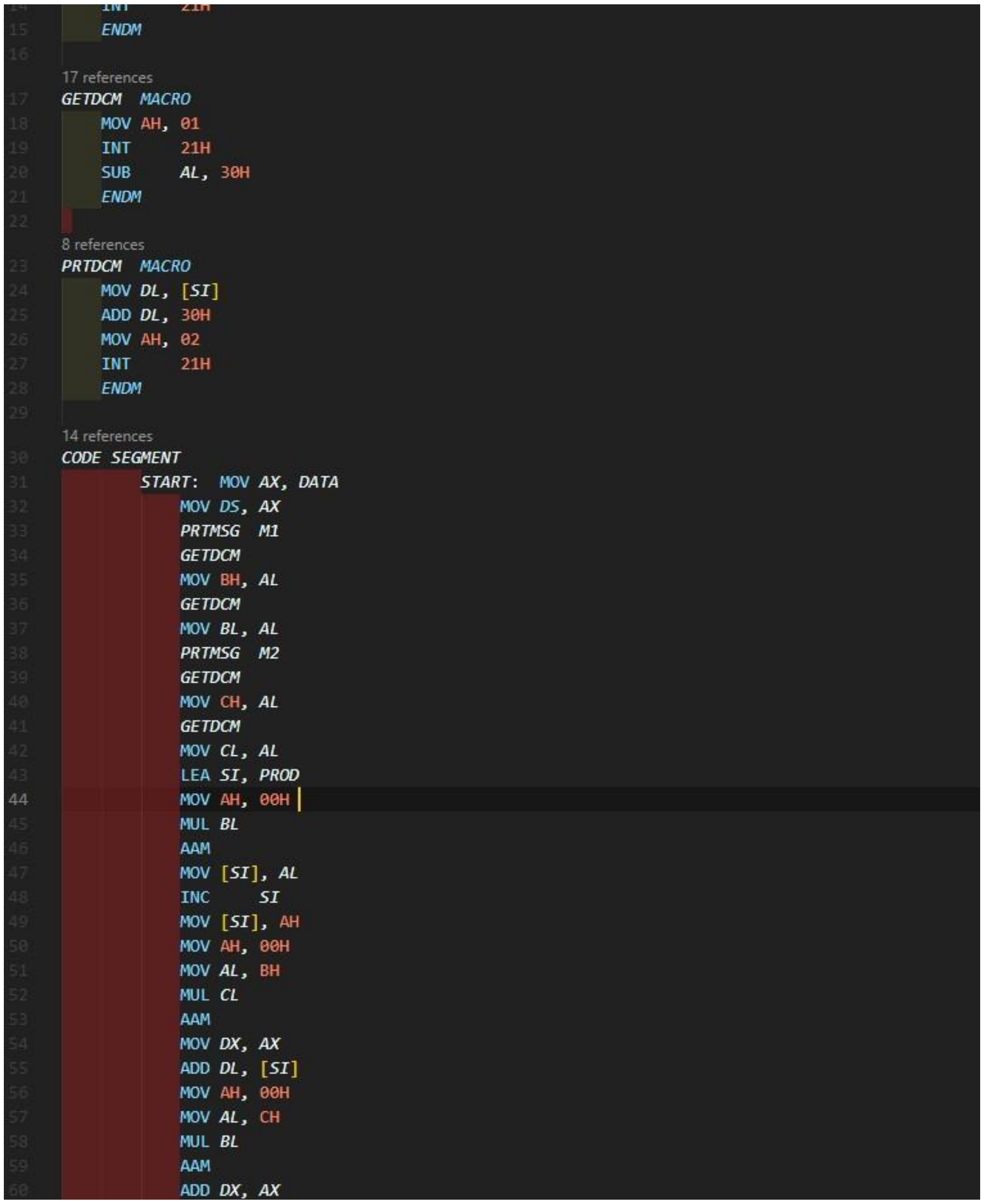
```
add16.asm > ...
        MOV AH, 01
        INT 21H
        SUB AL, 30H
        71
        ENDM

3 references
PRTDCM  MACRO
        MOV     DL,[SI]
        ADD     DL, 30H
        MOV     AH, 02
        INT     21H
        ENDM

11 references
CODE SEGMENT
        START: MOV  AX, DATA
            MOV DS, AX
            PRTMSG  M1
            GETDCM
            MOV BH, AL
            GETDCM
            MOV BL, AL
            PRTMSG  M2
            GETDCM
            MOV CH, AL
            GETDCM
            MOV CL, AL
            ADD BL, CL
            MOV AL, BL
            MOV AH, 00
            AAA
            LEA SI, SUM
            MOV [SI], AL
            ADD BH, AH
            ADD BH, CH
            MOV AL, BH
            MOV AH, 00
            AAA
            INC SI
            MOV [SI], AL
            INC SI
            MOV [SI], AH
            PRTMSG  M3
            PRTDCM
            DEC SI
            PRTDCM
            DEC SI
            PRTDCM
            MOV AH, 4CH
            INT 21H
11 references
CODE ENDS
4 references
END START
```

RESULT

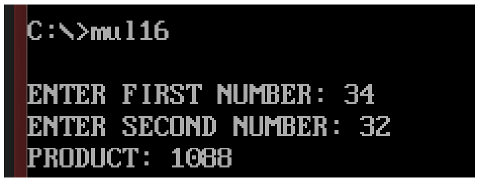


```
C:\>add16

ENTER FIRST NUMBER: 23
ENTER SECOND NUMBER: 23
SUM: 046
```

MULTPILICATION



```
    INT     21H
    ENDM

17 references
GETDCM  MACRO
    MOV AH, 01
    INT     21H
    SUB     AL, 30H
    ENDM

8 references
PRTDCM  MACRO
    MOV DL, [SI]
    ADD DL, 30H
    MOV AH, 02
    INT     21H
    ENDM

14 references
CODE SEGMENT
        START:  MOV AX, DATA
            MOV DS, AX
            PRTMSG  M1
            GETDCM
            MOV BH, AL
            GETDCM
            MOV BL, AL
            PRTMSG  M2
            GETDCM
            MOV CH, AL
            GETDCM
            MOV CL, AL
            LEA SI, PROD
            MOV AH, 00H
            MUL BL
            AAM
            MOV [SI], AL
            INC     SI
            MOV [SI], AH
            MOV AH, 00H
            MOV AL, BH
            MUL CL
            AAM
            MOV DX, AX
            ADD DL, [SI]
            MOV AH, 00H
            MOV AL, CH
            MUL BL
            AAM
            ADD DX, AX
```



```
C:\>mul16

ENTER FIRST NUMBER: 34
ENTER SECOND NUMBER: 32
PRODUCT: 1088
```

EXPERIMENT 11

Linear Search using MASM

AIM

Write a program to implement Linear Search using MASM

INPUT

Numbers

OUTPUT

Location of key

ALGORITHM

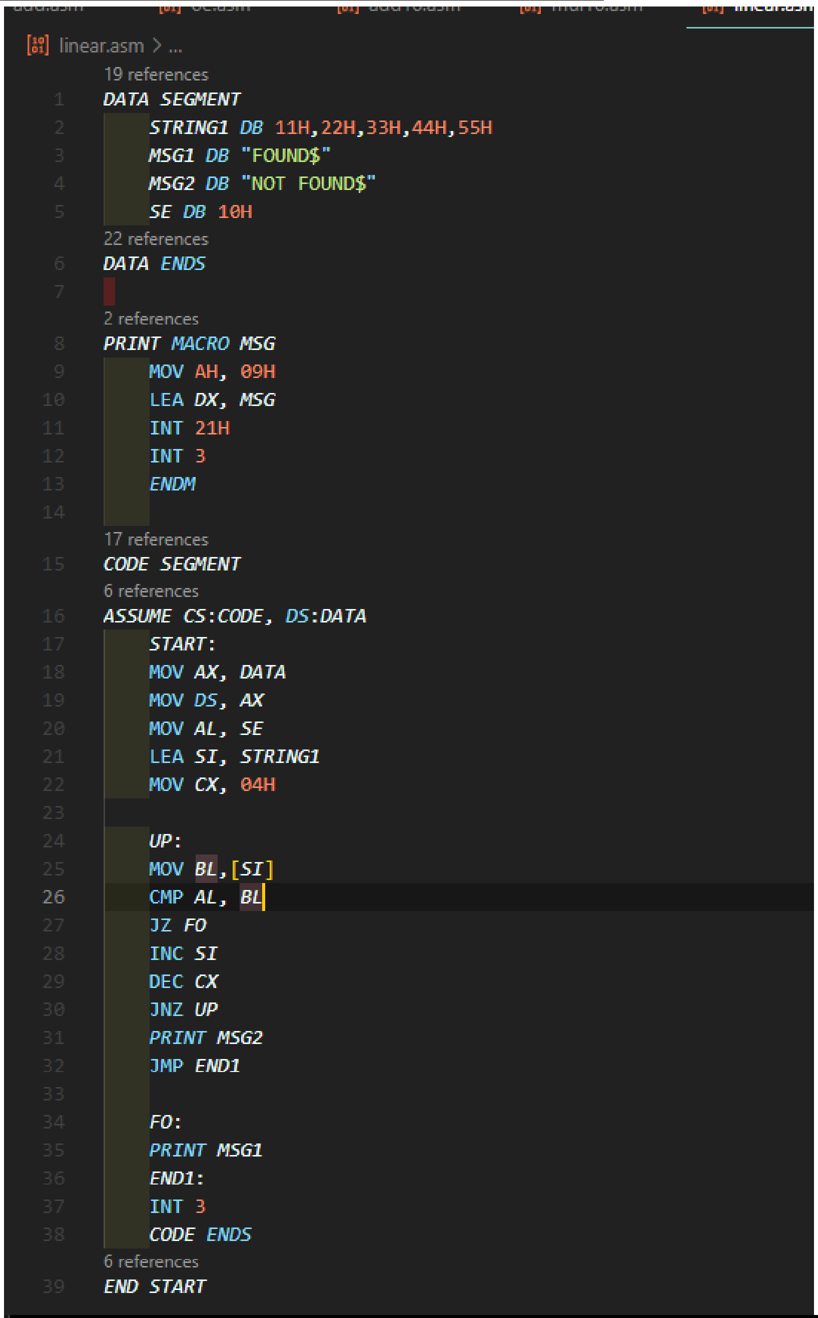
```
1) Load data segment starting address to DS
2) Copy value to be sreached to AL
3) Copy starting address of array to SI
4) Store size of the array to CX
5) UP:
6) Move first element to BL
7) Compare with AL
8) If Zero jump to FO:
9) Else
10)Increment SI
11) Decrement CX
12) If CX not zero Jump to UP:
13)  Print NOT FOUND
14)Jump to stop
15)FO:
16)Print FOUND
17)Stop
```

PROGRAM

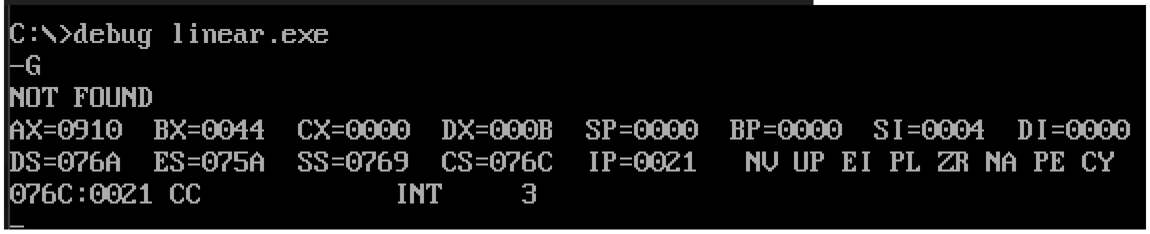
```
DATA SEGMENT
   STRING1 DB 11H,22H,33H,44H,55H
   MSG1 DB "FOUND$"
   MSG2 DB "NOT FOUND$"
   SE DB 10H
DATA ENDS

PRINT MACRO MSG
```

```
    MOV AH, 09H
    LEA DX, MSG
    INT 21H
    INT   3
    ENDM

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
    START:
    MOV AX, DATA
    MOV DS, AX
    MOV AL, SE
    LEA SI, STRING1
    MOV CX, 04H

    UP:
    MOV BL,[SI]
    CMP AL, BL
    JZ FO
    INC SI
    DEC CX
    JNZ UP
    PRINT MSG2
    JMP END1

    FO:
    PRINT MSG1
    END1:
    INT 3
    CODE ENDS
END START
```

SAMPLE CODE AND OUTPUT



```
DATA SEGMENT
     STRING1 DB 11H,22H,33H,44H,55H
     MSG1 DB "FOUND$"
     MSG2 DB "NOT FOUND$"
     SE DB 10H
DATA ENDS

PRINT MACRO MSG
     MOV AH, 09H
     LEA DX, MSG
     INT 21H
     INT 3
     ENDM

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
     START:
     MOV AX, DATA
     MOV DS, AX
     MOV AL, SE
     LEA SI, STRING1
     MOV CX, 04H

     UP:
     MOV BL,[SI]
     CMP AL, BL
     JZ FO
     INC SI
     DEC CX
     JNZ UP
     PRINT MSG2
     JMP END1

     FO:
     PRINT MSG1
     END1:
     INT 3
     CODE ENDS
END START
```



```
C:\>debug linear.exe
-G
NOT FOUND
AX=0910  BX=0044  CX=0000  DX=000B  SP=0000  BP=0000  SI=0004  DI=0000
DS=076A  ES=075A  SS=0769  CS=076C  IP=0021   NV UP EI PL ZR NA PE CY
076C:0021 CC            INT     3
-
```

RESULT

EXPERIMENT 12

String manipulation using MASM

AIM
To find number of vowels, consonants and digits in string

INPUT
String

OUTPUT
count

ALGORITHM

```
1) Load data segment starting address to DS
2) Load extra segment starting address to ES
3) Copy starting address of the string to SI
4) Move maxlength to CL
5) GETC:  call interrupt 21
6) Compare AL with DELIM
7) Jump to ENDET: if equal
8) Increament BL
9) Move AL to Destination
10)Increment SI
11) Loop GETC
12)ENDGET:  CLD
13)Copy starting address of string to SI
14)Move content of SI to AX
15)Increament SI
16)Copy the starting address of vowels to DI
17)Repeat when not zero SCASB:
18)Jump on not equal CHKC
19)Increament VCNT
20)Jump to ENDC
21)Display number of vowels, consonants,digits in the string
```

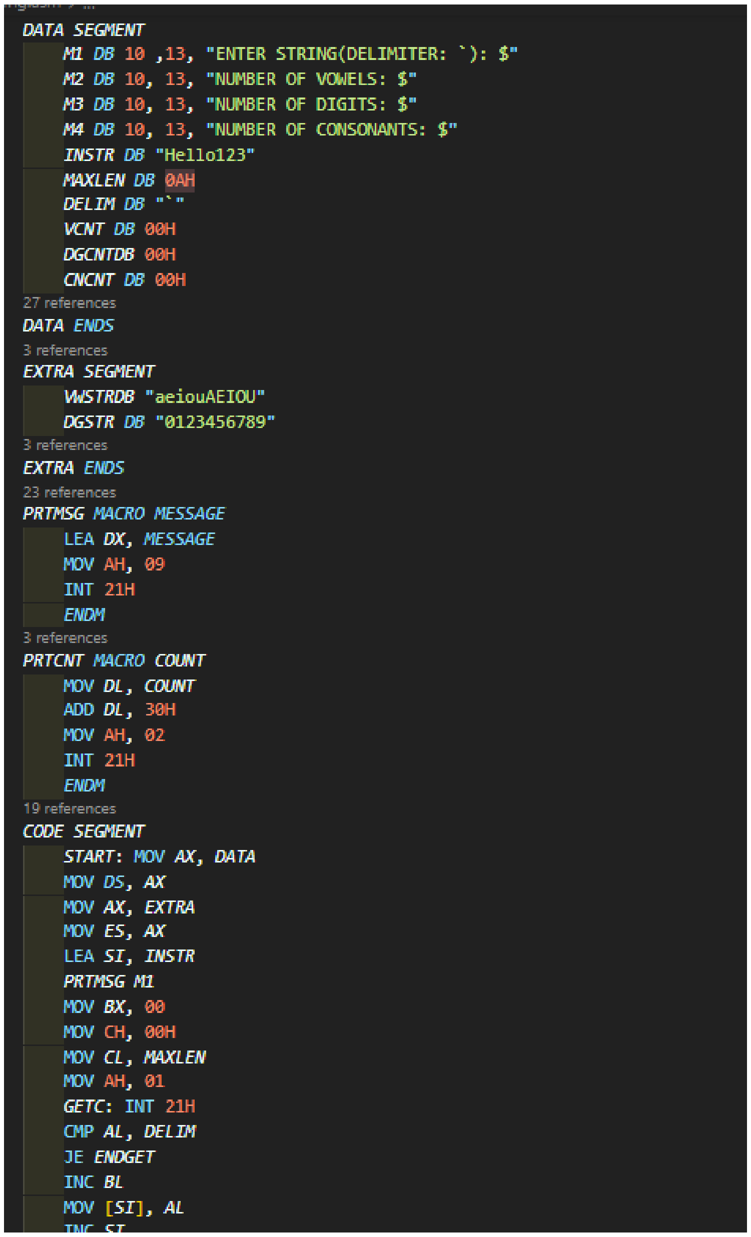
PROGRAM

```
ASSUME CS:CODE,DS:DATA,ES:EXTRA
```

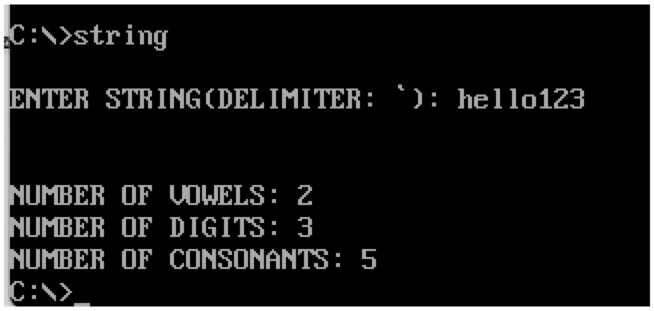
```
DATA SEGMENT
   M1 DB 10 ,13, "ENTER STRING(DELIMITER: `): $"
   M2 DB 10, 13, "NUMBER OF VOWELS: $"
   M3 DB 10, 13, "NUMBER OF DIGITS: $"
   M4 DB 10, 13, "NUMBER OF CONSONANTS: $"
   INSTR DB "Hello123"
   MAXLEN DB 0AH
   DELIM DB "`"
   VCNT DB 00H
   DGCNT DB 00H
   CNCNT DB 00H
DATA ENDS
EXTRA SEGMENT
   VWSTR DB "aeiouAEIOU"
   DGSTR DB "0123456789"
EXTRA ENDS
PRTMSG MACRO MESSAGE
   LEA DX, MESSAGE
   MOV AH, 09
   INT 21H
   ENDM
PRTCNT MACRO COUNT
   MOV DL, COUNT
   ADD DL, 30H
   MOV AH, 02
   INT 21H
   ENDM
CODE  SEGMENT
   START: MOV AX, DATA
   MOV DS, AX
   MOV AX, EXTRA
   MOV ES, AX
   LEA SI, INSTR
   PRTMSG M1
   MOV BX, 00
   MOV CH, 00H
   MOV CL, MAXLEN
   MOV AH, 01
   GETC: INT 21H
   CMP AL, DELIM
   JE ENDGET
   INC BL
   MOV [SI], AL
   INC  SI
   LOOP GETC
   ENDGET: CLD
   LEA SI, INSTR
   CHKA: MOV AX, [SI]
```

```
    INC SI
    MOV CL, 0AH
    LEA DI, VWSTR
    REPNZ SCASB
    JNE CHKD
    INC VCNT
    JMP ENDC
    CHKD: MOV CL, 0AH
    LEA DI, DGSTR
    REPNZ SCASB
    JNE CHKC
    INC DGCNT
    JMP ENDC
    CHKC: INC CNCNT
    ENDC: MOV CL, BL
    DEC BX
    LOOP CHKA
    PRTMSG M2
    PRTCNT VCNT
    PRTMSG M3
    PRTCNT DGCNT
    PRTMSG M4
    PRTCNT CNCNT
    MOV AH, 4CH
    INT 21H
    CODE ENDS
END START
```

SAMPLE CODE AND OUTPUT



```
DATA SEGMENT
    M1 DB 10 ,13, "ENTER STRING(DELIMITER: `): $"
    M2 DB 10, 13, "NUMBER OF VOWELS: $"
    M3 DB 10, 13, "NUMBER OF DIGITS: $"
    M4 DB 10, 13, "NUMBER OF CONSONANTS: $"
    INSTR DB "Hello123"
    MAXLEN DB 0AH
    DELIM DB "`"
    VCNT DB 00H
    DGCNTDB 00H
    CNCNT DB 00H
DATA ENDS
EXTRA SEGMENT
    VWSTRDB "aeiouAEIOU"
    DGSTR DB "0123456789"
EXTRA ENDS
PRTMSG MACRO MESSAGE
    LEA DX, MESSAGE
    MOV AH, 09
    INT 21H
    ENDM
PRTCNT MACRO COUNT
    MOV DL, COUNT
    ADD DL, 30H
    MOV AH, 02
    INT 21H
    ENDM
CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV AX, EXTRA
    MOV ES, AX
    LEA SI, INSTR
    PRTMSG M1
    MOV BX, 00
    MOV CH, 00H
    MOV CL, MAXLEN
    MOV AH, 01
    GETC: INT 21H
    CMP AL, DELIM
    JE ENDGET
    INC BL
    MOV [SI], AL
    INC SI
```



```
C:\>string

ENTER STRING(DELIMITER: `): hello123


NUMBER OF VOWELS: 2
NUMBER OF DIGITS: 3
NUMBER OF CONSONANTS: 5
C:\>_
```

RESULT

TRAINER KIT PROGRAM

EXPERIMENT NO : 13

ADDITION OF TWO 16 BIT NUMBERS USING 8086 TRAINER KIT

AIM

To add two 16-bit numbers using 8086 trainer kit.

ALGORITHM

1. Clear the AX by performing AND operation with 0000
2. Move the location where result is to be stored to BX
3. Move the location of operand 1 to SI
4. Move the location of operand 2 to DI
5. Move the contents of SI to AX
6. Add the contents of DI to AX
7. Move the result to the location stored in BX
8. Move 0000H to AX
9. Add the carry flag to AX
10. Move the result to the location stored in [BX + 2]
11. Halt

PROGRAM

ADDRESS	MNEMONICS
0400	AND AX,0000H
0403	MOV BX,0600H
0406	MOV SI,0500H
0409	MOV DI,0550H
040C	MOV AX,[SI]
040E	ADD AX,[DI]
0410	MOV [BX],AX
0412	MOV AX,0000H
0415	ADC AX,0000H
0418	MOV [BX+2],AX
041B	HLT

INPUT

0500 - B5
0501 - 7A
0550 - 2A
0551 – E5

OUTPUT

0600 - DF
0601 - 5F
0602 - 01

RESULT

EXPERIMENT NO : 14

SUBTRACTION OF TWO 16 BIT NUMBERS USING 8086 TRAINER KIT

AIM

To subtract two 16-bit numbers using 8086 trainer kit.

ALGORITHM

1. Clear the carry flag
2. Move the location where result is to be stored to BX
3. Move the location of operand 1 to SI
4. Move the location of operand 2 to DI
5. Move the contents of SI to AX
6. Subtract the contents of DI from AX including the borrow value
7. Move the result to the location stored in BX
8. Halt

PROGRAM

ADDRESS	MNEMONICS
0400	CLC
0401	MOV BX,0900H
0404	MOV SI,0700H
0407	MOV DI,0800H
040A	MOV AX,[SI]
040C	SBB AX,[DI]
040E	MOV [BX],AX
0410	HLT

INPUT

0700 - 18
0701 - 08
0800 - 40
0801 - 10

OUTPUT

```
0900 - D8
0901 - F7
```

RESULT

EXPERIMENT NO : 15

MULTIPLICATION OF TWO 16 BIT NUMBERS USING 8086 TRAINER KIT

AIM

To multiply two 16-bit numbers using 8086 trainer kit.

ALGORITHM

1. Clear the carry flag
2. Move the location where result is to be stored to BX
3. Move the location of operand 1 to SI
4. Move the location of operand 2 to DI
5. Move the contents of SI to AX
6. Move the contents of DI to CX
7. Multiply CX to AX
8. Move the result from AX to the location stored in BX
9. Move the higher bits of result from DX to the location stored in [BX+2]
10. Halt

PROGRAM

```
ADDRESS         MNEMONICS

0400            CLC
0401            MOV BX,0700H
0404            MOV SI,0750H
0407            MOV DI,0800H
040A            MOV AX,[SI]
040C            MOV CX,[DI]
040E            MUL CX
0410            MOV [BX],AX
0412            MOV [BX+2],DX
0415            HLT
```

INPUT

0750 - 1A
0751 - 2B
0800 - 4B
0801 - 12

OUTPUT

0700 - 9E
0701 - 74
0702 - 14
0703 - 03

RESULT

EXPERIMENT NO : 16

DIVISION OF A 16 BIT NUMBER BY AN 8 BIT NUMBER USING 8086 TRAINER KIT

AIM

To divide a 16-bit number by an 8 bit number using 8086 trainer kit.

ALGORITHM

1. Clear the carry flag
2. Move the location where result is to be stored to BX
3. Move the location of operand 1 to SI
4. Move the location of operand 2 to DI
5. Move the contents of SI to AX
6. Move the contents of DI to CX
7. Move 00 to CH
8. Divide CL from AX
9. Move the result from AX to the location stored in BX
10. Halt

PROGRAM

ADDRESS	MNEMONICS
0400	CLC
0401	MOV BX,0700H
0404	MOV SI,0750H
0407	MOV DI,0800H
040A	MOV AX,[SI]
040C	MOV CX,[DI]
040E	MOV CH,00H
0410	DIV CL
0412	MOV [BX],AX
0414	HLT

INPUT

0750 - 43	
0751 - 12	
0800 - 21	

OUTPUT

0700 - 8D	(Quotient)
0701 - 16	(Remainder)

RESULT

EXPERIMENT NO : 17

MAXIMUM OF N NUMBERS USING 8086 TRAINER KIT

AIM

To find the maximum of n numbers using the 8086 trainer kit.

ALGORITHM

1. Clear the carry flag
2. Move the location where the result has to be stored to BX
3. Move the starting location of array to SI
4. Move the total number of elements in the array to CX
5. Move 00 to AL
6. Compare the contents of SI with AL
7. Jump to step 9 if above instruction satisfies
8. Else move the contents of SI to AL
9. Move 00 to CH
10. Increment SI
11. Continue the loop of comparing the contents of SI and AL till the counter reaches zero (LOOPNZ only loops when the zero flag is not set)
12. Move the result, ie, maximum number from AL to the location stored in BX
13. Halt

PROGRAM

ADDRESS	MNEMONICS
0400	CLC
0401	MOV BX,0700H
0404	MOV SI,0800H
0407	MOV CX,0005H
040A	MOV AL,00H
040C	CMP AL,[SI]
040E	JA 0412H
0410	MOV CH,00H
0412	INC SI

```
0413                LOOPNZ 040CH
0415                MOV [BX],AL
0417                HLT
```

INPUT

```
0800 - 77
0801 - 81
0802 - B4
0803 - F1
0804 - AB
```

OUTPUT

```
0700 - F1
```

RESULT

EXPERIMENT NO : 18

SORTING NUMBERS IN ASCENDING ORDER USING 8086 TRAINER KIT

AIM

To sort the numbers in ascending order using 8086 trainer kit.

ALGORITHM

1. Set the value of SI to 500.
2. Load data from offset SI to register CL.
3. Decrease value of register CL by 1.
4. Set the value of SI to 500.
5. Load data from offset SI to register CH. Decrease value of register CH by 1
6. Increase the value of SI by 1.
7. Load value from offset SI to register AL.
8. Increase the value of SI by 1.
9. Compare the value of register AL and [SI] ,ie,(AL-[SI]).
10. Jump to address 41C if carry is generated.
11. Exchange the contents of register AL and SI.
12. Decrease the value of SI by 1.
13. Exchange the contents of register AL and SI
14. Increase the value of SI by 1.
15. Decrease the value of register CH by 1.
16. Jump to address 40F if zero flat reset
17. Decrease the value of register CL by 1.
18. Jump to address 407 if zero flat reset.
19. Stop

PROGRAM

```
ADDRESS         MNEMONICS

0400            MOV SI,500
0403            MOV CL,[SI]
0405            DEC CL
```

```
0407                MOV SI,500
0409                MOV CH,[SI]
040C                DEC CH
040E                INC SI
040F                MOV AL,[SI]
0411                INC SI
0412                CMP AL,[SI]
0414                JC 041C
0416                XCHG AL,[SI]
0418                DEC SI
0419                XCHG AL,[SI]
041B                INC SI
041C                DEC CH
041E                JNZ 40F
0420                DEC CL
0422                JNZ 407
0424                HLT
```

INPUT

0500 - 5
0501 - 6
0502 - 8
0503 - 3
0504 - 5
0505 - 4

OUTPUT

0500 - 5
0501 - 3
0502 - 4
0503 - 5
0504 - 6
0505 - 8

RESULT