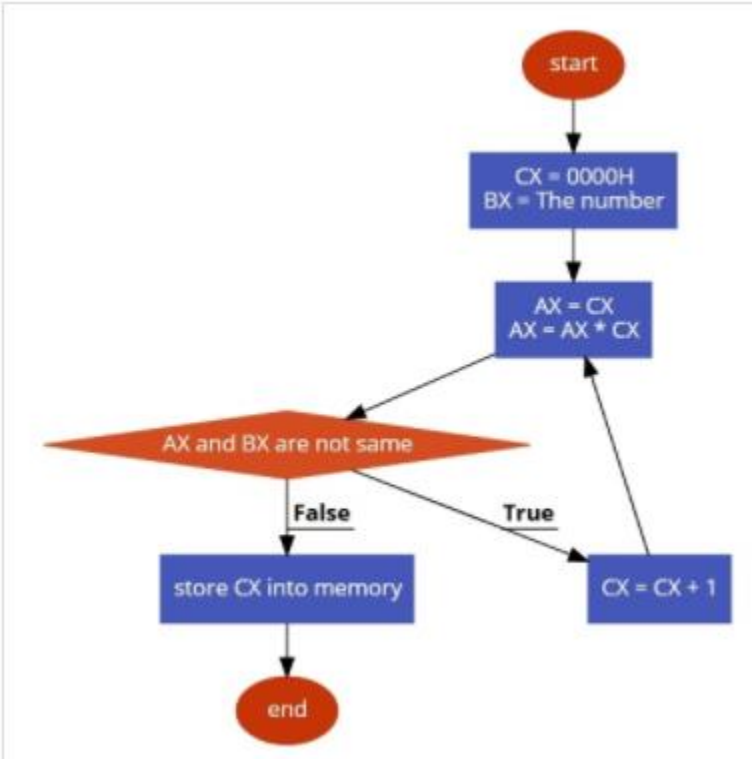
Important Topics

- ✓ Verify the square root of a perfect squared number among number
- ✓ Generate and verify GP series
- ✓ To generate AP series
- ✓ Addition and average of a series
- ✓ To arrange a given series of hexadecimal bytes (67, 25, 14, and 02) in descending order
- ✓ To arrange a given series of hexadecimal bytes (67, 25, 14, and 02) in ascending order
- ✓ Assembly language program to generate Fibonacci sequence
- ✓ To convert into the following number system.
  - Binary to 2s Complement
  - BCD to binary
  - Binary to BCD
- ✓ Program to find and display the factorial of a number stored in memory offset
- ✓ Largest and smallest number in an array
- ✓ Write and verify 8086 assembly language program that will perform following string operation for a string: "This is a microprocessor and interfacing lab exam"
  - i. Calculate length of a string
  - ii. Count number of spaces in a string
  - iii. Reverse the given string and print the reversed string

1) SQRT:



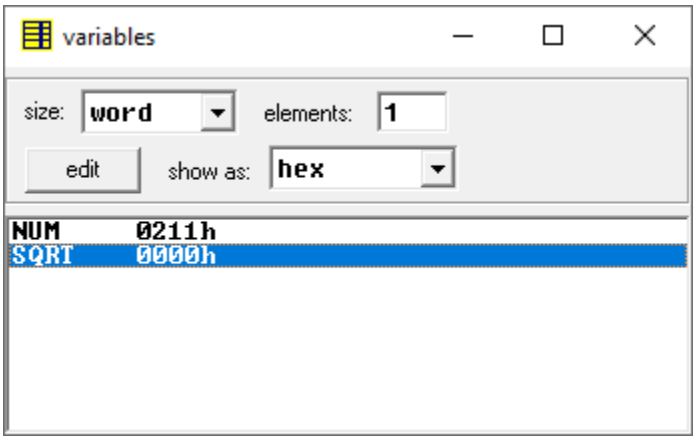
Code

```
Assume CS: Code DS: Data
DATA SEGMENT

   NUM DW 211H
   SQRT DW ?

DATA ENDS
CODE SEGMENT
   START:
   MOV AX, @DATA
   MOV DS, AX
   MOV CX,0000H
   MOV BX, Num
 L1:MOV AX,CX
   MUL CX
   CMP AX, BX
   JZ STORE
   INC CX
   JNZ L1
STORE:MOV DX,CX
    MOV SQRT,DX
    HLT
    CODE ENDS
```

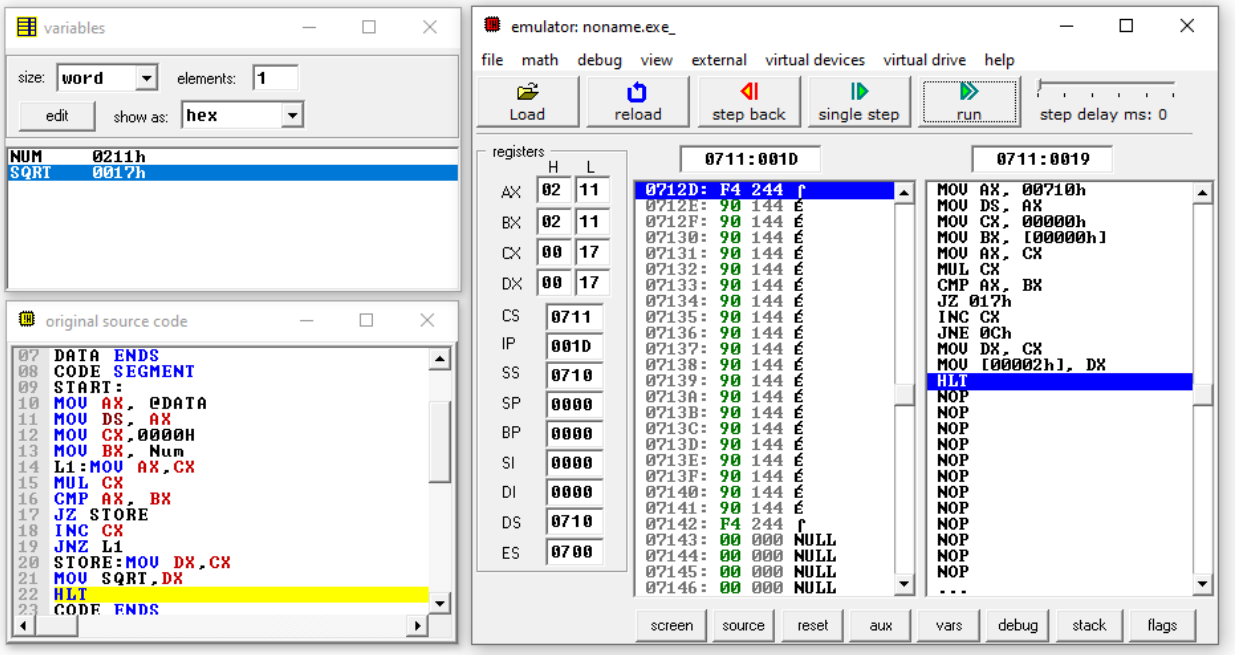
END START

```
01 Assume CS: Code DS: Data
02 DATA SEGMENT
03
04     NUM DW 211H
05     SQRT DW ?
06
07 DATA ENDS
08 CODE SEGMENT
09     START:
10     MOV AX, @DATA
11     MOV DS, AX
12     MOV CX,0000H
13     MOV BX, Num
14  L1:MOV AX,CX
15     MUL CX
16     CMP AX, BX
17     JZ STORE
18     INC CX
19     JNZ L1
20 STORE:MOV DX,CX
21       MOV SQRT,DX
22       HLT
23       CODE ENDS
24 END START
25
```

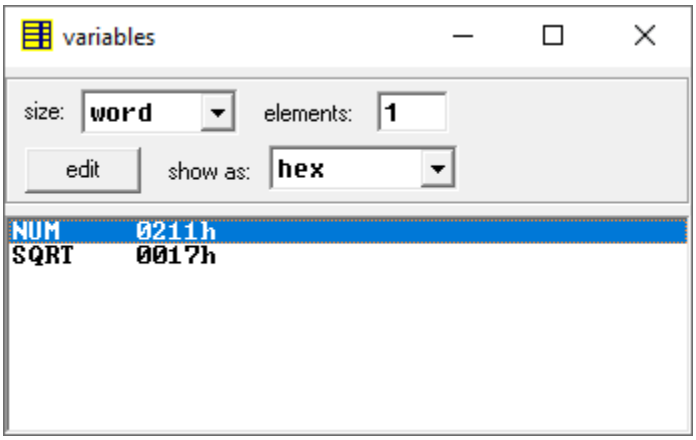
Before Exe



OUTPUT:



After Exe



2)GP

Code:

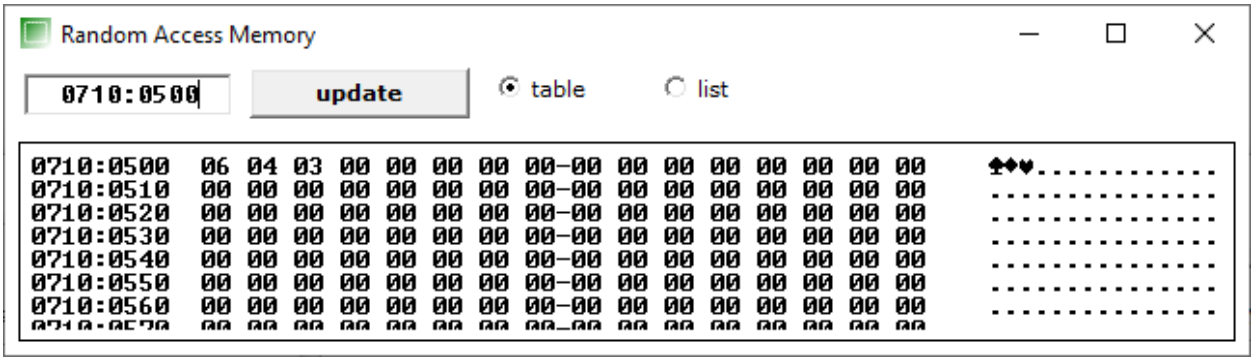
```
Assume CS: Code DS: Data
DATA SEGMENT

    Input DW 0500H
    Output DW 0600H

DATA ENDS
CODE SEGMENT
    START:
    MOV AX, @DATA
    MOV DS, AX
    MOV SI, Input
    MOV CL, [SI]
```

```
    MOV CH, 00H
    INC SI
    MOV AL, [SI]
    MOV AH, 00H
    MOV DI, Output
    MOV [DI], AL
    INC DI
    DEC CL
    INC SI
    MOV BL, [SI]
 L1:MUL  BL
    MOV [DI], AL
    INC DI
    LOOP L1
    HLT
    CODE ENDS
END START
```

```
01  Assume CS: Code DS: Data
02  DATA SEGMENT
03
04      Input DW 0500H
05      Output DW 0600H
06
07  DATA ENDS
08  CODE SEGMENT
09      START:
10      MOV AX, @DATA
11      MOV DS, AX
12      MOV SI, Input
13      MOV CL, [SI]
14      MOV CH, 00H
15      INC SI
16      MOV AL, [SI]
17      MOV AH, 00H
18      MOV DI, Output
19      MOV [DI], AL
20      INC DI
21      DEC CL
22      INC SI
23      MOV BL, [SI]
24   L1:MUL  BL
25      MOV [DI], AL
26      INC DI
27      LOOP L1
28      HLT
29      CODE ENDS
30  END START
31
32
```

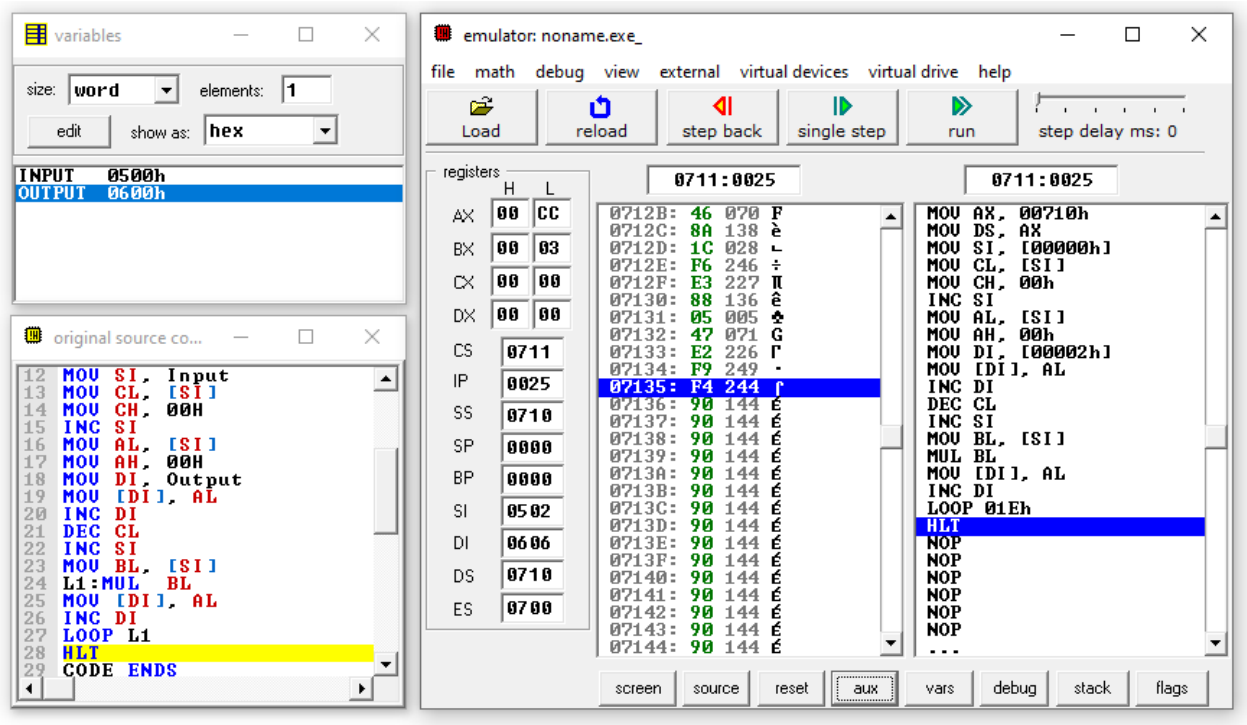
Input:

Before Execution

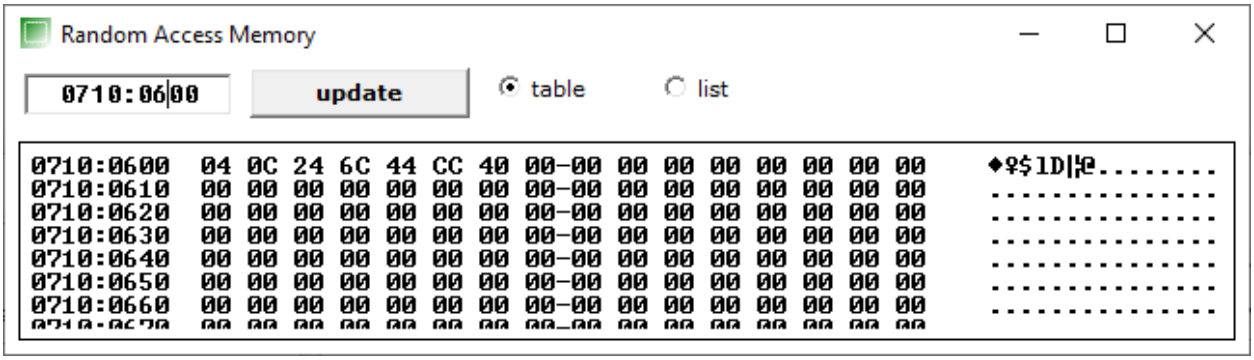


Given input (Number of terms) n=06h in 0710:0500h and first term a=04h in 0710:0501h and common ratio r =03h in 0710:0502h

Output:

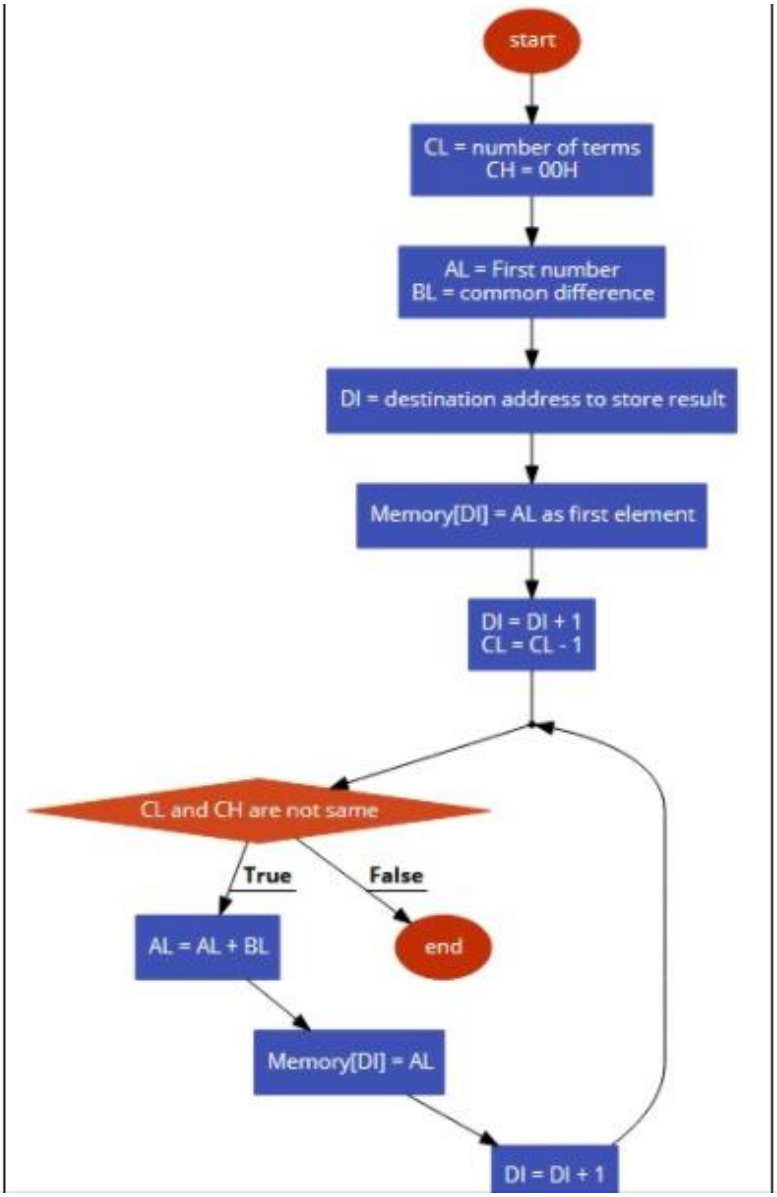


After Execution:



GP Series obtained starting from 600 location

3)AP



CODE:

Assume CS: Code DS: Data

DATA SEGMENT

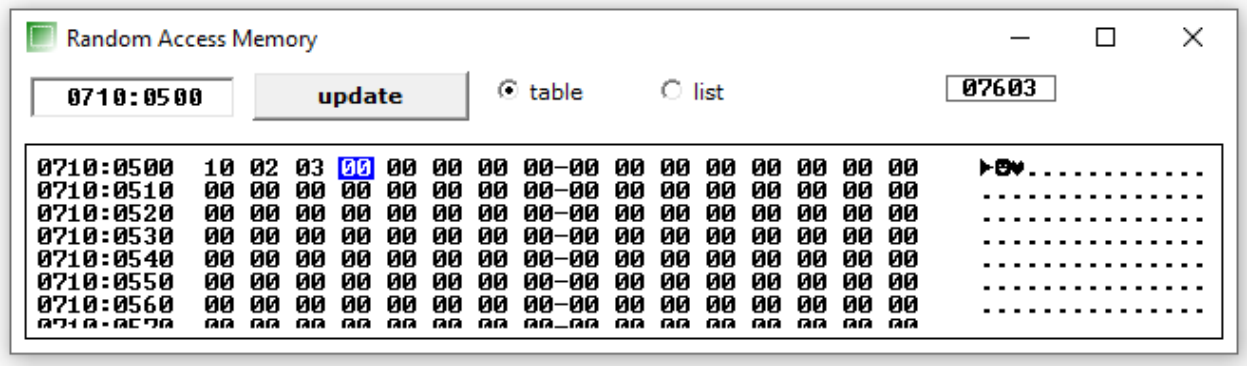
```
    Input DW 0500H
    Output DW 0600H

DATA ENDS
CODE SEGMENT
    START:
    MOV AX, @DATA
    MOV DS, AX
    MOV SI, Input
    MOV CL, [SI]
    MOV CH, 00H
    INC SI
    MOV AL, [SI]
    MOV AH, 00H
    MOV DI, Output
    MOV [DI], AL
    INC DI
    DEC CL
```

```
    INC SI
    MOV BL, [SI]
 L1:ADD AL, BL
    MOV [DI], AL
    INC DI
    LOOP L1
    HLT
    CODE ENDS
END START
```

```
01 Assume CS: Code DS: Data
02 DATA SEGMENT
03
04     Input DW 0500H
05     Output DW 0600H
06
07 DATA ENDS
08 CODE SEGMENT
09     START:
10     MOV AX, @DATA
11     MOV DS, AX
12     MOV SI, Input
13     MOV CL, [SI]
14     MOV CH, 00H
15     INC SI
16     MOV AL, [SI]
17     MOV AH, 00H
18     MOV DI, Output
19     MOV [DI], AL
20     INC DI
21     DEC CL
22     INC SI
23     MOV BL, [SI]
24  L1:ADD AL, BL
25     MOV [DI], AL
26     INC DI
27     LOOP L1
28     HLT
29     CODE ENDS
30 END START
31
32
```

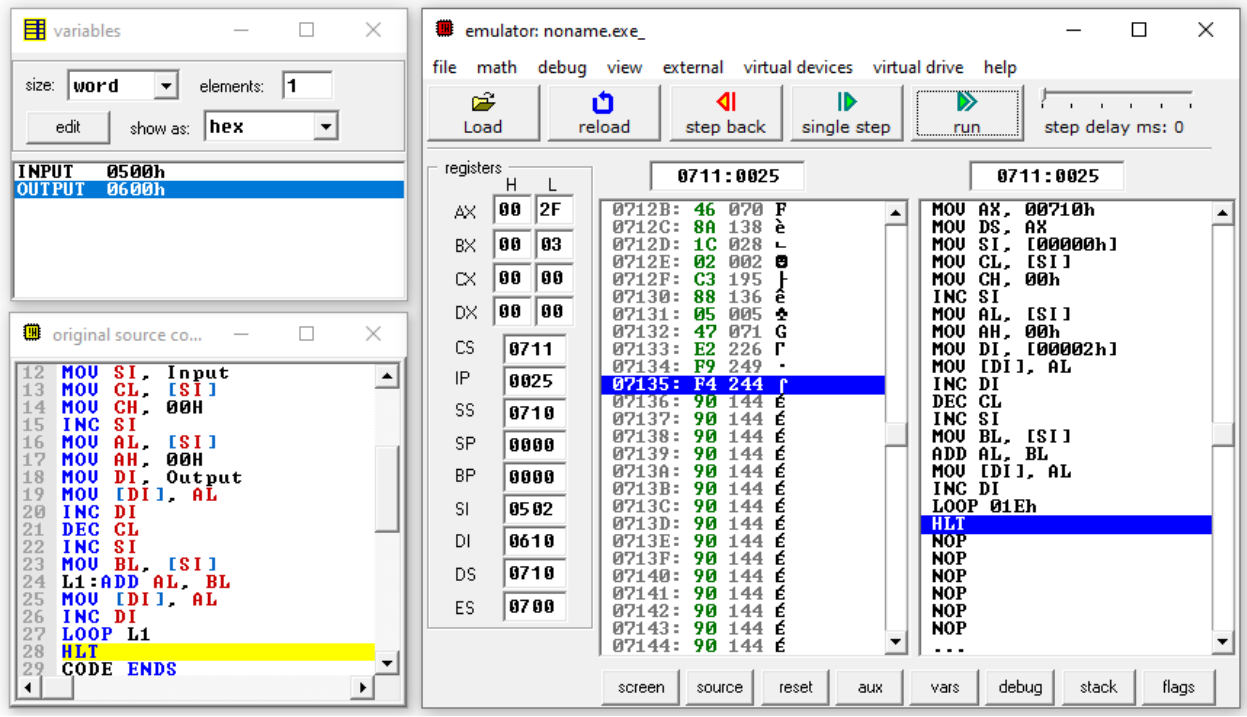
Input:

Before Execution

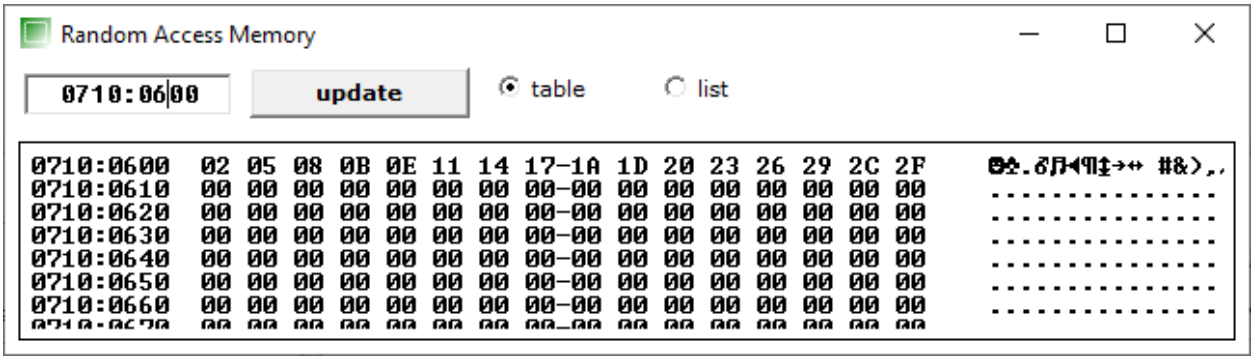


Given input Number of Terms n=10h in 0710:0500h and first term a=02h in 0710:0501h and common difference d=03h in 0710:0502h

Output:



After Execution:



AP Series obtain starting from 600 location.

4)ADDITION AND AVERAGE OF A SERIES

Code

```
Assume CS: Code DS: Data
DATA SEGMENT

    InputLoc DW 0500H
    SIZE DB 6h
    SUM DW ?
    AVG DW ?
DATA ENDS
CODE SEGMENT
    START:
    MOV AX, @DATA
    MOV DS, AX
    MOV SI, InputLoc
```

```
    MOV CL, size
    MOV CH, 00H
    MOV AL, [SI]
    MOV AH, 00H
    DEC CL
    INC SI
L1:MOV BL,[SI]
    MOV BH,00H
    ADD AX,BX
    INC SI
    LOOP L1
    MOV SUM,AX
    MOV SUM,AX
    MOV BL, SIZE
    DIV BL
    MOV AVG,AX
    HLT
    CODE ENDS
```

END START

edit: C:\emu8086\MySource\Avg and Sum of series.asm

file edit bookmarks assembler emulator math ascii codes help

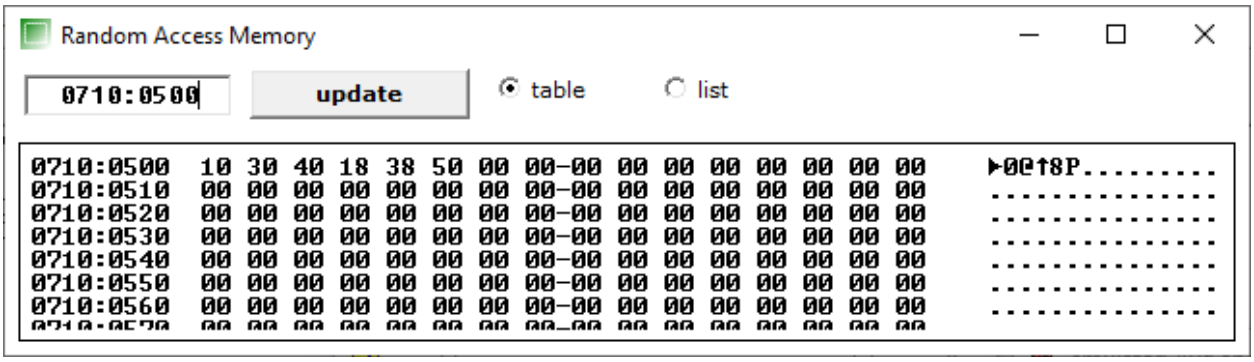
new open examples save compile emulate calculator convertor options help

```
01 Assume CS: Code DS: Data
02 DATA SEGMENT
03
04     InputLoc DW 0500H
05     SIZE DB 6h
06     SUM DW ?
07     AVG DW ?
08 DATA ENDS
09 CODE SEGMENT
10     START:
11     MOV AX, @DATA
12     MOV DS, AX
13     MOV SI, InputLoc
14     MOV CL, size
15     MOV CH, 00H
16     MOV AL, [SI]
17     MOV AH, 00H
18     DEC CL
19     INC SI
20  L1:MOV BL,[SI]
21     MOV BH,00H
22     ADD AX,BX
23     INC SI
24     LOOP L1
25     MOV SUM,AX
26     MOV SUM,AX
27     MOV BL, SIZE
28     DIV BL
29     MOV AVG,AX
30     HLT
31     CODE ENDS
32 END START
33
```

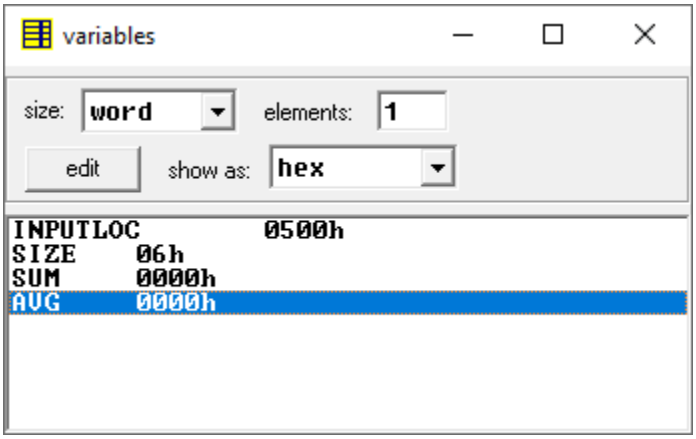
Input:

Before Execution

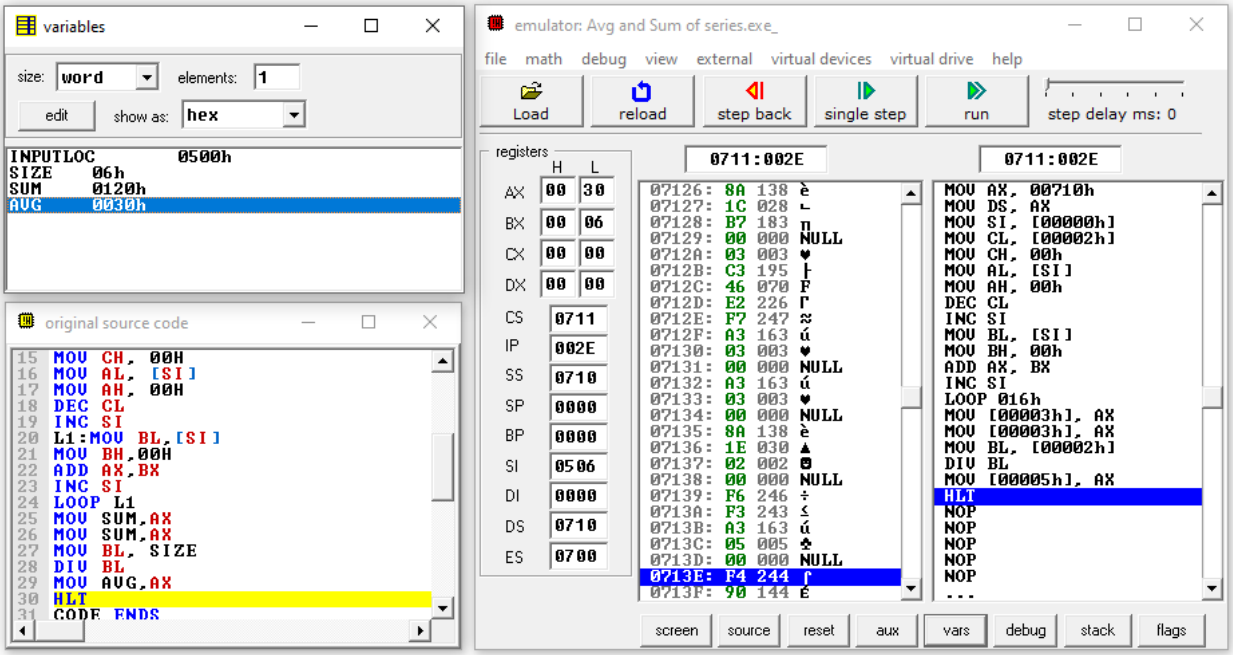
Memory



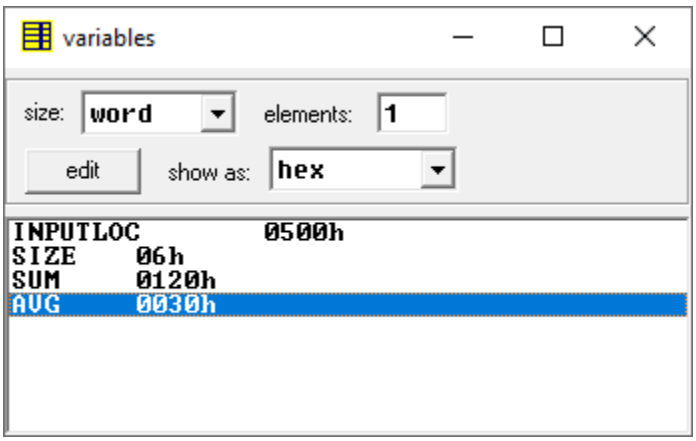
Variables



OUTPUT



Variables


variables
size:
word
elements:
1
edit
show as:
hex
INPUTLOC 0500h
SIZE 06h
SUM 0120h
AVG 0030h

5)DESCENDING

→ We can follow the same Procedure as in Ascending order

Only 1 line of code will change i.e.

In Ascending code

CMP AL,BL

In Descending code

CMP BL,AL

CODE

```
DATA SEGMENT
Array DB 57H,10H,56H,25H,32H,35H
SIZE DB 6H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA
```

```
START: MOV AX,DATA
MOV DS,AX

MOV CH,SIZE
DEC CH

UP2: MOV CL,SIZE
DEC CL
LEA SI,Array

UP1: MOV AL,[SI]
MOV BL,[SI+1]
CMP BL,AL
JC DOWN
MOV DL,[SI+1]
XCHG [SI],DL
MOV [SI+1],DL

DOWN: INC SI
```

DEC CL

JNZ UP1

DEC CH

JNZ UP2

INT 3

CODE ENDS

END START

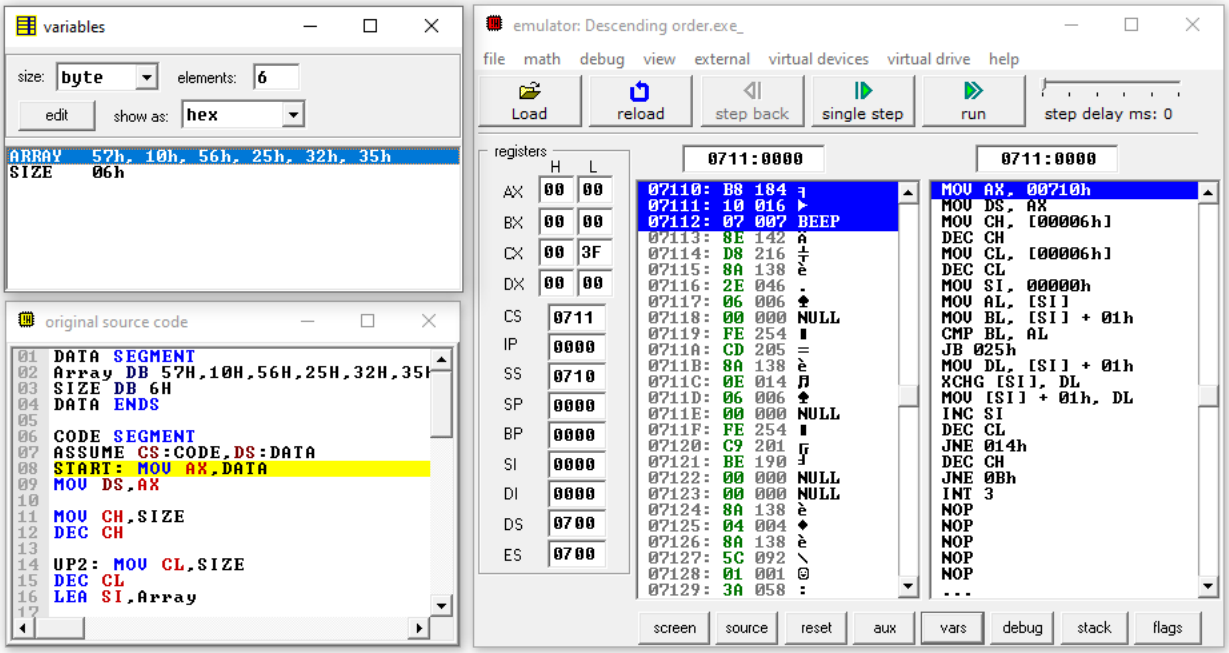
edit: C:\emu8086\MySource\Descending order.asm

file edit bookmarks assembler emulator math ascii codes help

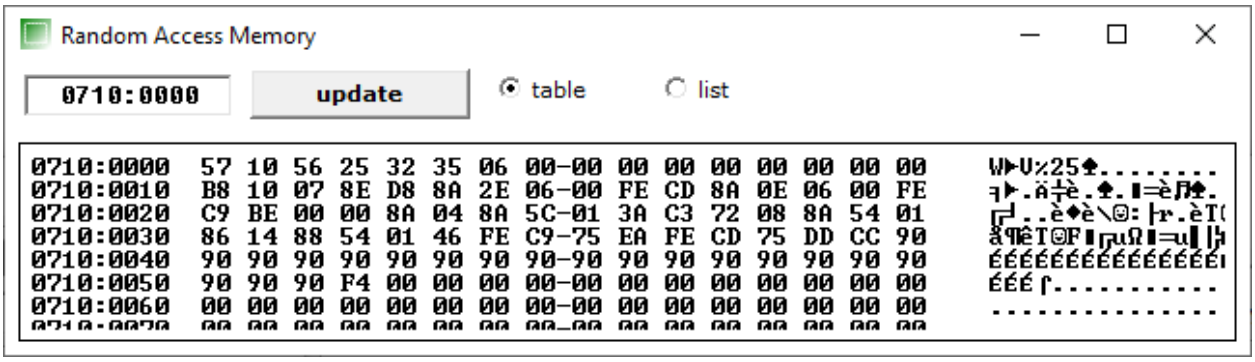
new open examples save compile emulate calculator convertor options help about

```
01 DATA SEGMENT
02 Array DB 57H,10H,56H,25H,32H,35H
03 SIZE DB 6H
04 DATA ENDS
05
06 CODE SEGMENT
07 ASSUME CS:CODE,DS:DATA
08 START: MOV AX,DATA
09 MOV DS,AX
10
11 MOV CH,SIZE
12 DEC CH
13
14 UP2: MOV CL,SIZE
15 DEC CL
16 LEA SI,Array
17
18 UP1: MOV AL,[SI]
19 MOV BL,[SI+1]
20 CMP BL,AL
21 JC DOWN
22 MOV DL,[SI+1]
23 XCHG [SI],DL
24 MOV [SI+1],DL
25
26 DOWN: INC SI
27 DEC CL
28 JNZ UP1
29 DEC CH
30 JNZ UP2
31
32 INT 3
33 CODE ENDS
34 END START
35
```

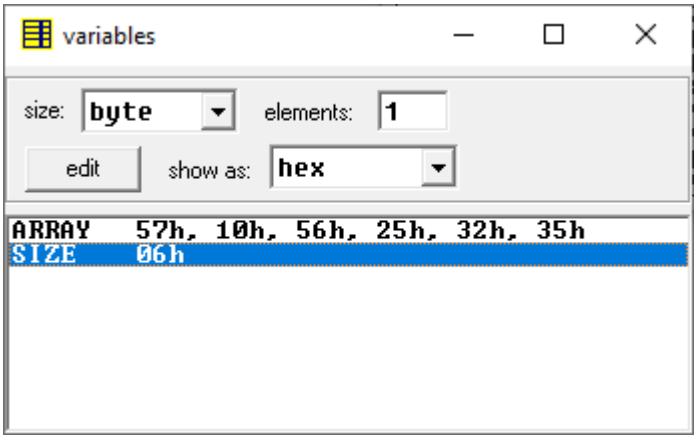
Before Execution:



Memory Location

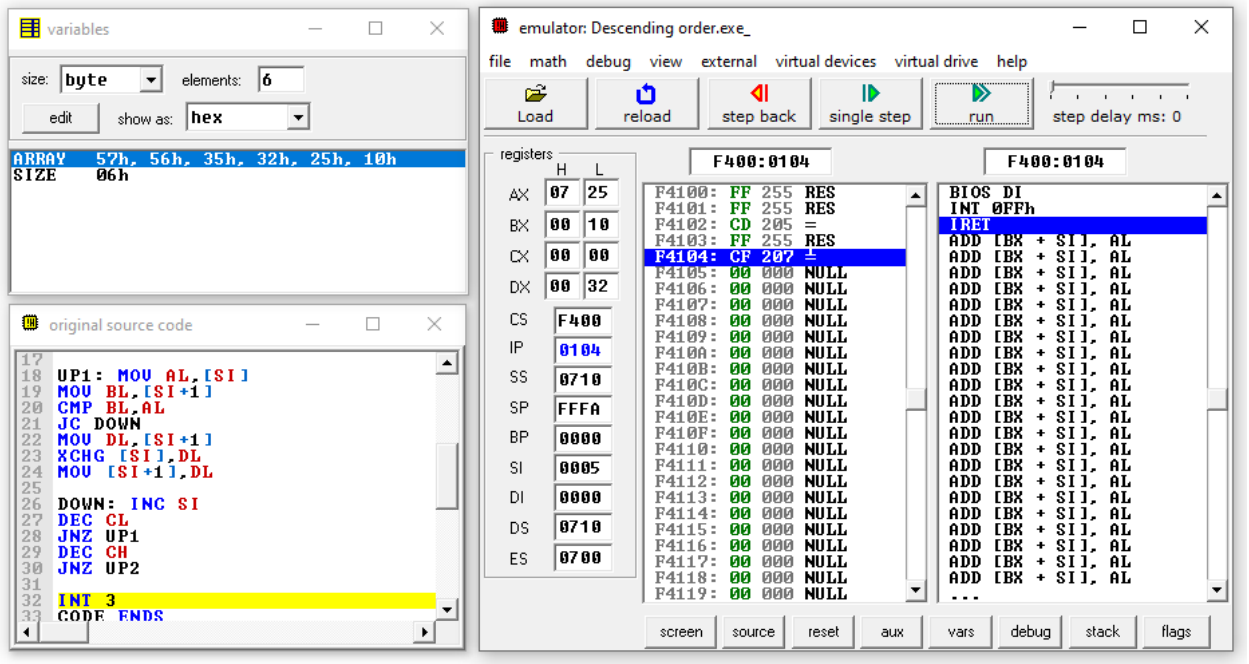


Variables

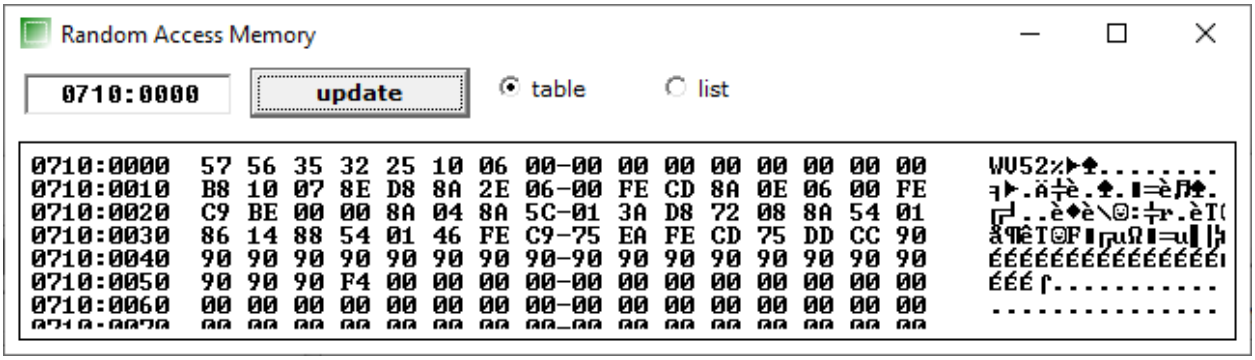


OUTPUT:

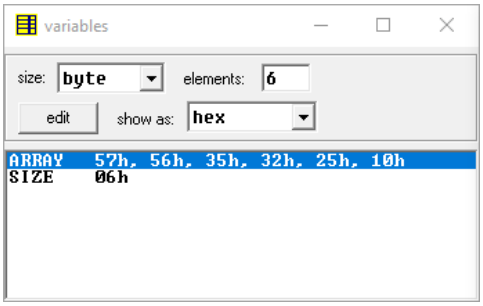
After Execution:



Memory Location



Variables



6)ASCENDING

Copy Algo, Flowchart, design from the lab doc change Calculations an below SS

Lab DA6 →Experiment 8.1 →Question 1

CODE

```
DATA SEGMENT
Array DB 57H,10H,56H,25H,32H,35H
SIZE DB 6H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX

MOV CH,SIZE
DEC CH
```

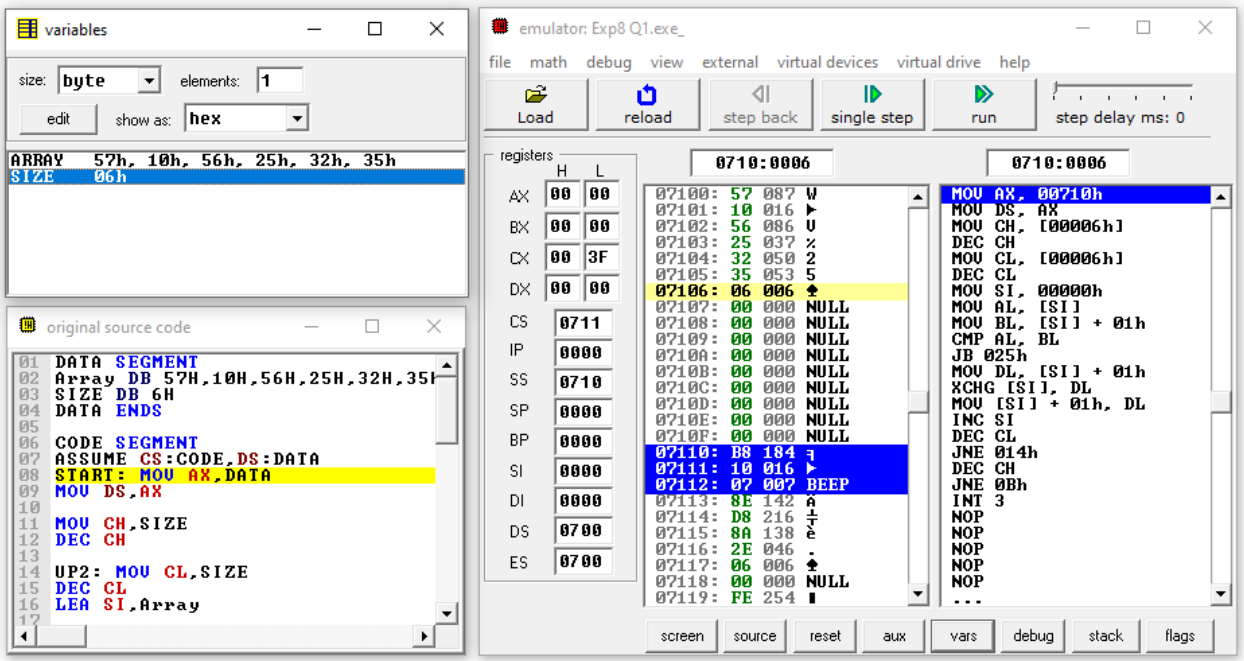
```
UP2: MOV CL,SIZE
DEC CL
LEA SI,Array

UP1: MOV AL,[SI]
MOV BL,[SI+1]
CMP AL,BL
JC DOWN
MOV DL,[SI+1]
XCHG [SI],DL
MOV [SI+1],DL

DOWN: INC SI
DEC CL
JNZ UP1
DEC CH
JNZ UP2
 INT 3
```

CODE ENDS

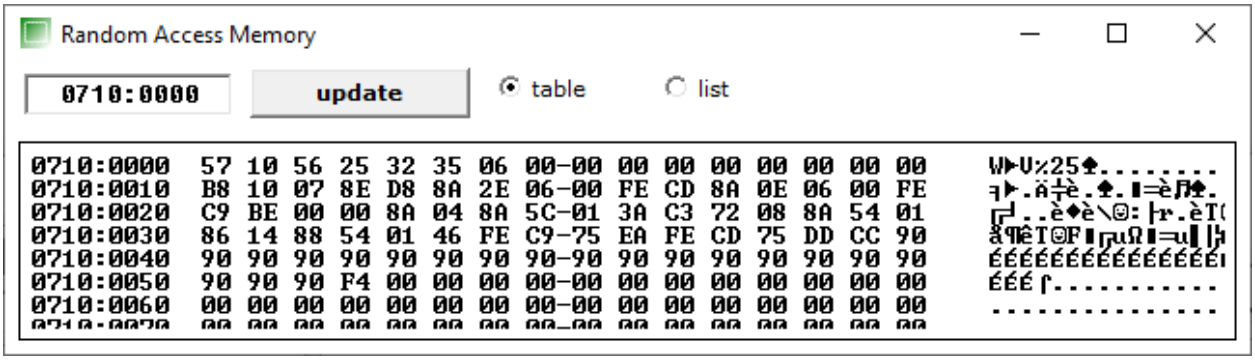
END START

```
01 DATA SEGMENT
02 Array DB 57H,10H,56H,25H,32H,35H
03 SIZE DB 6H
04 DATA ENDS
05
06 CODE SEGMENT
07 ASSUME CS:CODE,DS:DATA
08 START: MOV AX,DATA
09 MOV DS,AX
10
11 MOV CH,SIZE
12 DEC CH
13
14 UP2: MOV CL,SIZE
15 DEC CL
16 LEA SI,Array
17
18 UP1: MOV AL,[SI]
19 MOV BL,[SI+1]
20 CMP AL,BL
21 JC DOWN
22 MOV DL,[SI+1]
23 XCHG [SI],DL
24 MOV [SI+1],DL
25
26 DOWN: INC SI
27 DEC CL
28 JNZ UP1
29 DEC CH
30 JNZ UP2
31
32 INT 3
33 CODE ENDS
34 END START
35
```

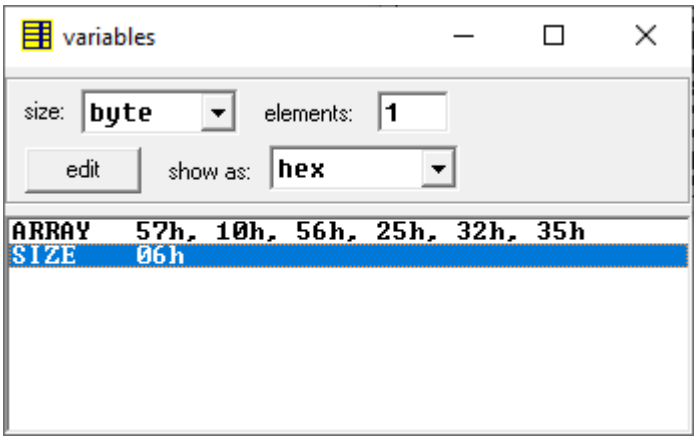
Before Execution:



Memory Location

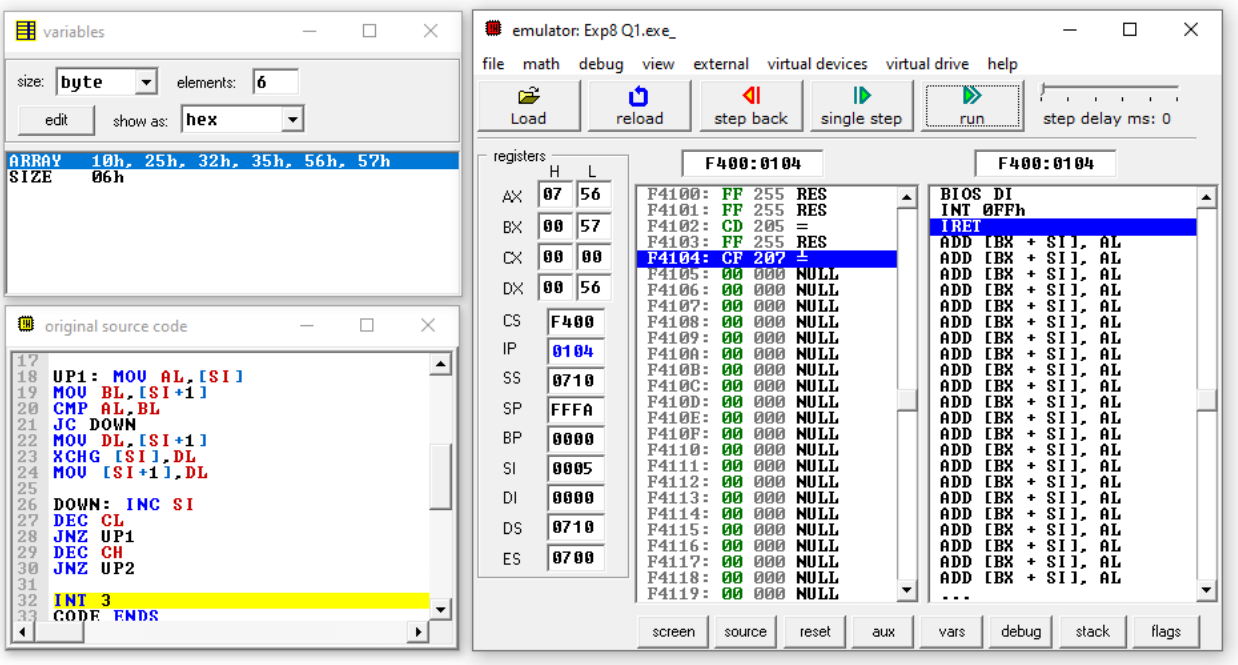


Variables

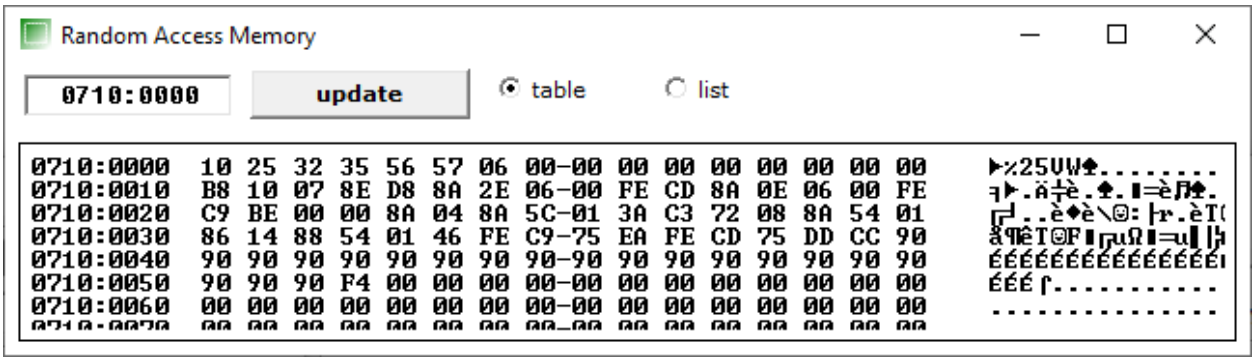


OUTPUT:

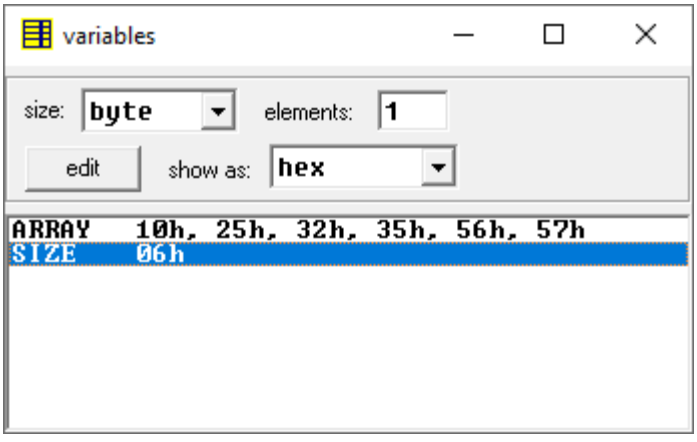
After Execution:



Memory Location



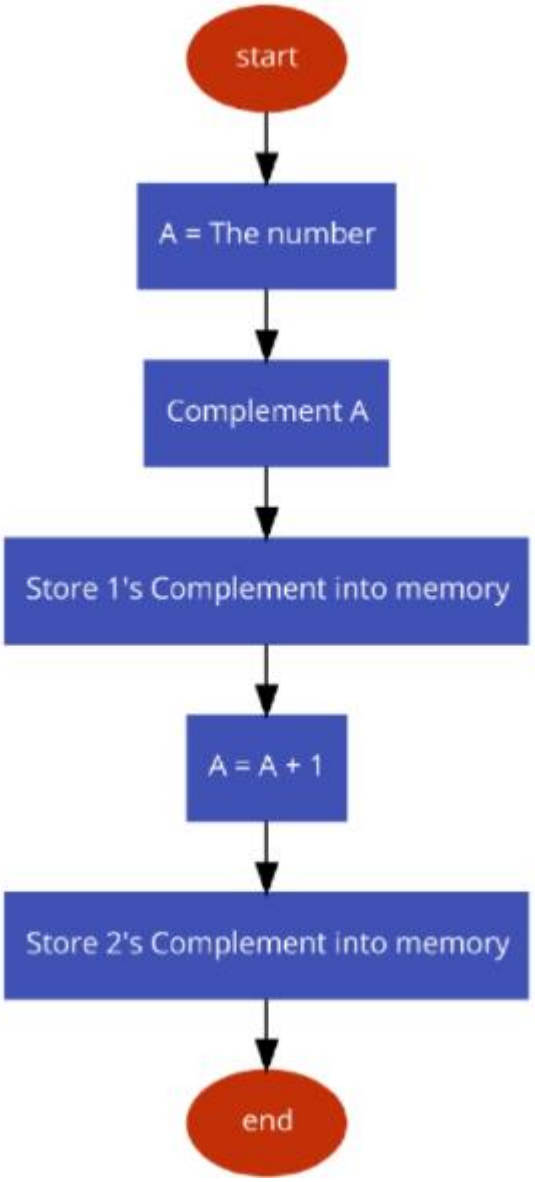
Variables



7)FIBANOCCI SERIES

Lab DA5 → Experiment 6 → Question 2 →2nd part

8 A)BINARY TO 2'S COMP



Code

Data Segment

bin db 00000010B

2comp db ?

```
Data Ends

Code Segment
  Assume cs:code, ds:data

  Begin:
    mov ax, data
    mov ds, ax
    mov ah, 0000h
    mov al, bin
    NOT al
    mov bl, al
    adc al, 00000001B
    mov bl, al

  Exit:
    mov 2comp,al
```

mov ax, 4c00h

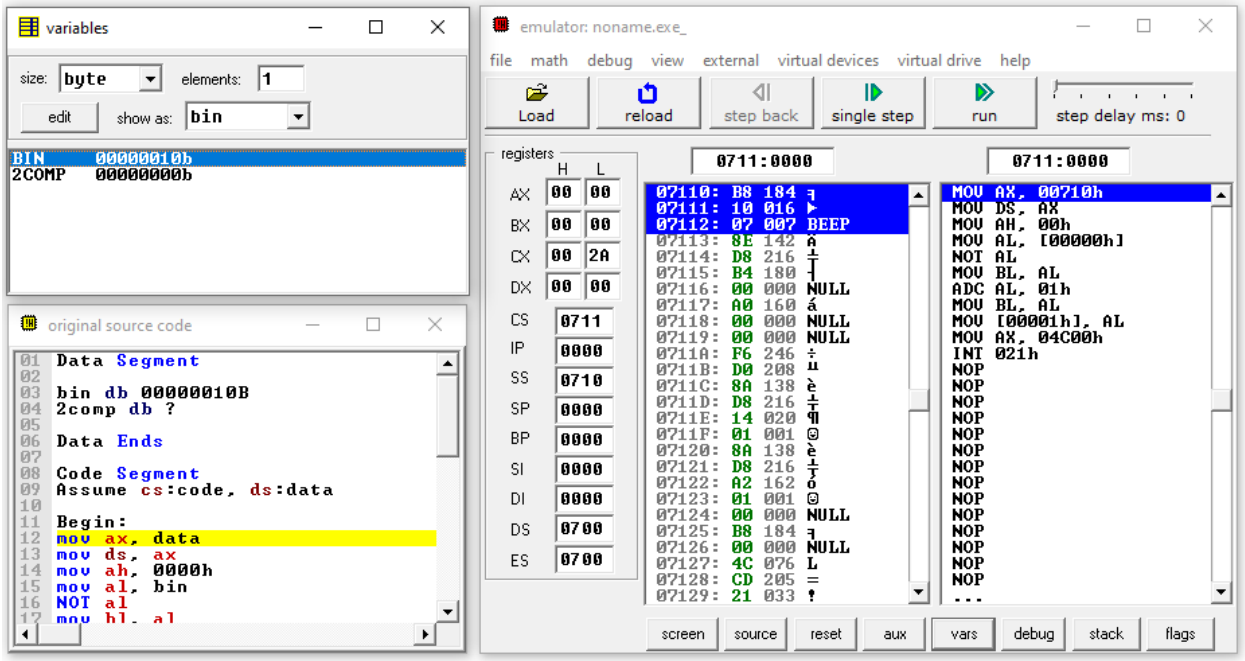
int 21h

Code Ends

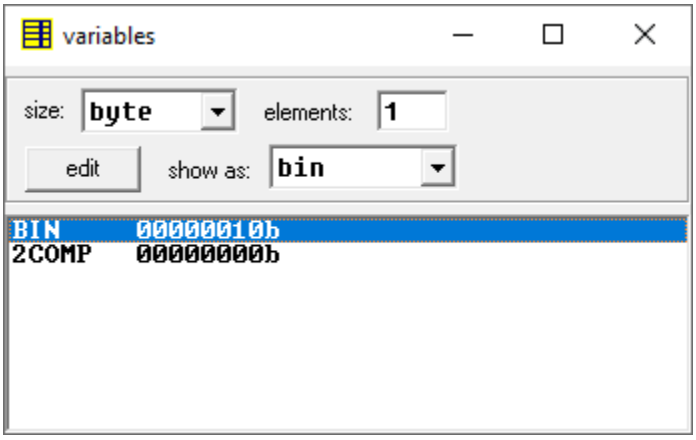
End Begin

```
01 Data Segment
02
03   bin db 00000010B
04   2comp db ?
05
06 Data Ends
07
08 Code Segment
09   Assume cs:code, ds:data
10
11   Begin:
12     mov ax, data
13     mov ds, ax
14     mov ah, 0000h
15     mov al, bin
16     NOT al
17     mov bl, al
18     adc al, 00000001B
19     mov bl, al
20
21   Exit:
22     mov 2comp,al
23     mov ax, 4c00h
24     int 21h
25 Code Ends
26 End Begin
```

Before Exe

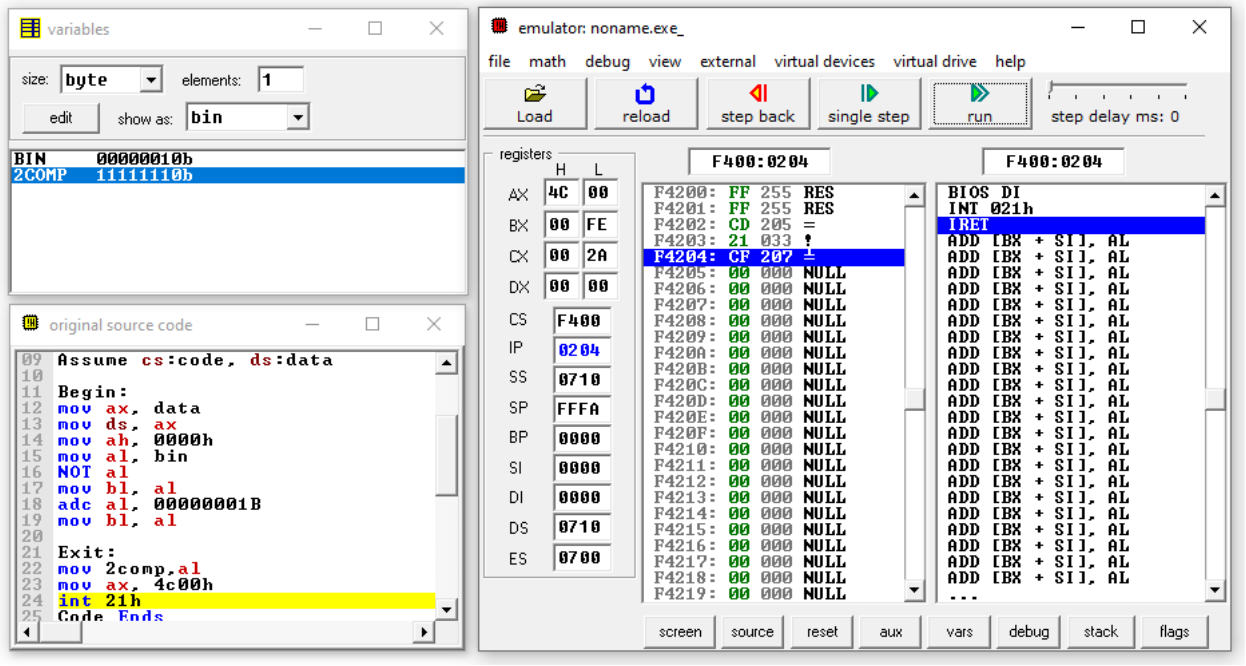


Variables

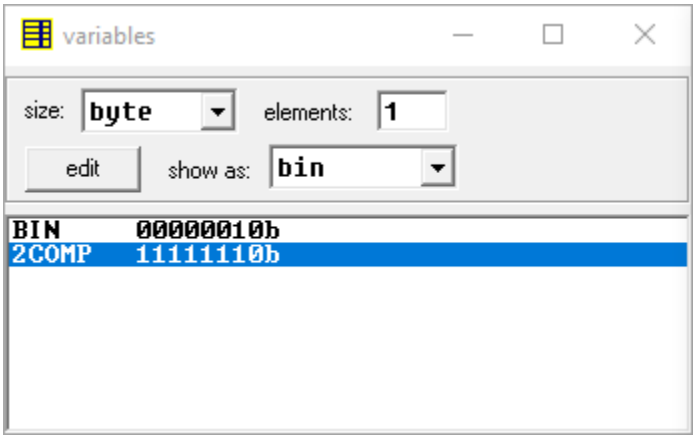


OUTPUT

After Exe



Variables:



8 B)BCD TO BINARY

Lab DA6 →Experiment7 → Question2

8 C)BINARY TO BCD

Lab DA6 →Experiment7 → Question1

9) FACTORIAL OF NUMBER WHOSE INPUT IS IN MEMORY

Lab DA5 →Exp 6→ Question2 → Part1

(Change Memory Location according to Question)

10) LARGEST AND SMALLEST NUMBERS IN ARRAY

CODE

```
data segment
array db 67h,25h,14h,02h
size db 04h
small db 0h
large db 0h
data ends
code segment
start:
    mov ax,data
    mov ds,ax
    mov cl, size
    mov si , offset array
    mov al,[si]
    dec cl
```

```
up1:inc si
    cmp al,[si]
    jc dn1
    mov al,[si]
    jc dn1
dn1:dec cl
    jnz up1
    mov cl, size
    mov si , offset array
    mov bl,[si]
    dec cl
up2:inc si
    cmp [si], bl
    jc dn2
    mov bl,[si]
    jc dn2
dn2:dec cl
    jnz up2
```

mov small ,al

mov large, bl

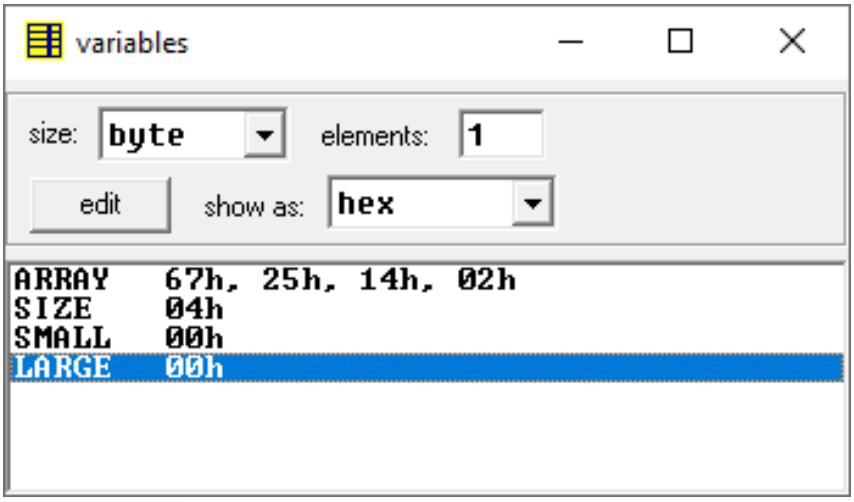
HLT

code ends

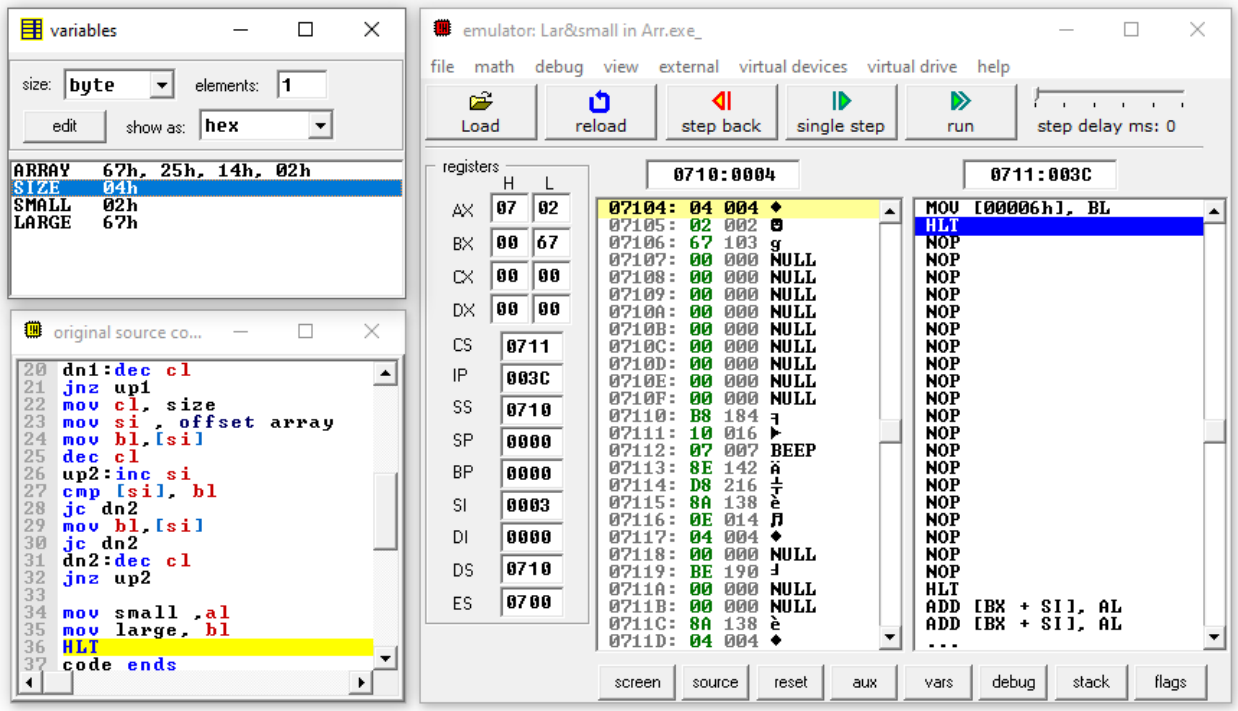
end start

```
01 data segment
02 array db 67h,25h,14h,02h
03 size db 04h
04 small db 0h
05 large db 0h
06 data ends
07 code segment
08 start:
09     mov ax,data
10     mov ds,ax
11     mov cl, size
12     mov si , offset array
13     mov al,[si]
14     dec cl
15  up1:inc si
16      cmp al,[si]
17      jc dn1
18      mov al,[si]
19      jc dn1
20  dn1:dec cl
21      jnz up1
22      mov cl, size
23      mov si , offset array
24      mov bl,[si]
25      dec cl
26  up2:inc si
27      cmp [si], bl
28      jc dn2
29      mov bl,[si]
30      jc dn2
31  dn2:dec cl
32      jnz up2
33
34     mov small ,al
35     mov large, bl
36     HLT
37 code ends
38 end start
```

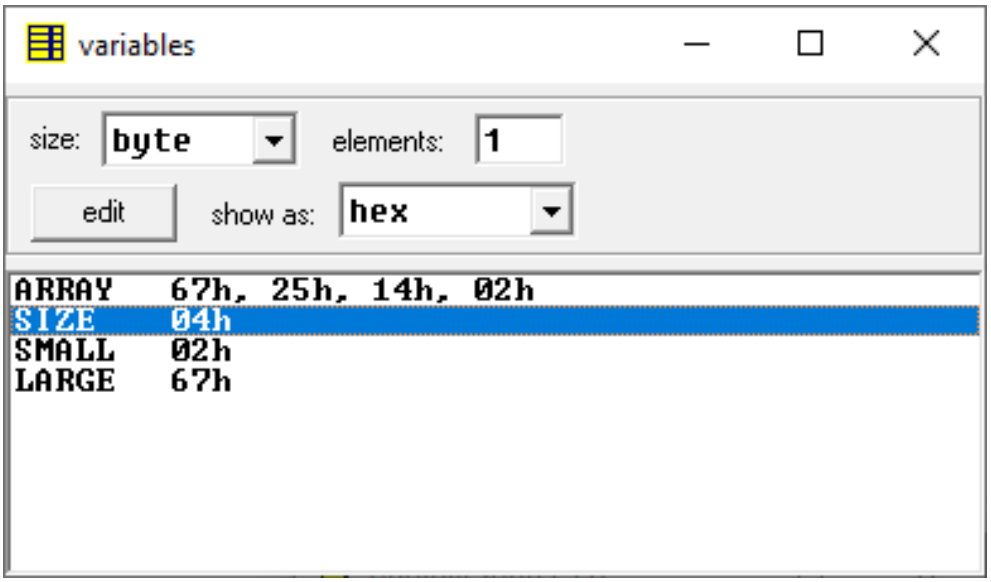
Variables Before Execution



OUTPUT



Variables After Execution



11)

A) CALCULATE LENGTH OF STRING

CODE

```
;macro for printing a string
print macro m
mov ah,09h
mov dx,offset m
int 21h
endm
```

```
.model small

.data

msg db 10,13, "Exiting the program $"

empty db 10,13, "   $"
str1 db 25,?,25 dup('$')

len db ?
mstring db 10,13, "Enter the string: $"
mlength db 10,13, "Length is: $"
;********** Code Segment ************

.code

start:
        mov ax,@data
        mov ds,ax

        print mstring
```

```
        call accept_string   ;function call to accept a
string


    mov cl,str1+1       ;storing length in cl from first
byte of the array
    mov bl,cl           ;copying in bl for displaying
        print mlength
    call display1       ;printing the length


exit:
mov ah,4ch      ;exit the program
int 21h



;accept procedure

accept proc near

mov ah,01
int 21h
ret
accept endp
```

```
display1 proc near

   mov al,bl
   mov bl,al
   and al,0f0h
   mov cl,04
   rol al,cl

   cmp al,09
   jbe number
   add al,07
number:  add al,30h
        mov dl,al
        mov ah,02
        int 21h
        mov al,bl
        and al,00fh
        cmp al,09
        jbe number2
        add al,07
```

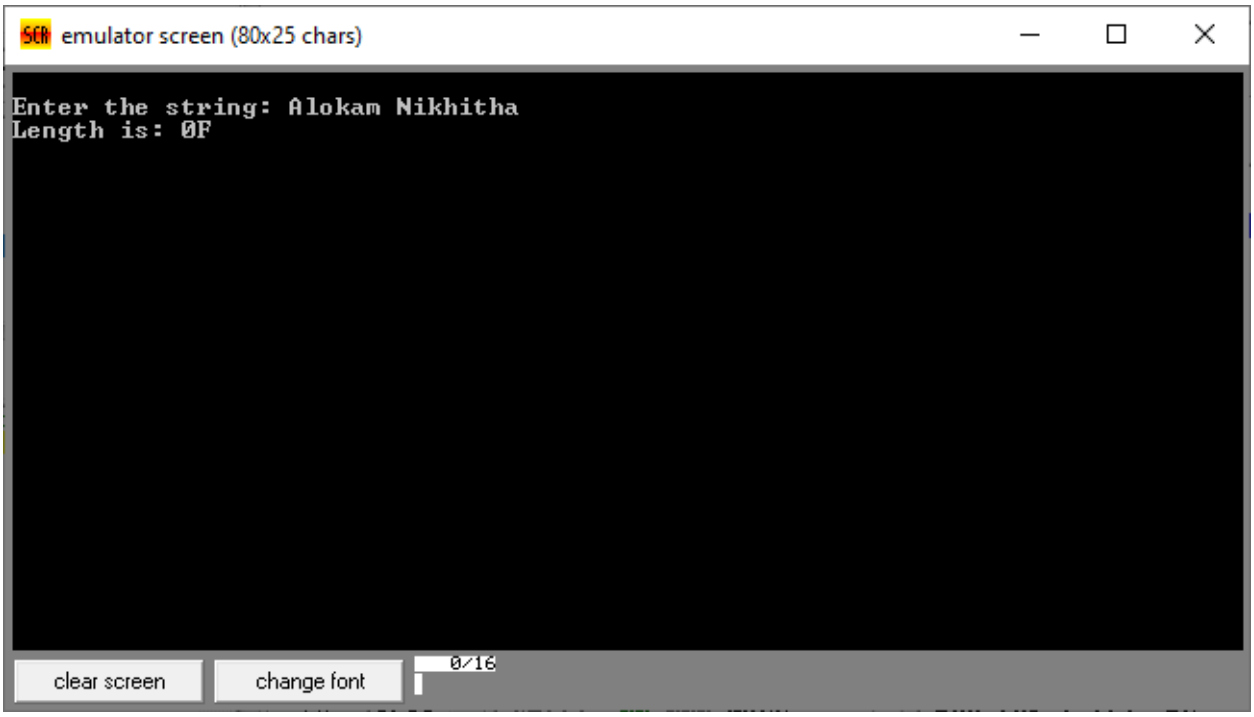
```
number2:  add al,30h
        mov dl,al
        mov ah,02
        int 21h
ret
display1 endp

accept_string proc near
mov ah,0ah          ;accept string from user function
mov dx,offset str1  ; store the string in memory
pointed by "DX"
int 21h
ret
accept_string endp
end start
end
```

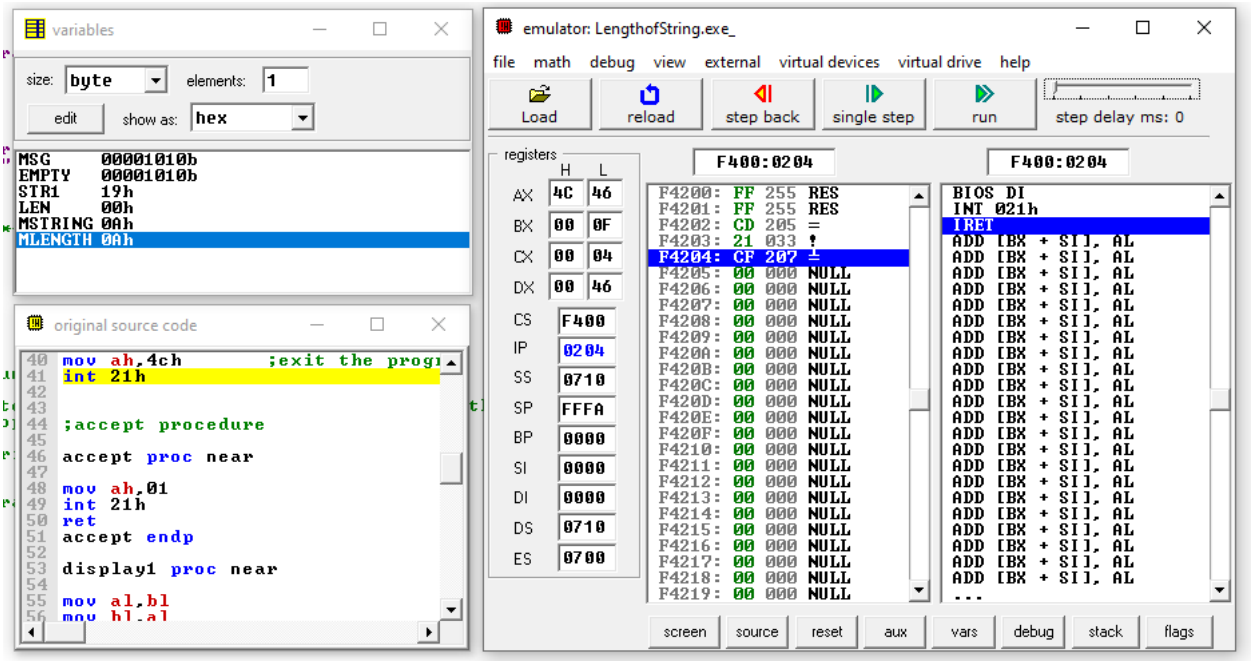
```
;macro for printing a string
print macro m
mov ah,09h
mov dx,offset m
int 21h
endm

.model small

.data

msg db 10,13, "Exiting the program $"

empty db 10,13, "   $"
str1 db 25,?,25 dup('$')

len db ?
mstring db 10,13, "Enter the string: $"
mlength db 10,13, "Length is: $"


;********** Code Segment ************

.code

start:
        mov ax,@data
        mov ds,ax

        print mstring
        call accept_string   ;function call to accept a string

        mov cl,str1+1        ;storing length in cl from first byte of the array
        mov bl,cl            ;copying in bl for displaying
        print mlength
        call display1        ;printing the length

exit:
mov ah,4ch       ;exit the program
int 21h


;accept procedure

accept proc near

mov ah,01
int 21h
ret
```

```
;accept procedure

accept proc near

mov ah,01
int 21h
ret
accept endp

display1 proc near

   mov al,bl
   mov bl,al
   and al,0f0h
   mov cl,04
   rol al,cl

   cmp al,09
   jbe number
   add al,07
number:  add al,30h
         mov dl,al
         mov ah,02
         int 21h

         mov al,bl
         and al,00fh
         cmp al,09
         jbe number2
         add al,07
number2:  add al,30h
          mov dl,al
          mov ah,02
          int 21h
ret
display1 endp


accept_string proc near

mov ah,0ah          ;accept string from user function
mov dx,offset str1  ; store the string in memory pointed by "DX"
int 21h
ret
accept_string endp

end start
end
```

Variables before Execution

output


emulator screen (80x25 chars)
Enter the string: Alokam Nikhitha
Length is: 0F
clear screen
change font
0/16


variables
size: byte
elements: 1
edit
show as: hex
MSG 00001010b
EMPTY 00001010b
STR1 19h
LEN 00h
MSTRING 0Ah
MLENGTH 0Ah
original source code
mov ah,4ch ;exit the progr
int 21h
;accept procedure
accept proc near
mov ah,01
int 21h
ret
accept endp
display1 proc near
mov al,bl
emulator: LengthofString.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
AX 4C 46
BX 00 0F
CX 00 04
DX 00 46
CS F400
IP 0204
SS 0710
SP FFFA
BP 0000
SI 0000
DI 0000
DS 0710
ES 0700
F400:0204
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
F4214: 00 000 NULL
F4215: 00 000 NULL
F4216: 00 000 NULL
F4217: 00 000 NULL
F4218: 00 000 NULL
F4219: 00 000 NULL
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
screen
source
reset
aux
vars
debug
stack
flags

Result in BX

B) COUNT OF SPACES IN STRING

CODE

```
;macro for printing a string
print macro m
mov ah,09h
mov dx,offset m
int 21h
endm

.model small


;******  Data Segment ******
.data
```

```
empty db 10,13, "   $"
str1 db 25,?,25 dup('$')
mstring db 10,13, "Enter the string: $"
mscount db 10,13, "Number of spaces: $"
mlength db 10,13, "Length is: $"
scount db ?



;********** Code Segment ************

.code

start:
mov ax,@data
mov ds,ax

        print mstring
        call accept_string
```

```
        mov si,offset str1+2  ;position si to start of
the string


            mov cl,str1+1         ;copy length in cl
        mov dh,00             ;counter to store number
of spaces
cmpagain1:  mov al,[si]      ;copy content at
memory location "si" in "al"
                cmp al,' '        ;compare "al" with space
           jne below         ;if not equal jump to label
"below"
                inc dh


below:  inc si               ;move to next character
        dec cl               ;decrement string length
counter
        jnz cmpagain1        ;if not zero check again


            mov scount,dh       ;save the count in
memory location "scount"
```

```
        mov bl,scount       ;copy count to "bl" for
printing
        print mscount
            call display1

exit:
mov ah,4ch       ;exit the program
int 21h



;accept procedure

accept proc near

mov ah,01
int 21h
ret
accept endp

display1 proc near
```

```
   mov al,bl
   mov bl,al
   and al,0f0h
   mov cl,04
   rol al,cl

   cmp al,09
   jbe number
   add al,07
number:  add al,30h
         mov dl,al
         mov ah,02
         int 21h

         mov al,bl
         and al,00fh
         cmp al,09
         jbe number2
```

```
        add al,07
number2:  add al,30h
         mov dl,al
         mov ah,02
         int 21h
ret
display1 endp



accept_string proc near


mov ah,0ah          ;accept string from user function
mov dx,offset str1  ; store the string in memory pointed by "DX"
int 21h
ret
accept_string endp
```

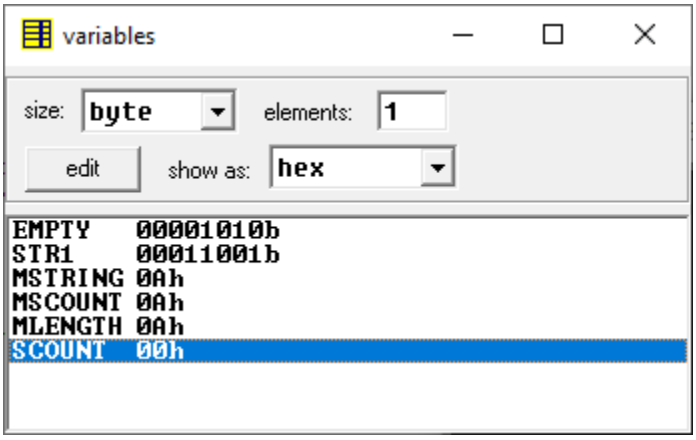
end start

end

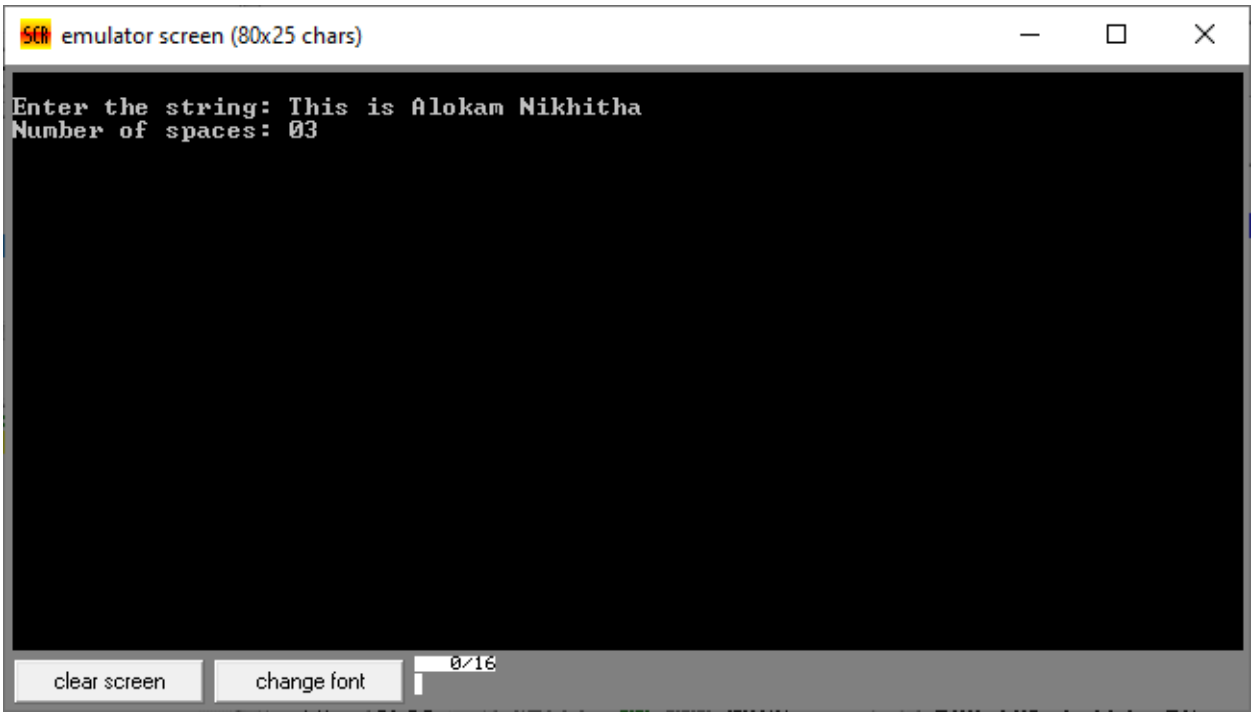
```
;macro for printing a string
print macro m
mov ah,09h
mov dx,offset m
int 21h
endm

.model small


;******  Data Segment ******
.data

empty db 10,13, "   $"
str1 db 25,?,25 dup('$')
mstring db 10,13, "Enter the string: $"
mscount db 10,13, "Number of spaces: $"
mlength db 10,13, "Length is: $"
scount db ?



;********** Code Segment ************

.code

start:
mov ax,@data
mov ds,ax

        print mstring
        call accept_string
        mov si,offset str1+2  ;position si to start of the string

        mov cl,str1+1          ;copy length in cl
        mov dh,00              ;counter to store number of spaces
cmpagain1:  mov al,[si]       ;copy content at memory location "si" in "al"
            cmp al,' '        ;compare "al" with space
            jne below         ;if not equal jump to label "below"
            inc dh

below:  inc si                ;move to next character
        dec cl                ;decrement string length counter
        jnz cmpagain1         ;if not zero check again

        mov scount,dh        ;save the count in memory location "scount"
        mov bl,scount        ;copy count to "bl" for printing
        print mscount
        call display1
```

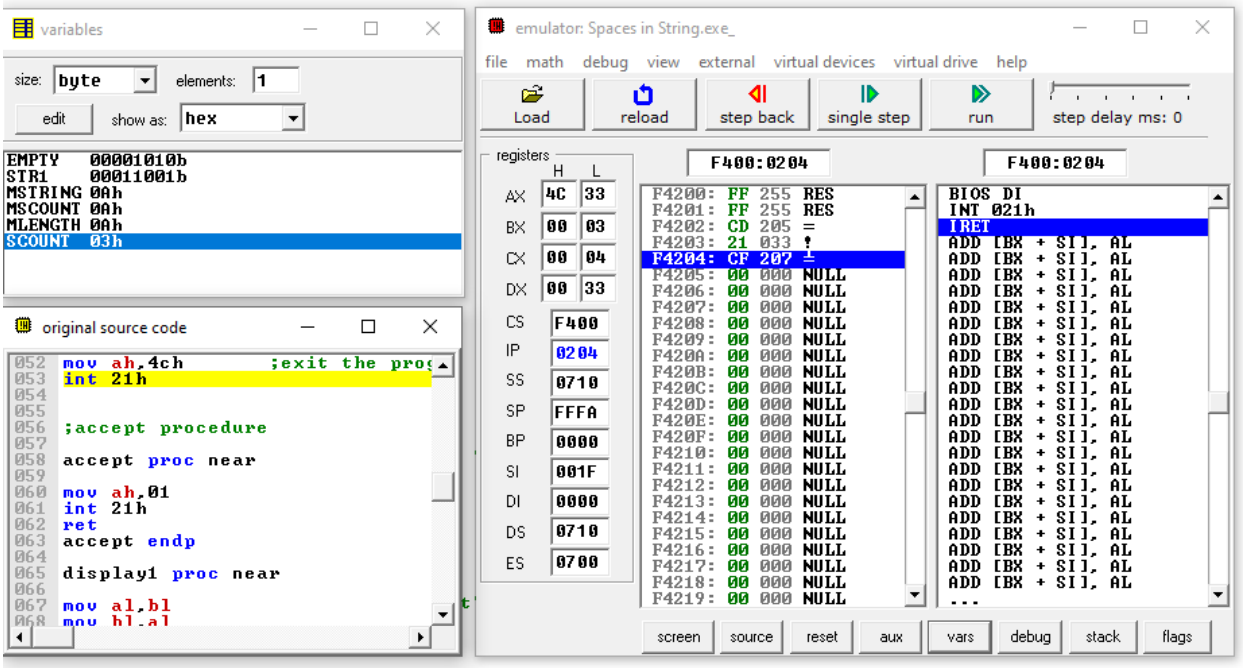
```
exit:
mov ah,4ch        ;exit the program
int 21h
;accept procedure

accept proc near

mov ah,01
int 21h
ret
accept endp

display1 proc near

   mov al,bl
   mov bl,al
   and al,0f0h
   mov cl,04
   rol al,cl

   cmp al,09
   jbe number
   add al,07
number:  add al,30h
         mov dl,al
         mov ah,02
         int 21h

         mov al,bl
         and al,00fh
         cmp al,09
         jbe number2
         add al,07
number2:  add al,30h
          mov dl,al
          mov ah,02
          int 21h
ret
display1 endp

accept_string proc near

mov ah,0ah           ;accept string from user function
mov dx,offset str1   ; store the string in memory pointed by "DX"
int 21h
ret
accept_string endp

end start
end
```

Variables before Execution

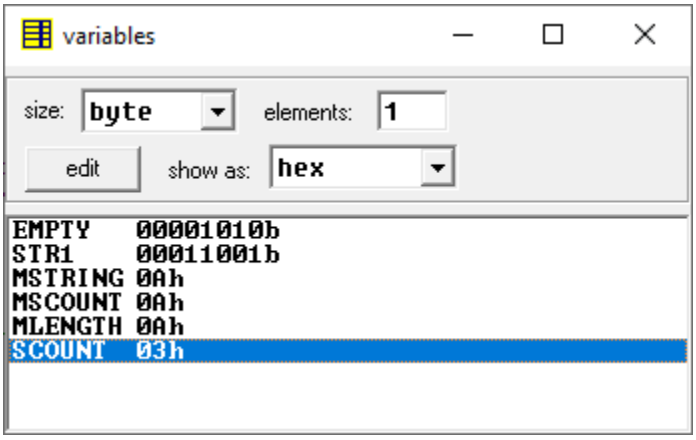


Output


emulator screen (80x25 chars)
Enter the string: This is Alokam Nikhitha
Number of spaces: 03
clear screen
change font
0/16


variables
size: byte
elements: 1
edit
show as: hex
EMPTY 00001010b
STR1 00011001b
MSTRING 0Ah
MSCOUNT 0Ah
MLENGTH 0Ah
SCOUNT 03h
original source code
mov ah,4ch ;exit the proc
int 21h
;accept procedure
accept proc near
mov ah,01
int 21h
ret
accept endp
display1 proc near
mov al,bl
emulator: Spaces in String.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
AX 4C 33
BX 00 03
CX 00 04
DX 00 33
CS F400
IP 0204
SS 0710
SP FFFA
BP 0000
SI 001F
DI 0000
DS 0710
ES 0700
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ╧
BIOS DI
INT 021h
IRET
screen
source
reset
aux
vars
debug
stack
flags

Variables after Execution



Result in Scount

C) REVERSE STRING AND PRINT IT

```
;macro for printing a string
print macro m
mov ah,09h
mov dx,offset m
int 21h
endm

.model small
```

```
;******  Data Segment ******
.data

empty db 10,13, "   $"
str1 db 25,?,25 dup('$')
str2 db 25,?,25 dup('$')
mstring db 10,13, "Enter the string: $"
mreverse db 10,13, "Reversed string: $"

;********** Code Segment ************

.code

start:
mov ax,@data
mov ds,ax

        print mstring
        call accept_string
```

```
        mov si,offset str1      ;point si to start of
string1
        mov di,offset str2      ;point di to start of
string2




        mov al,[si]         ;copy first two locations of
string1 to string2
        mov [di],al         ;since these contain the size
and length of the string
        inc si              ;which are same in reverse
string also
        inc di



        mov al,[si]
        mov [di],al
        inc si
```

```
        inc di


        mov cl,str1+1 ;       copy length in cl
        mov ch,00
        add si,cx             ;add length of string1 to si
to move it to last location
        dec si                ;si at last location of string1


move_more: mov al,[si]       ;copying character
one by one from string1 pointed by si
          mov [di],al       ; to string2 pointed by "di" in
reverse order as si moves
          dec si            ; from last character to first
character
          inc di
          dec cl
          jnz move_more



             print mreverse
```

```
        print str2+2     ;  printing the reversed string
              print empty

exit:
mov ah,4ch       ;exit the program
int 21h



;accept procedure

accept proc near

mov ah,01
int 21h
ret
accept endp

display1 proc near
```

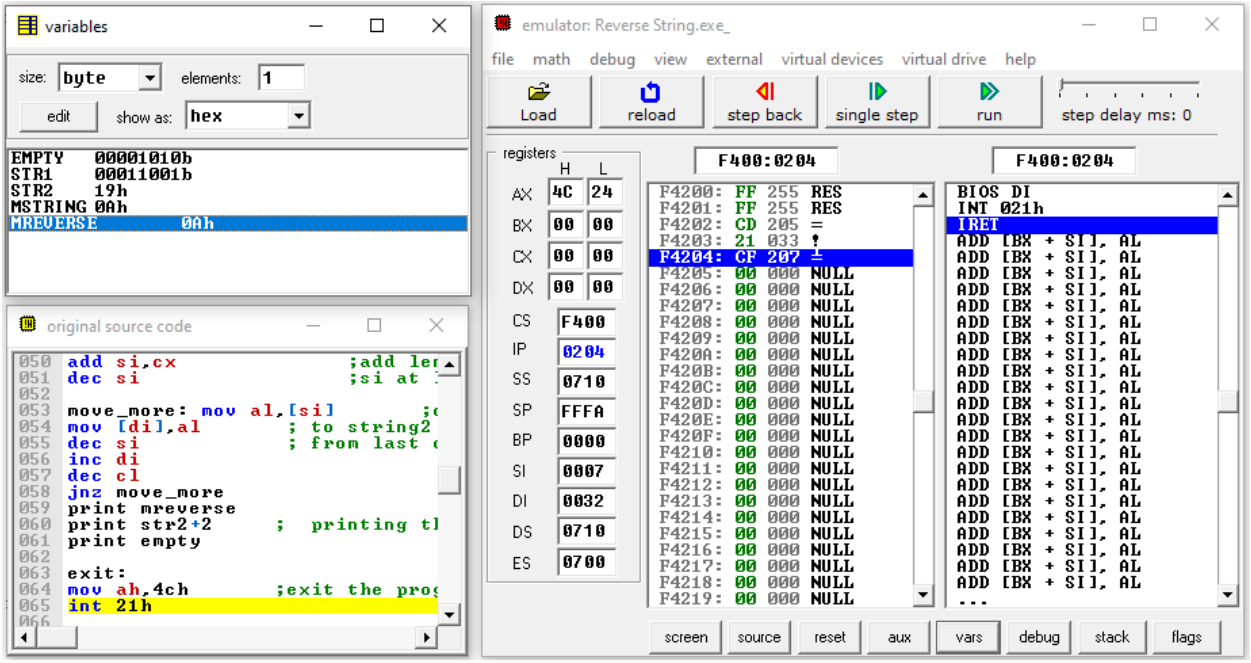
```
   mov al,bl
   mov bl,al
   and al,0f0h
   mov cl,04
   rol al,cl

   cmp al,09
   jbe number
   add al,07
number:  add al,30h
         mov dl,al
         mov ah,02
         int 21h

         mov al,bl
         and al,00fh
         cmp al,09
         jbe number2
         add al,07
```

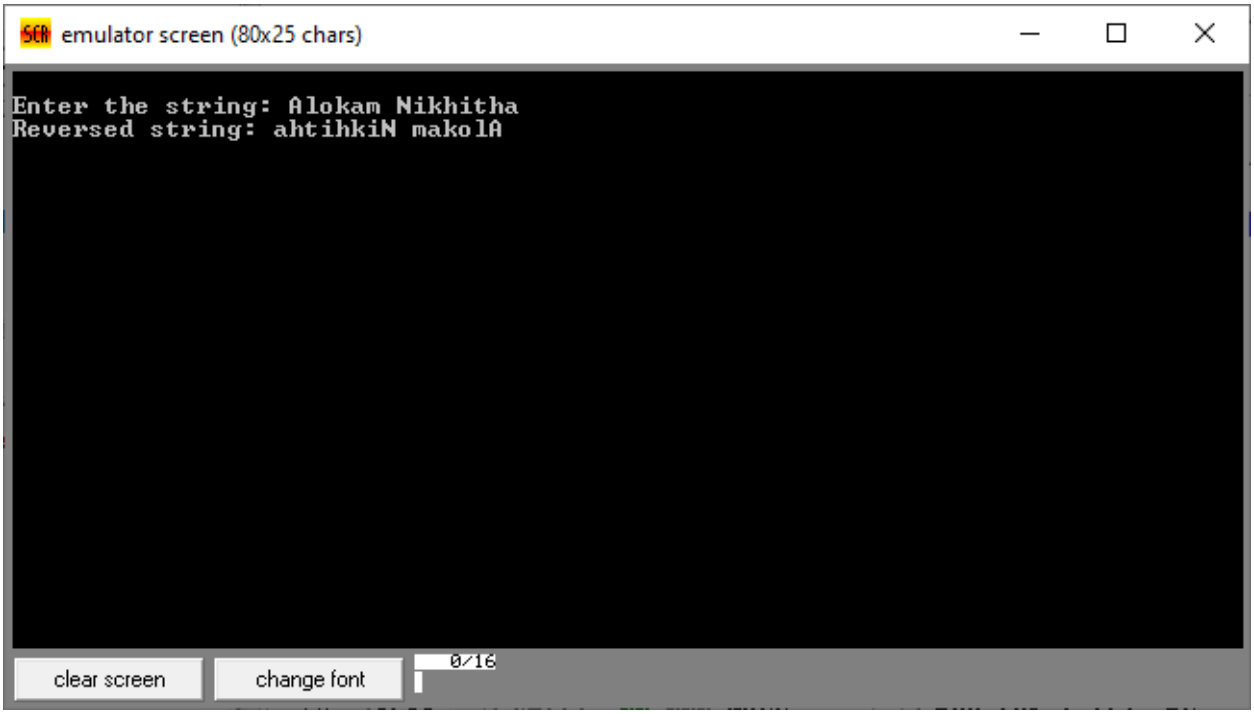
```
number2:  add al,30h
          mov dl,al
          mov ah,02
          int 21h
ret
display1 endp



accept_string proc near


mov ah,0ah          ;accept string from user function
mov dx,offset str1  ; store the string in memory pointed by "DX"
int 21h
ret
accept_string endp


end start
```

end

```
;macro for printing a string
print macro m
mov ah,09h
mov dx,offset m
int 21h
endm

.model small


;******  Data Segment ******
.data

empty db 10,13, "   $"
str1 db 25,?,25 dup('$')
str2 db 25,?,25 dup('$')
mstring db 10,13, "Enter the string: $"
mreverse db 10,13, "Reversed string: $"

;********** Code Segment ************

.code

start:
mov ax,@data
mov ds,ax

        print mstring
        call accept_string


        mov si,offset str1      ;point si to start of string1
        mov di,offset str2      ;point di to start of string2



        mov al,[si]         ;copy first two locations of string1 to string2
        mov [di],al         ;since these contain the size and length of the string
        inc si              ;which are same in reverse string also
        inc di


        mov al,[si]
        mov [di],al
        inc si
        inc di

        mov cl,str1+1 ;         copy length in cl
        mov ch,00
        add si,cx               ;add length of string1 to si to move it to last location
        dec si                  ; from last character to first character
        inc di
        dec cl
        jnz move_more
        print mreverse
        print str2+2      ;  printing the reversed string
        print empty

exit:
mov ah,4ch       ;exit the program
int 21h

;accept procedure
accept proc near
mov ah,01
int 21h
ret
accept endp
display1 proc near

   mov al,bl
   mov bl,al
   and al,0f0h
   mov cl,04
   rol al,cl

   cmp al,09
   jbe number
   add al,07
number:  add al,30h
         mov dl,al
         mov ah,02
         int 21h
         mov al,bl
         and al,00fh
         cmp al,09
         jbe number2
         add al,07
number2:  add al,30h
          mov dl,al
          mov ah,02
          int 21h
ret
display1 endp
accept_string proc near
mov ah,0ah          ;accept string from user function
mov dx,offset str1  ; store the string in memory pointed by "DX"
int 21h
ret
accept_string endp
```

Output


variables
size: byte
elements: 1
edit
show as: hex
EMPTY 00001010b
STR1 00011001b
STR2 19h
MSTRING 0Ah
MREVERSE 0Ah
original source code
050 add si,cx ;add ler
051 dec si ;si at
052
053 move_more: mov al,[si] ;
054 mov [di],al ; to string2
055 dec si ; from last
056 inc di
057 dec cl
058 jnz move_more
059 print mreverse
060 print str2+2 ; printing tl
061 print empty
062
063 exit:
064 mov ah,4ch ;exit the prog
065 int 21h
066
emulator: Reverse String.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 4C 24
BX 00 00
CX 00 00
DX 00 00
CS F400
IP 0204
SS 0710
SP FFFA
BP 0000
SI 0007
DI 0032
DS 0710
ES 0700
F400:0204
F400:0204
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
F4214: 00 000 NULL
F4215: 00 000 NULL
F4216: 00 000 NULL
F4217: 00 000 NULL
F4218: 00 000 NULL
F4219: 00 000 NULL
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
...
screen
source
reset
aux
vars
debug
stack
flags


emulator screen (80x25 chars)
Enter the string: Alokam Nikhitha
Reversed string: ahtihkiN mako1A
clear screen
change font
0/16