Olp :-

-u :-

```
076B : 0000  B86A07      MOV   AX, 076A
076B : 0003  8ED8        MOV   DS, AX
076B : 0005  B800000     MOV   AX, 0000
076B : 0008  B104        MOV   CL, 04
076B : 000A  8D360000    LEA   SI, [0000]
076B : 000E  8A04        MOV   AL, [SI]
076B : 0010  3A4401      CMP   AL, [SI+01]
076B : 0013  7303        JNB   0018
076B : 0015  8A4401      MOV   AL, [SI+01]
076B : 0018  46          INC   SI
076B : 0019  FEC9        DEC   CL
076B : 001B  75F3        JNZ   0010
076B : 001D  A20500      MOV   [0005], AL
076B : 0020  CC          INT   3
```

-g -

AX=0021 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:00

076A : 0000 04 0A 01 21 09 21 00

Program - 19

Write an ALP to find largest number from given series of numbers.

```
assume cs:code, ds:data
Data segment
list db 04H, 0AH, 01H, 21H, 09H
larg db 00H
Data ends
Code segment
Start : mov AX, data
        mov DS, AX
        mov AX, 00H
        mov CL, 04H
        lea SI, list
        mov AL, [SI]
L2 :    cmp AL, [SI+1]
        JNC L1
        mov AL, [SI+1]
L1 :    INC SI
        DEC CL
        JNZ L2
        mov larg, AL
        INT 03H
        Code ends
        end start
```

NOTE

Cmp D, S

S>D JC CF=1

S<D JNC CF=0

S=D ZF=1

(largest = 21)

olp.

-u.

```
076B:0000  B86A07    MOV  AX,076A
076B:0003  8ED8      MOV  DS,AX
076B:0005  B80000    MOV  AX,0000
076B:0008  B104      MOV  CL,04
076B:000A  8D360000  LEA  SI,[0000]
076B:000E  8A04      MOV  AL,[SI]
076B:0010  3A4401    CMP  AL,[SI+01]
076B:0013  7203      JB   0018
076B:0015  8A4401    MOV  AL,[SI+01]
076B:0018  46        INC  SI
076B:0019  FEC9      DEC  CL
076B:001B  75F3      JNZ  0010
076B:001D  A20500    MOV  [0005],AL
076B:0020  CC        INT  3
```

-g.

```
AX=0001  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:00,

```
076A:0000  04 0A 01 21 09 01 00
```

Program - 10

→ Write an ALP to find smallest number from given series

```
Assume CS:code, ds:data
Data segment
list db 04H, 0AH, 01H, 21H, 09H
small db 00H
Data ends
code segment
start:  mov AX,data
        mov DS,AX
        mov AX,00H
        mov CL,04H
        lea SI,list
        mov AL,[SI]
L2:     cmp AL,[SI+1]
        JC  L1
        mov AL,[SI+1]
L1:     INC SI
        DEC CL
        JNZ L2
        mov small,AL
        INT 03H
code ends
end start
```


Mal 8/3

dp:-

```
-u
076B:0000  B86A07      MOV     AX,076A
076B:0003  8ED8        MOV     DS,AX
076B:0005  B80000      MOV     AX,0000
076B:0008  B104        MOV     CL,04
076B:000A  8D360000    LEA     SI,[0000]
076B:000E  B204        MOV     DL,04
076B:0010  8A04        MOV     AL,[SI]
076B:0012  3A4401      CMP     AL,[SI+1]
076B:0015  7305        JNB     001C
076B:0017  864401      XCHG    AL,[SI+1]
076B:001A  8804        MOV     [SI],AL
076B:001C  46          INC     SI
076B:001D  FECA        DEC     DL
076B:001F  75EF        JNZ     0010
076B:0021  FEC9        DEC     CL
076B:0023  75E5        JNZ     000A
076B:0025  CC          INT     3

-g

AX=0002 BX=0000 CX=0000 DX=0000 SP=0000
BP=0000 SI=0004 DI=0000 DS=076A ES=075A
SS=0769 CS=076B IP=0025 NV UP EI PL ZR
NA PE NC
076B:0025  CC    INT   3

-d ds:00

076A:0000  45 32 22 02 01 00 00 00
```

Program-21.

Write an ALP to sort the given series of numbers in descending order.

```
Assume 'CS:code', DS:data
Data Segment
list db 02H, 22H, 45H, 01H, 32H
Data ends
code segment
start:
    Mov Ax,data
    mov DS,Ax
    mov Ax,0H
    mov CL,04
G03:
    lea SI,list
    mov DL,04H
G01:  MOV AL,[SI]
    CMP AL,[SI+1]
    JNC G02
    XCHG AL,[SI+1]
    MOV [SI],AL
G02:  INC SI
    DEC DL
    JNZ G01
    DEC CL
    JNZ G03
    INT 03H
code ends
end start
```

descending order

45, 32, 22, 2, 1

Olp:-

-u :-

```
076B:0000  B86A07    MOV  AX,076A
076B:0003  8ED8      MOV  DS,AX
076B:0005  B80000    MOV  AX,0000
076B:0008  B104      MOV  CL,04
076B:000A  8D360000  LEA  SI,[0000]
076B:000E  B204      MOV  DL,04
076B:0010  8A04      MOV  AL,[SI]
076B:0012  3A4401    CMP  AL,[SI+01]
076B:0015  7205      JB   001C
076B:0017  864401    XCHG AL,[SI+01]
076B:001A  8804      MOV  [SI],AL
076B:001C  46        INC  SI
076B:001D  FECA      DEC  DL
076B:001F  75EF      JNZ  0010
076B:0021  FEC9      DEC  CL
076B:0023  75E5      JNZ  000A
076B:0025  CC        INT  3
```

-g

```
AX=0032  BX=0000  CX=0000  DX=0000  SP=0000
BP=0000  SI=0004  DI=0000  DS=076A  ES=076A
SS=0769  CS=076B  IP=0025  NV UP EI PL ZR
NA PE CY
076B:0025  CC  INT 3
```

-d ds:00

```
076A:0000  01 02 22 32 45 00 00
```

Program-22

Write an ALP to sort the given series of numbers in ascending order.

```
Assume cs:code, ds:data
Data segment
list db 02H, 22H, 45H, 01H, 32H
Data ends
Code segment
Start:
        mov Ax,data
        mov ds,Ax
        mov Ax,0H
        mov CL,04
G03:
        Lea SI,list
        mov DL,04H
G01:
        mov AL,[SI]
        CMP AL,[SI+1]
        JC  G02
        XCHG AL,[SI+1]
        mov [SI],AL
G02:
        INC SI
        DEC DL
        JNZ G01
        DEC CL
        JNZ G03
        INT 03H
Code ends
end Start.
```

ascending order.

1, 2, 22, 32, 45

Operational code

```
076B : 0000   B8GA07      mov  AX,076A
076B : 0000   8ED8        mov  DS, AX
076B : 0005   8A0E0000    mov  CL,[0000]
076B : 0009   8D360100    lea  SI,[0001]
076B : 000D   B000        mov  AL,00
076B : 000F   8804        mov  [SI], AL
076B : 0011   B301        mov  BL,01
076B : 0013   46          Inc  SI
076B : 0014   02D8        Add  BL,AL
076B : 0016   881C        mov  [SI],BL
076B : 0018   8A44FF      mov  AL,[SI-01]
076B : 001B   FEC9        Dec  CL
076B : 001D   75F4        JNZ  0013
076B : 001F   CC          INT  3
```

```
-g

AX = 0722   BX = 0037   CX = 0000

DX = 0000

-d ds:00

076A : 0000   0A 00 01 01 02 03

              05 08 0D 15 22 37

              00 00 00
```

Program - 23

Write an ALP to display fibonacci series

```
Assume  cs: code, ds: data
 data segment
 count db 10 Dup(?)
 data ends
 code segment
 start :
        mov Ax, data
        mov DS, Ax
        mov CL, count

    lea  SI, Fib
    mov  AL, 01H
    mov  [SI], AL
    mov  BL, 01H

    GO :
        INC SI
        Add BL, AL
        mov [SI], BL
        mov AL, [SI-1]
        Dec CL
        JNZ GO

    INT  03H
    code ends
    end start
```

-u

```
076B:0000  B8GA07    MOV  AX,076A
076B:0003  8ED8      MOV  DS,AX
076B:0005  B8GB07    MOV  AX,076B
076B:0008  8EC0      MOV  ES,AX
076B:000A  8D360000  LEA  SI,[0000]
076B:000E  8D3E0000  LEA  DI,[0000]
076C:0012  B104      MOV  CL,04
076C:0014  FC        CLD
076C:0015  F3        REPZ
076C:0016  A4        MOVSB
076C:0017  CC        INT  3
```

-g

```
AX=076B  BX=0000  CX=0000  DX=0000
076C:0017  CC  INT 3
```

-d ds:00

```
076A:0000  61 69 6D 6C 00 00 00 00
                                    -aiml...
076A:0000  61 69 6D 6C 00 00 00 00
                                    -aiml....
```

program - 24

Write an ALP to move a string of datatypes from one location to another [copy a string]

```
assume cs:code, ds:data
Data segment
str1 db 'AIML'
Data ends
Extra segment
str2 db 00H
Extra ends
code segment
Start:
    mov AX,data
    mov DS,AX
    mov AX,Extra
    mov ES,AX
lea SI,str1
lea DI,str2
mov CL,04H
CLD
Rep MOVSB
INT 03H
code ends
end start
```

d ds:00 - AIML

d es:00 - AIML

operational output

```
076B:0000  B8 6A 07    MOV  AX,076A
076B:0003  8E D8       MOV  DS,AX
076B:0005  B8 0000     MOV  AX,0000
076B:0008  B1 09       MOV  CL,09
076B:000A  8D 36 0000  LEA  SI,[0000]
076B:000E  8D 3E 0900  LEA  DI,[0009]
076B:0012  8A 04       MOV  AL,[SI]
076B:0014  88 05       MOV  [DI],AL
076B:0016  46          INC  SI
076B:0017  47          INC  DI
076B:0018  FE C9       DEC  CL
076B:001A  75 F8       JNZ  0012
076B:001C  CC          INT  3
```

Conclusion output

AX=0074 BX=0000 CX=0000 DX=0000

076B:001C CC INT 3

Data segment output

-d ds:00

076A:0000 61 69 6D 6C 73 6E 69

73 - 74 61 69 6D 6C 73

6E 69

aimlsnist aimlsnist

Program-25 :-

Write an ALP to concatenate two strings

```
Assume cs:code, ds:data
Data segment
 str1 db 'AIML'
 str2 db 'SNIST'
 str3 db 00H
Data ends
 code segment
 start:
      mov AX,data
      mov DS,AX
      mov AX,0H
      mov CL,09H
    lea SI,str1
    lea DI,str3
    GO:
      mov AL,[SI]
      mov [DI],AL
      INC SI
      INC DI
      DEC CL
      JNZ GO
  INT 03H
  Code ends
  end start
```

Output :-

```
-u
076B:0000  B86A07    MOV   AX,076A
076B:0003  8ED8      MOV   DS,AX
076B:0005  B80000    MOV   AX,0000
076B:0008  B104      MOV   CL,04
076B:000A  8D36000   LEA   SI,[0000]
076B:000E  8D3E0800  LEA   DI,[0008]
076B:0012  8A04      MOV   AL,[SI]
076B:0014  8805      MOV   AL,[SI]
076B:0016  46        INC   SI
076B:0017  4F        DEC   DI
076B:0018  FEC9      DEC   CL
076B:001A  75F6      JNZ   0012
076B:001C  CC        INT   3

-g

AX=004C  BX=0000  CX=0000  DX=0000

-d ds:00

076A:0000  41 49 4D 4C 00 4C 4D 49-41

AIML.LMIA  → taking
             Str1 db 'AIML'
             CL, 04H
             DI, Str1+8
```

Program – 26

Write an ALP to find Reverse of a string

```
Assume CS:code, ds:data
data segment
str1 db 'CSE'
Data ends
Code segment
start:
    mov Ax, data
    mov DS, Ax
    mov AX,00H
    mov CL,03H
    Lea SI, str1
    Lea DI, Str1+6
GO:
    mov AL,[SI]
    MOV [DI], AL
    INC SI
    DEC DI
    DEC CL
    JNZ GO
    INT 03H
Code ends
end start
```

Olp:- CSE.ESC

output

-u

```
076C:0000  B86A07    MOV   AX,076A
076C:0003  8ED8      MOV   DS,AX
076C:0005  8D360000  LEA   SI,[0000]
076C:0009  B86B07    MOV   AX,076B
076C:000C  8EC0      MOV   ES,AX
076C:000E  8D360000  LEA   SI,[0000]
076C:0012  F3        REPZ
076C:0013  A6        CMPSB
076C:0014  7404      JZ    001A
076C:0016  BB1111    MOV   BX,1111
076C:0019  CC        INT   3
076C:001A  BB0000    MOV   BX,0000
076C:001D  CC        INT   3
```

-g

```
AX=076B  BX=1111  CX=0038  DX=0000
SP=0000  BP=0000  SI=0006  DI=0006
DS=076A  ES=076B  SS=0769  CS=076C
IP=0019  NV UP EI PL NZ NA PE NC
076C:0019  CC  INT 3
```

-d ds:00

```
076A:0000  4D 49 43 52 4F 05 00 00
```

Program - 27

Write an ALP to write compare two strings

```
Assume cs:code, ds:data, es:extra
Data segment
str1 db 'MICRO'
count db 05H
Data ends
Extra segment
str2 db 'MACRO'
Extra ends
Code segment
start : mov Ax,data
        mov DS,Ax
        lea SI, str1

        mov Ax,extra
        mov ES,Ax
        lea DI, str2
        mov CX,05H
        Repe cmpsb
        JNZ L1
        mov Bx,1111H
        INT 03H
L1 : mov Bx,0000H
     INT 03H
Code ends
end start
```

Repe
- Repeat equal until zero

Jz
- Jump if zero

correct - 0000 wrong = 1111

Output

```
-u
076B:0000  B86A07    MOV  AX,076A
076B:0003  8ED8      MOV  DS,AX
076B:0005  8EC0      MOV  ES,AX
076B:0007  B000      MOV  AL,00
076B:0009  8D3E0000  LEA  DI,[0000]
076B:000D  FC        CLD
076B:000E  F2        REPNZ
076B:000F  AE        SCASB
076B:0010  4F        DEC  DI
076B:0011  893E0B00  MOV  [000B],DI
076B:0015  CC        INT  3

-g

AX=0700  BX=0000  CX=001B  DX=0000
SP=0000  BP=0000  SI=0000  DI=000A
DS=076A  ES=076A  SS=0769  CS=076B
IP=0015  NV UP EI PL NZ NA PE NC
076B:0015  CC  INT 3

-d ds:00

076A:0000  63 6F 6E 74 6F 6C 6C
           65 72 00 0A
```

controller . . . . . .

Program - 28

Write an ALP to find the length of the given string.

```
Assume CS:code, ds:data
data segment
str1 db 'controller', 0   ; zero
res dw 00H
Data ends
code segment
 start :
        mov Ax, data
        mov DS, Ax
        mov ES, AX
        mov AL, 00H
        lea DI, str1
        CLD
        Repne scasb
        DEC DI
        mov res, DI
        INT 03H
        code ends
        end start
```

CLD - clear direction flag

Repne - Repeat not equal

Scasb - scanbyte

length = 10

↓

Hexadecimal - A

DI = 000A

Output

```
-u
076B:0000   B86A07    MOV   AX,076A
076B:0003   8ED8      MOV   DS,AX
076B:0005   8ECD      lea   SI,[0000]
076B:0007   B000      MOV   AL,00
076B:0009   8D3E0000  Lea   DI,[0000]
076B:000D   FC        CLD
076B:000E   F2        REPNZ
076B:000F   AE        SCASB
076B:0010   7404      JZ    0016
076B:0012   BBFF00    MOV   BX,00FF
076B:0015   CC        INT   3
076B:0016   BB0000    MOV   BX,0000
076B:0019   CC        INT   3

-g

AX=0700  BX=0000  CX=001F  DX=0000
SP=0000  SI=0000  DI=000B

-d ds:00

076A:0000  63 6F 6E 74 72 6F
           6C 6C 65 72 00 00-

                    Controller.
```

Program-29

Write an ALP to find the given string or character, string or not ?

```
Assume cs:code, ds:data
Data segment
 Str1 db 'controller', 0-zero
 res  dw 00H
Data ends
code segment
 start:
      MOV Ax,data
      mov DS,Ax
      mov ES,Ax
      MOV AL,00H
      lea DI,str1
      CLD
      Repne scasb
      JZ  L1
      MOV BX,00FFH
       INT 03H
 L1 : MOV BX,0000H
      INT 03H
 Code ends
 end start
```