VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Fall Semester 2021-22

Microprocessors and Interfacing LAB

CSE2006

Slot – L43+L44

Digital Assignment 6

Name: Ishan Sagar Jogalekar

Reg. No: 19BCE2250

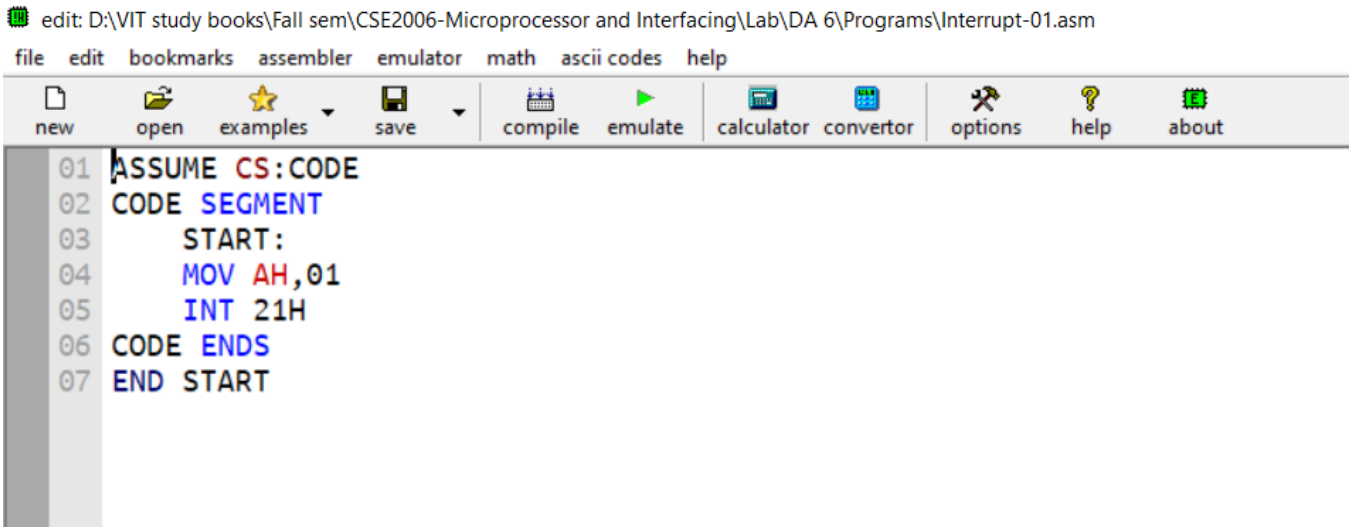
DOS BIOS Interrupts –

1. INT21H / AH=01h –

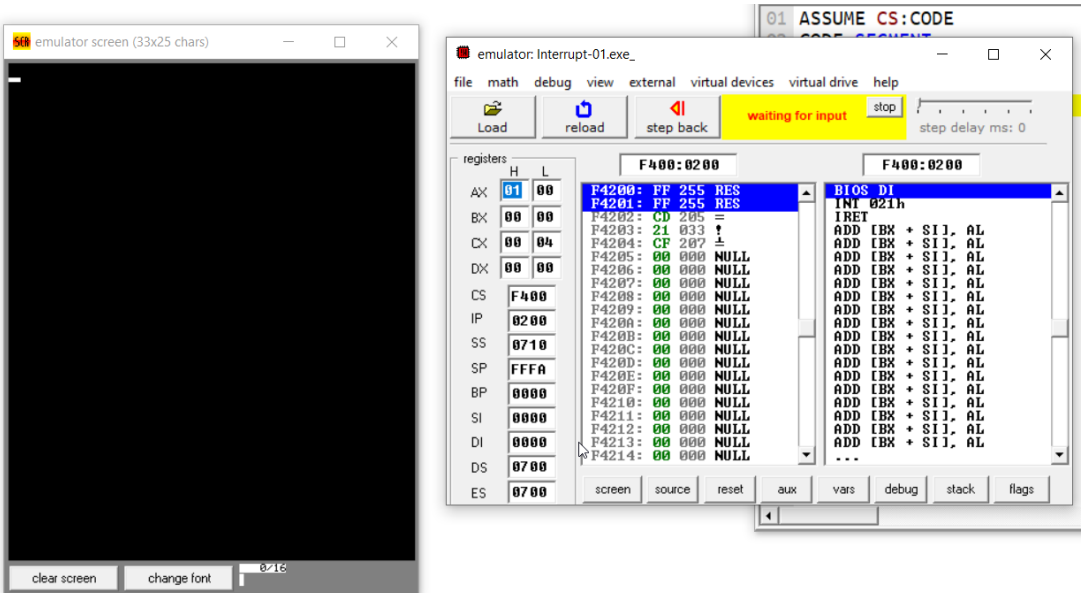
Read character from standard input (from user) and stored it in AL with echo. Function will wait until key pressed when there is no character in keyboard buffer.

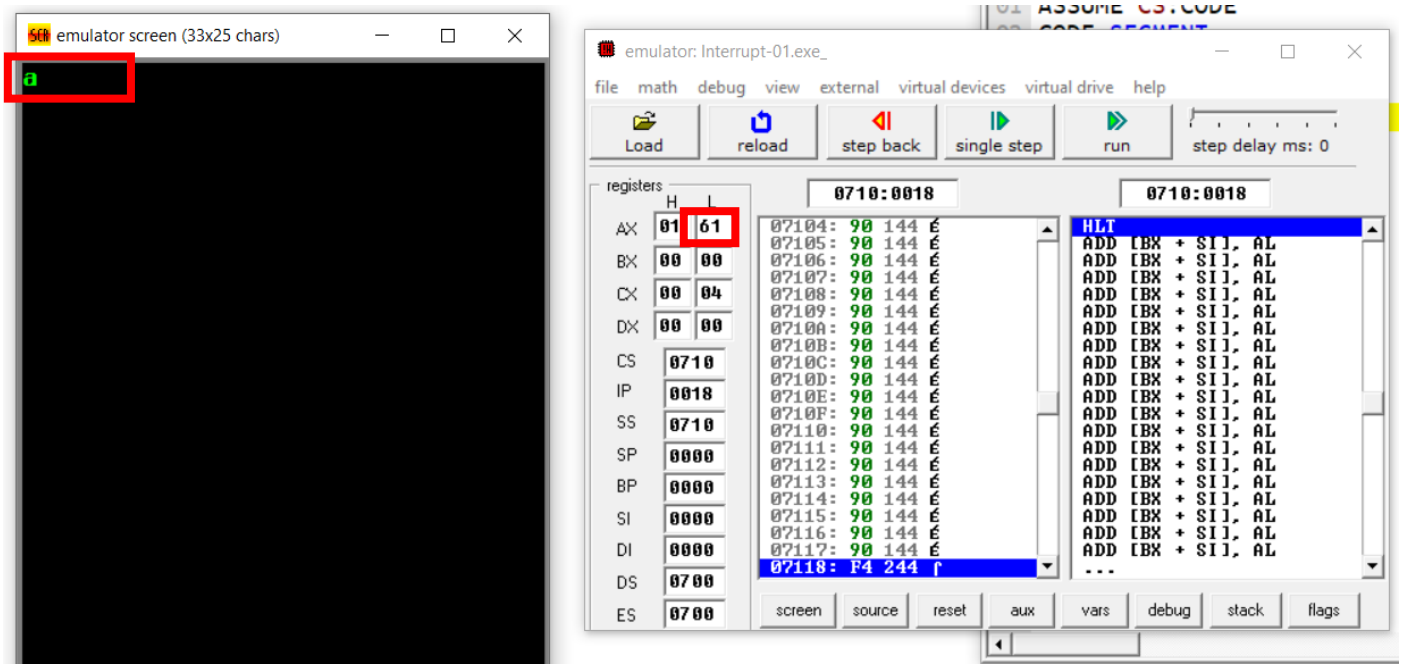
Program:

```
ASSUME CS:CODE
CODE SEGMENT
    START:
    MOV AH,01
    INT 21H
CODE ENDS
END START
```



Output:





2. **INT21H / AH=09h –**
   Display of string, output of string at DS:DX but string must be terminated by $ symbol.

Program:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    STR1 DB 0DH,0AH,"ISHAN$"
    STR2 DB 0DH,0AH,"JOGALEKAR$"
DATA ENDS
CODE SEGMENT
    START:
    MOV AX,DATA
    MOV DS,AX
    MOV DX, OFFSET STR1
    MOV AH,09H
    INT 21H
    MOV DX, OFFSET STR2
    MOV AH,09H
    INT 21H
CODE ENDS
```

END START

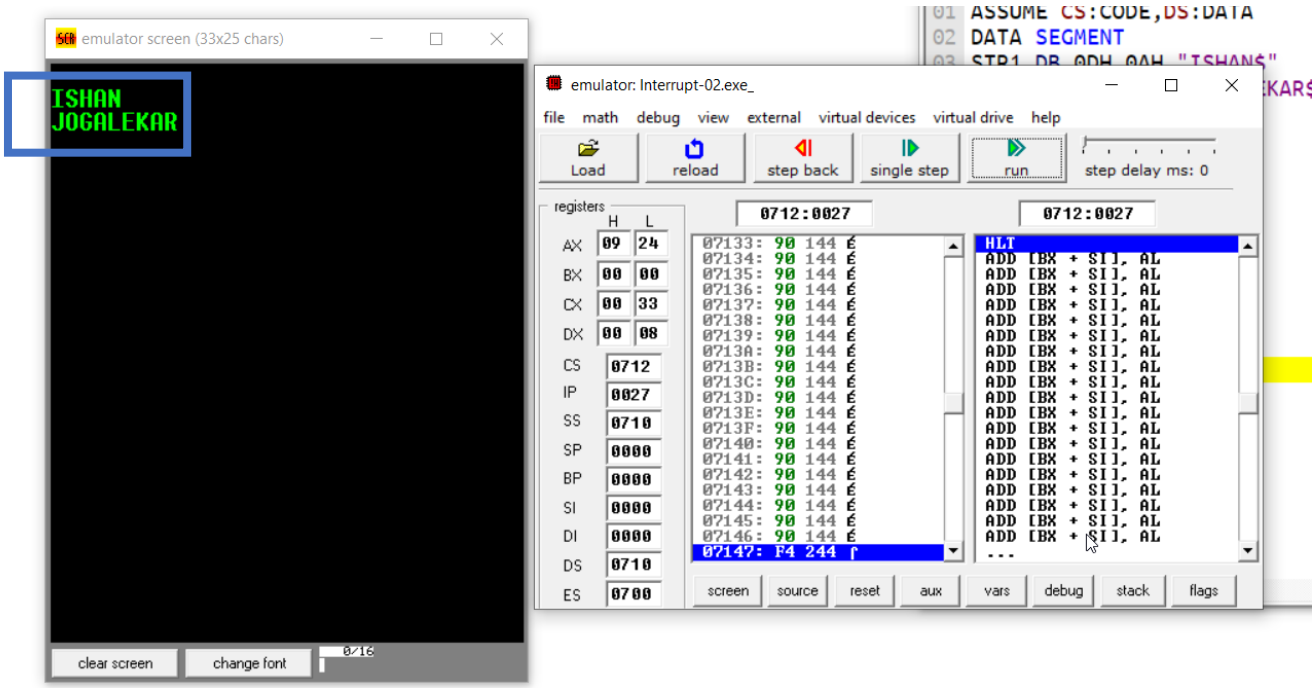
edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 6\Programs\Interrupt-02.asm

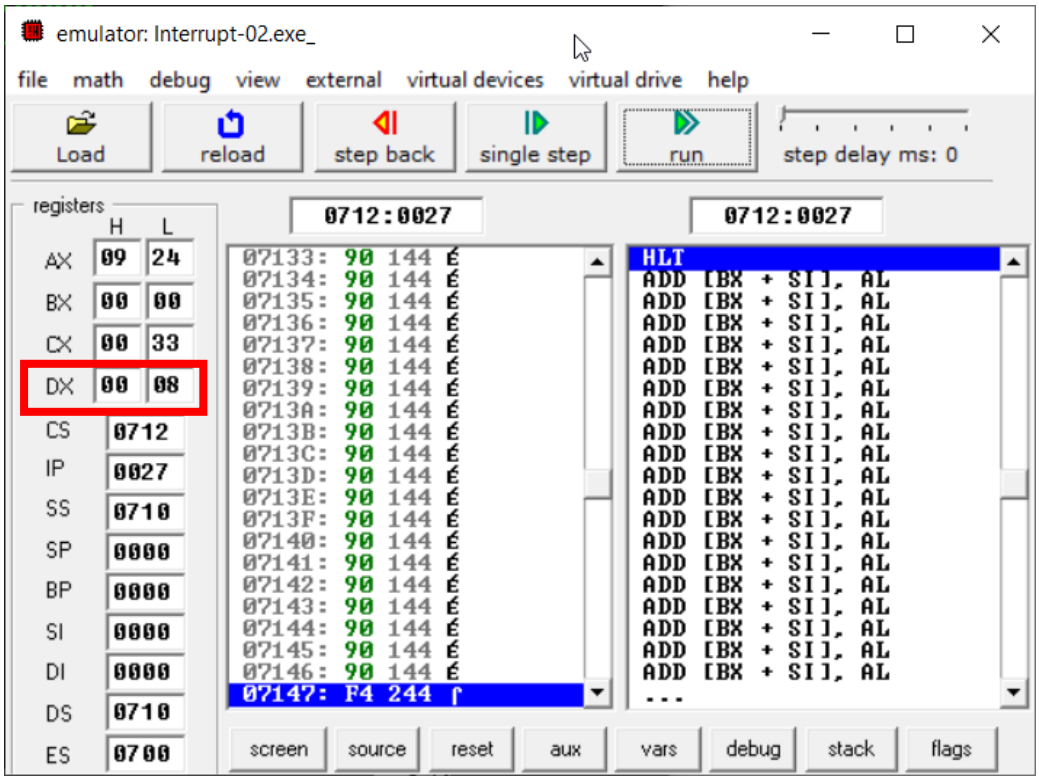
file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    STR1 DB 0DH,0AH,"ISHAN$"
    STR2 DB 0DH,0AH,"JOGALEKAR$"
DATA ENDS
CODE SEGMENT
    START:
    MOV AX,DATA
    MOV DS,AX
    MOV DX, OFFSET STR1
    MOV AH,09H
    INT 21H
    MOV DX, OFFSET STR2
    MOV AH,09H
    INT 21H
CODE ENDS
END START
```

Output:

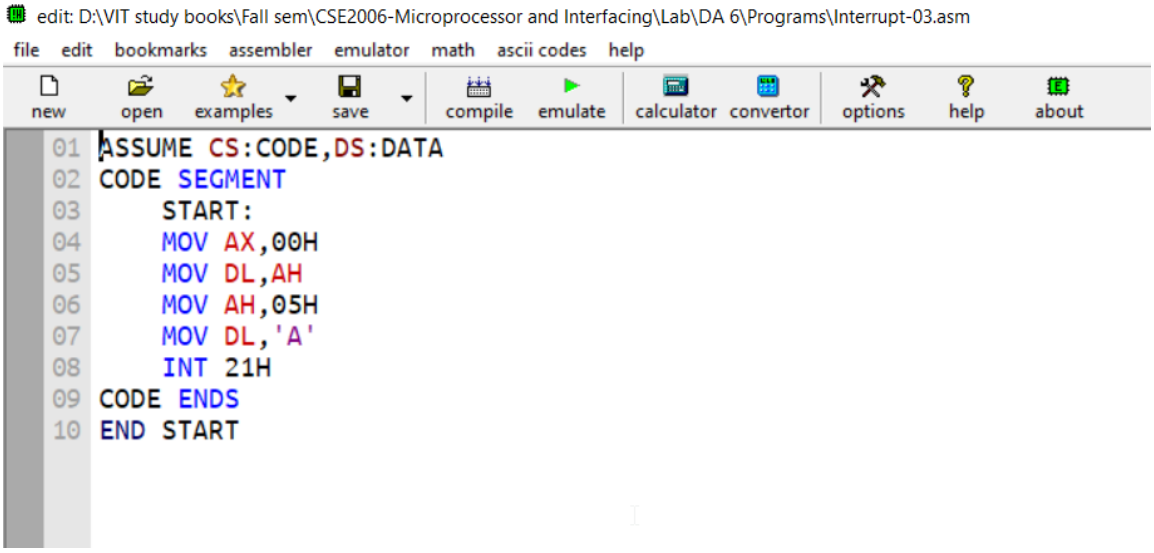




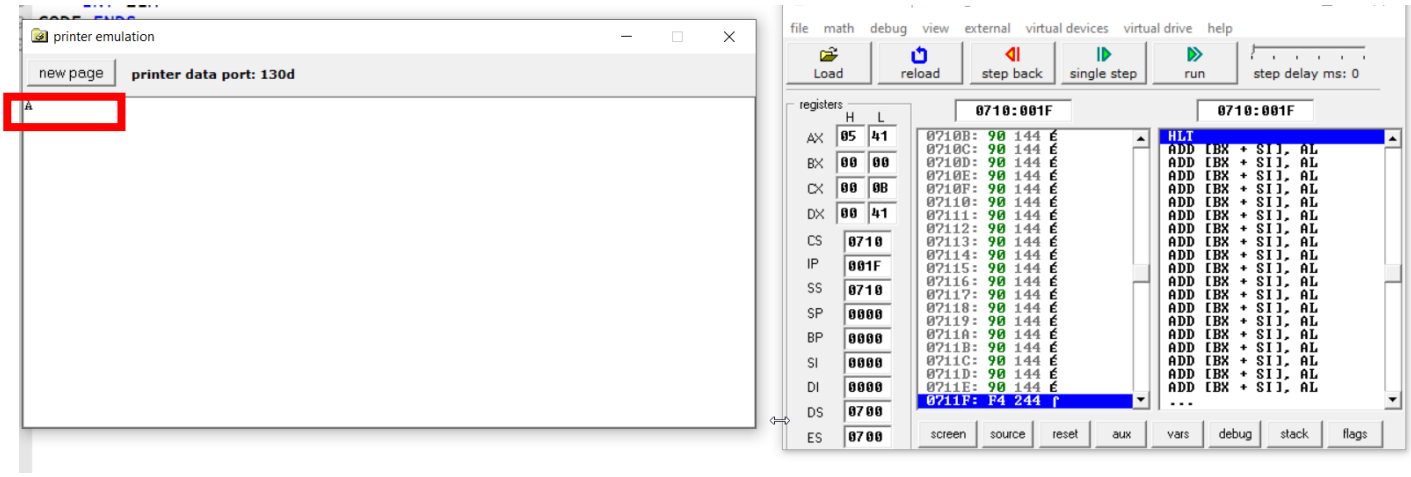
3. **INT21H / AH=05h –**
   Output character to printer, DL is input character after execution DL=AL.

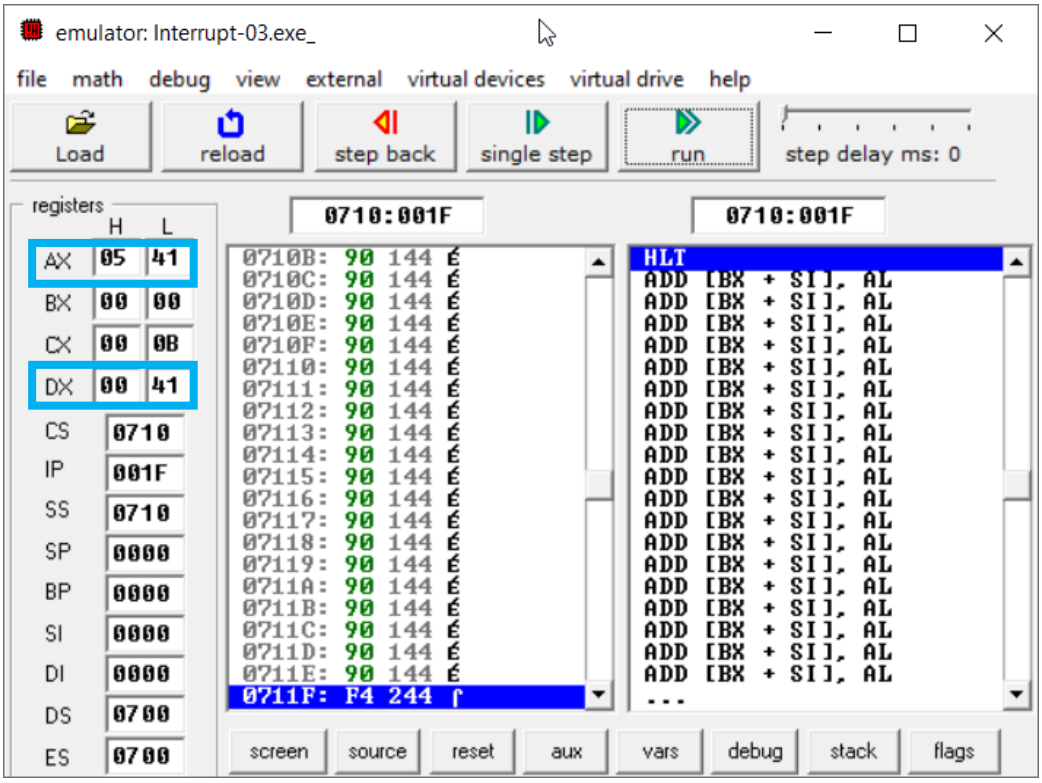
Program:

```
ASSUME CS:CODE,DS:DATA
CODE SEGMENT
    START:
    MOV AX,00H
    MOV DL,AH
    MOV AH,05H
    MOV DL,'A'
    INT 21H
CODE ENDS
END START
```



Output:

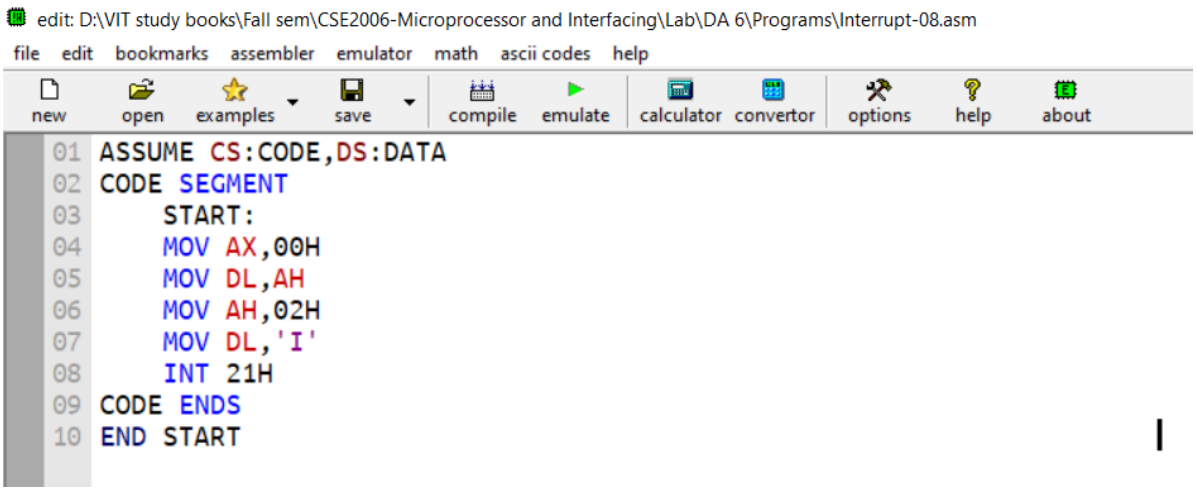

printer emulation
new page
printer data port: 130d
A
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 05 41
BX 00 00
CX 00 0B
DX 00 41
CS 0710
IP 001F
SS 0710
SP 0000
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0710:001F
0710:001F
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
0711B: 90 144 É
0711C: 90 144 É
0711D: 90 144 É
0711E: 90 144 É
0711F: F4 244 ⌠
HLT
ADD [BX + SI], AL
...
screen source reset aux vars debug stack flags


emulator: Interrupt-03.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 05 41
BX 00 00
CX 00 0B
DX 00 41
CS 0710
IP 001F
SS 0710
SP 0000
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0710:001F
0710:001F
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
0711B: 90 144 É
0711C: 90 144 É
0711D: 90 144 É
0711E: 90 144 É
0711F: F4 244 ⌠
HLT
ADD [BX + SI], AL
...
screen source reset aux vars debug stack flags

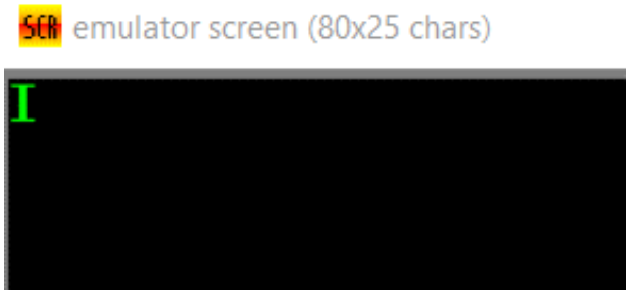
4. **INT21H / AH=02h –**
   Write character to standard output. DL is standard input character and after execution DL = AL.

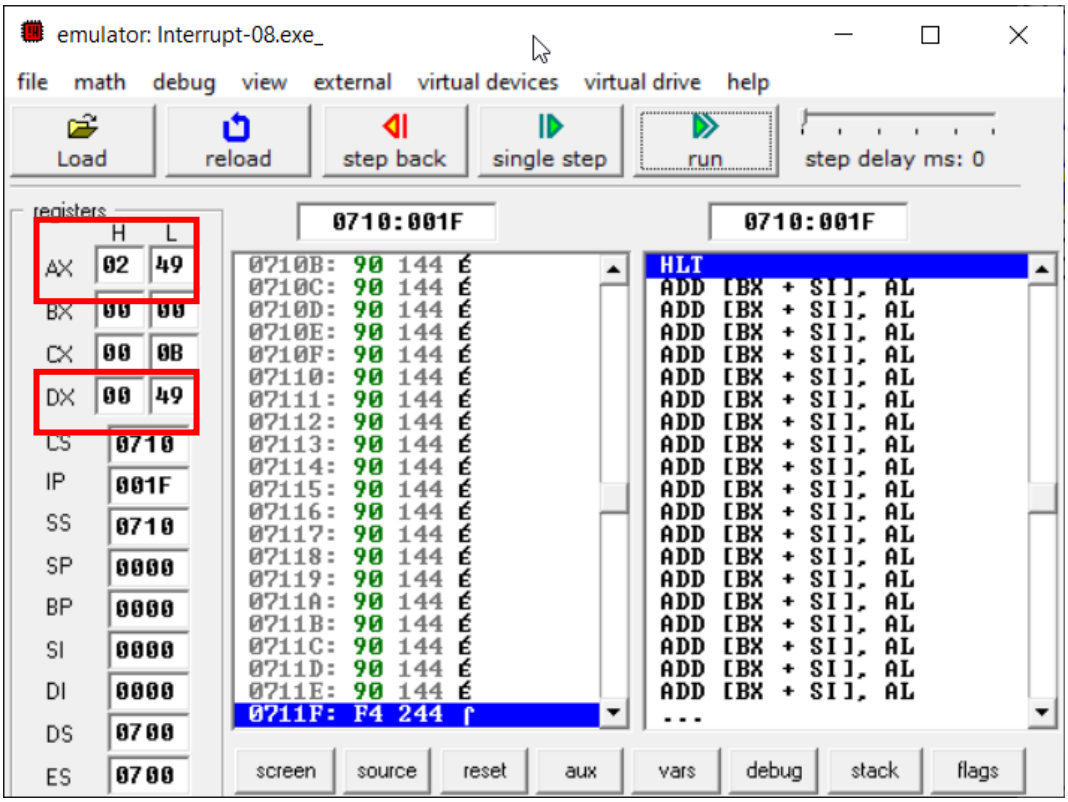
Program:

```
ASSUME CS:CODE,DS:DATA
CODE SEGMENT
    START:
    MOV AX,00H
    MOV DL,AH
    MOV AH,02H
    MOV DL,'I'
    INT 21H
CODE ENDS
END START
```



Output:





5. **INT21H / AH=0Ah –**
   input of a string to DS:DX, fist byte is buffer size, second byte is number of chars actually read. this function does not add '$' in the end of string.

Program:

```
org 100h
   MOV DX, OFFSET buffer
   MOV AH, 0AH
  INT 21H
  JMP print
  buffer DB 10,?, 10 dup(' ')
  print:
          XOR BX, BX
          MOV BL, buffer[1]
          MOV buffer[bx+2], '$'
          MOV DX, OFFSET buffer + 2
          MOV AH, 09H
  INT 21H
RET
```

edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 6\Programs\Interrupt-04.asm

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
01 org 100h
02     MOV DX, OFFSET buffer
03     MOV AH, 0AH
04     INT 21H
05     JMP print
06     buffer DB 10,?, 10 dup(' ')
07     print:
08         XOR BX, BX
09         MOV BL, buffer[1]
10         MOV buffer[bx+2], '$'
11         MOV DX, OFFSET buffer + 2
12         MOV AH, 09H
13     INT 21H
14 RET
```

Output:

emulator screen (80x25 chars)

```
19BCE2250_
```

emulator screen (80x25 chars)

```
19BCE2250 19BCE2250
```

emulator: Interrupt-04.com_

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	09	24
BX	00	09
CX	00	28
DX	01	0B
CS	F400	
IP	0154	
SS	0700	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

F400:0154

```
F4150: FF 255 RES
F4151: FF 255 RES
F4152: CD 205 =
F4153: 20 032 SPA
F4154: CF 207 ±
F4155: 00 000 NULL
F4156: 00 000 NULL
F4157: 00 000 NULL
F4158: 00 000 NULL
F4159: 00 000 NULL
F415A: 00 000 NULL
F415B: 00 000 NULL
F415C: 00 000 NULL
F415D: 00 000 NULL
F415E: 00 000 NULL
F415F: 00 000 NULL
F4160: FF 255 RES
F4161: FF 255 RES
F4162: CD 205 =
F4163: 1A 026 →
F4164: CF 207 ±
```

F400:0154

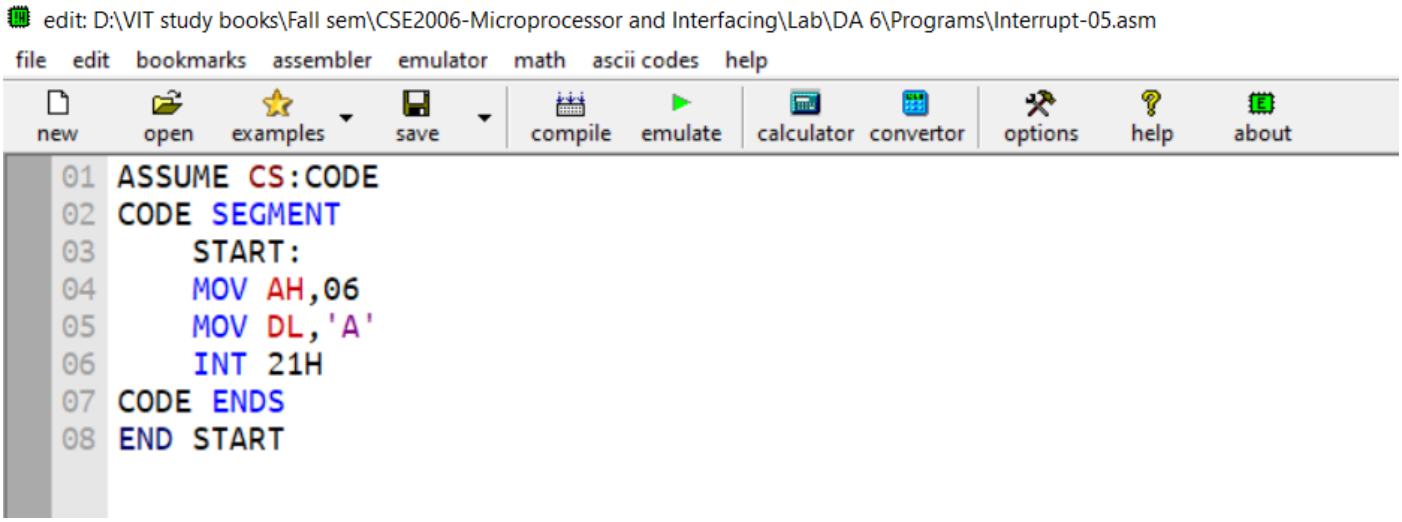
```
BIOS DI
INT 020h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
SBB CL, BH
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
ADD BH, CL
ADD [BX + SI], AL
...
```

screen source reset aux vars debug stack flags

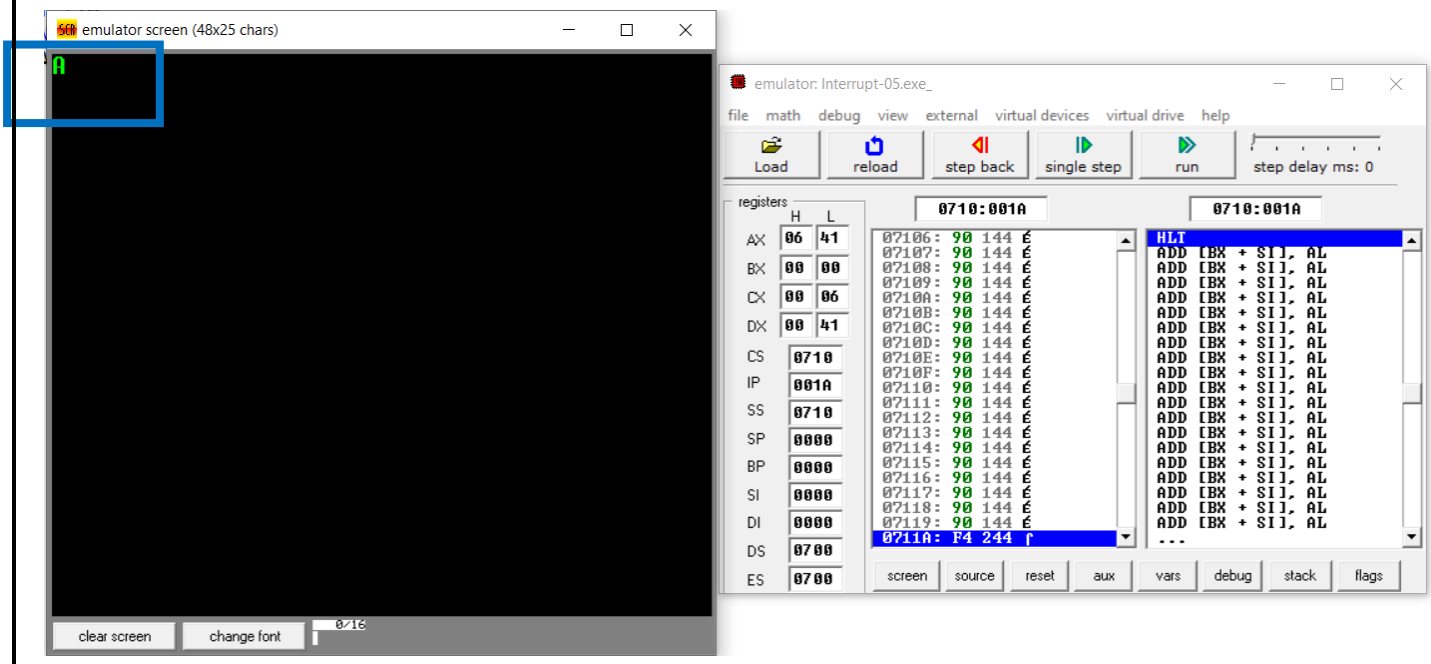
6. INT21H / AH=06h –
Direct console input or output.

Program:

```
ASSUME CS:CODE
CODE SEGMENT
    START:
    MOV AH,06
    MOV DL,'A'
    INT 21H
CODE ENDS
END START
```



Output:

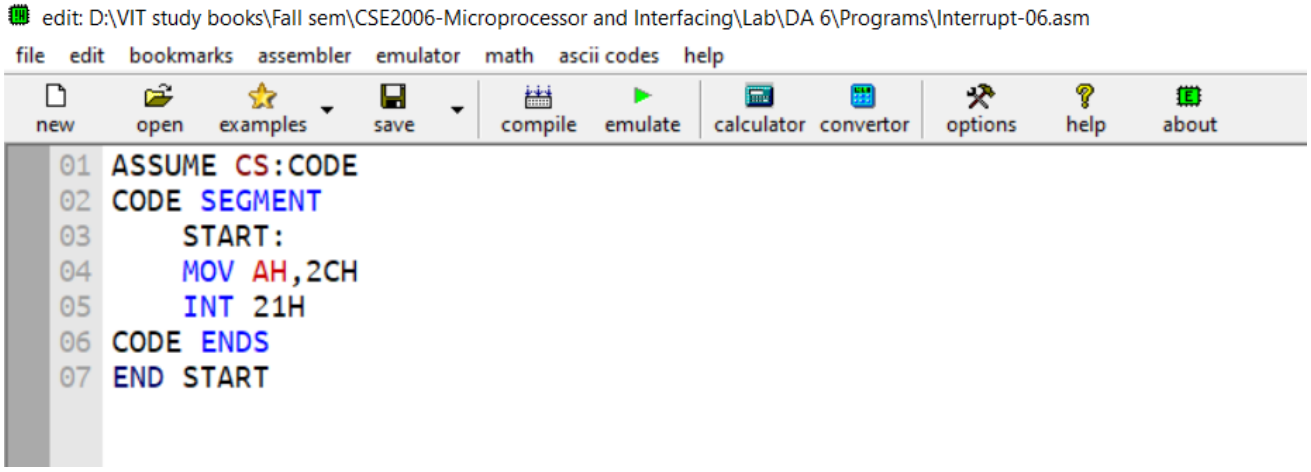


7. INT21H / AH=2Ch –

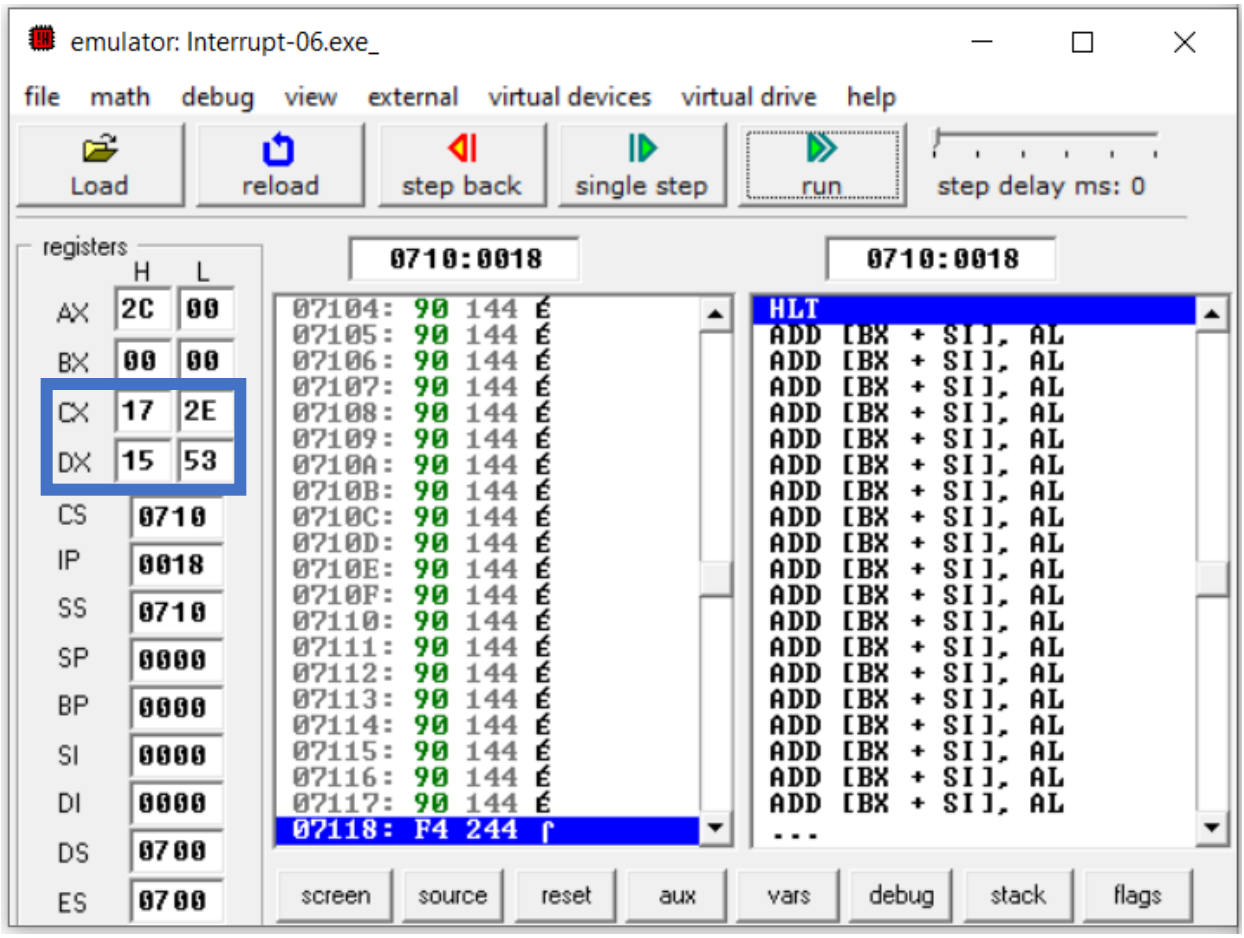
Get system time. CH = hour. CL = minute. DH = second. DL = 1/100 seconds.

Program:

```
ASSUME CS:CODE
CODE SEGMENT
   START:
   MOV AH,2CH
   INT 21H
CODE ENDS
END START
```



Output:

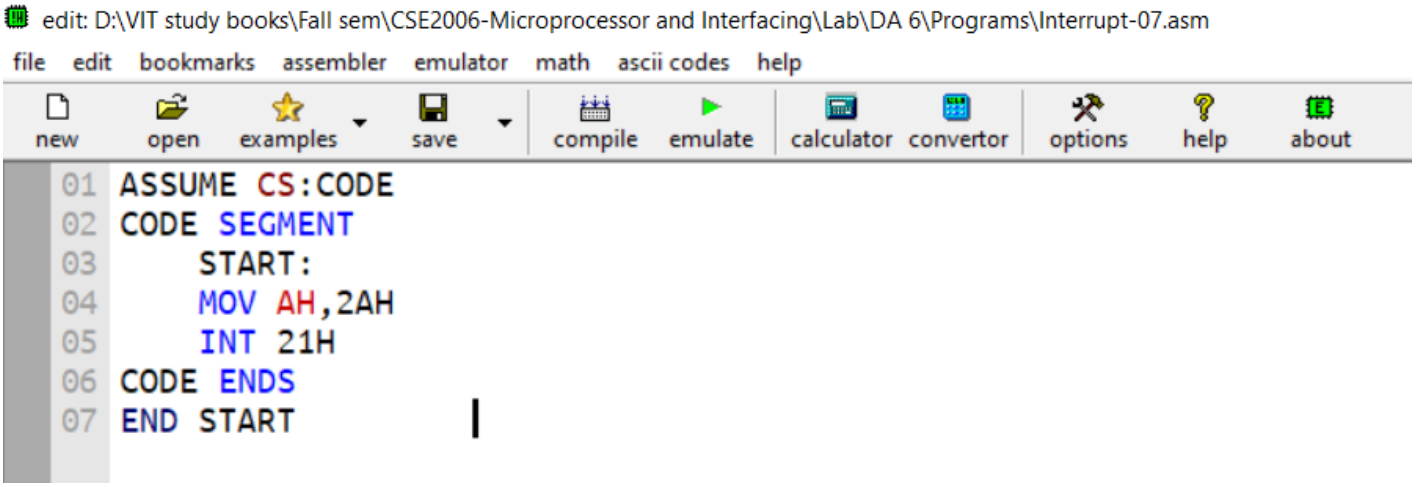


8. INT21H / AH=2Ah –

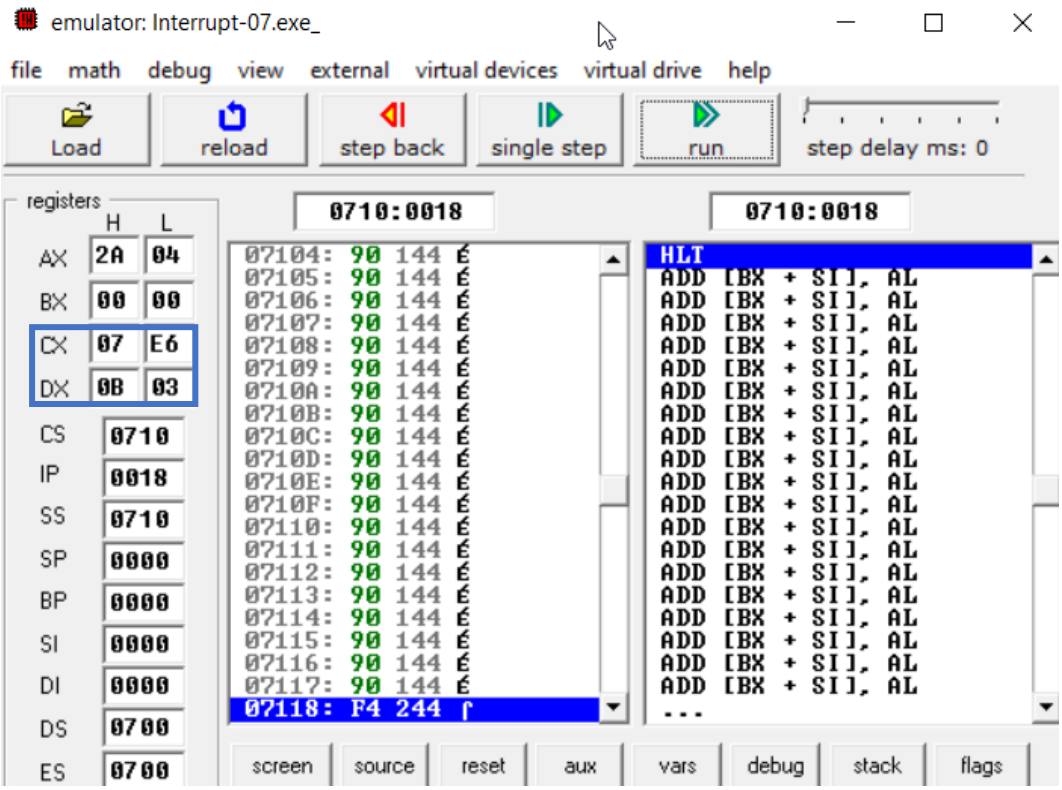
Get system Date, CX = year (1980-2099). DH = month. DL = day. AL = day of week

Program:

```
ASSUME CS:CODE
CODE SEGMENT
   START:
   MOV AH,2AH
   INT 21H
CODE ENDS
END START
```



Output:



String Instructions –

1. String Reverse –

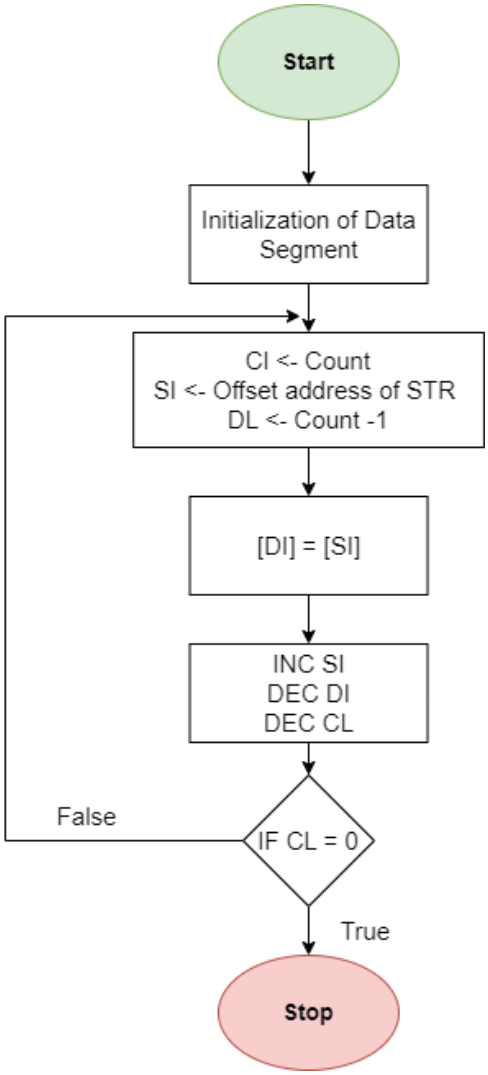
Input:

Input string as in Data stack. At the end add $ sign to indicate as end of string.

Algorithm:

1. Create a string
2. Traverse through the string
3. Push the characters in the stack
4. Count the number of characters
5. Load the starting address of the string
6. POP the top character of the stack until count is not equal to zero
7. Put the character and reduce the count and increase the address
8. Continue until the count is greater than zero
9. Load the effective address of the string in dx using LEA command
10. Print the string by calling the interrupt with 9H in AH
11. The string must be terminated by '$' sign

Flowchart:

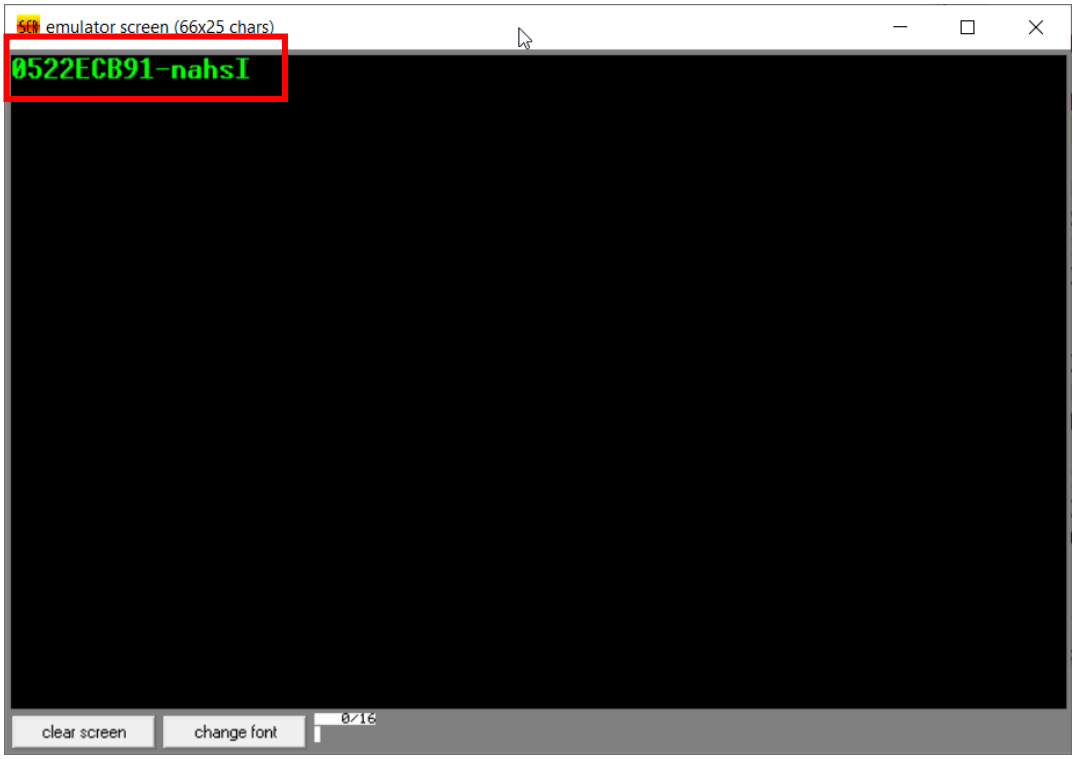


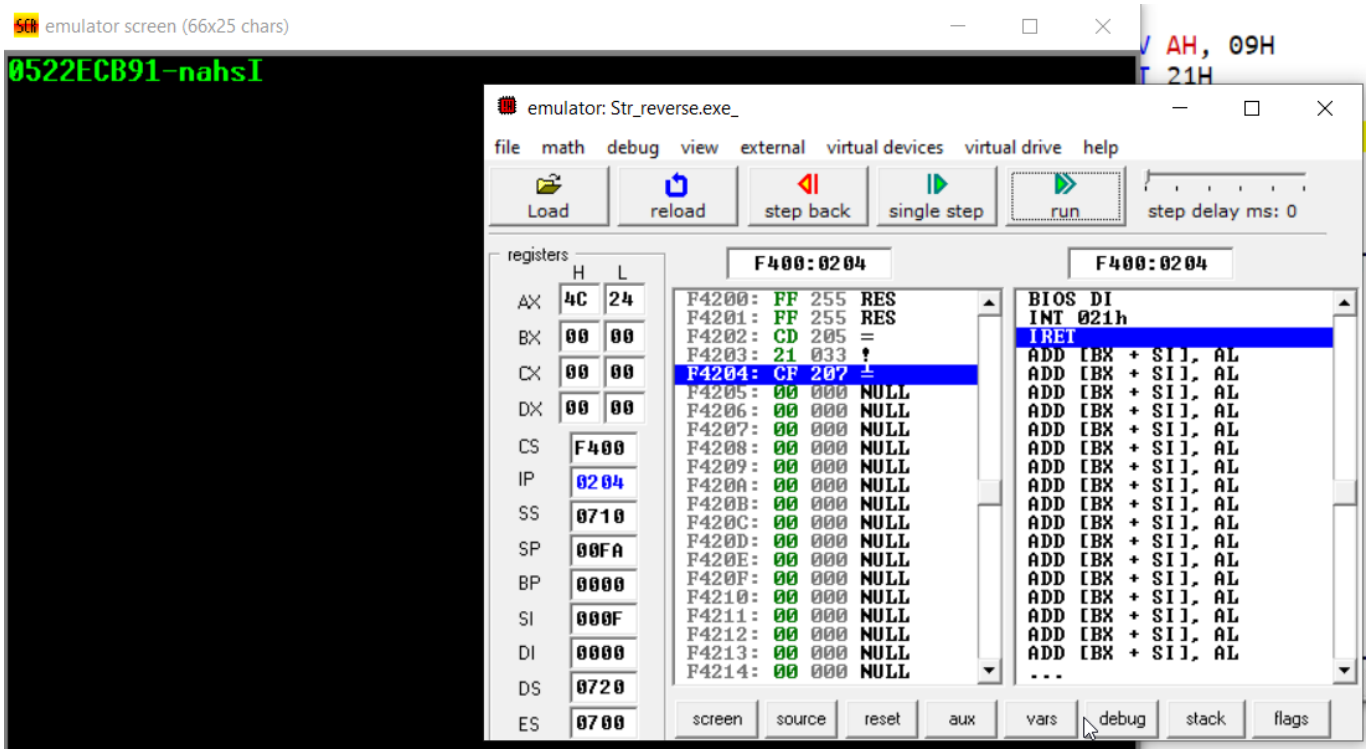
Program:

```
.MODEL SMALL
.STACK 100H
.DATA
STR1 DB 'Ishan-19BCE2250$'

.CODE
MAIN PROC FAR
MOV AX,@DATA
MOV DS,AX
CALL REVERSE
LEA DX,STR1

MOV AH, 09H
INT 21H
MOV AH, 4CH
INT 21H

MAIN ENDP
REVERSE PROC
   MOV SI, OFFSET STR1
   MOV CX, 0H

   LOOP1:
   MOV AX, [SI]
   CMP AL, '$'
   JE LABEL1
   PUSH [SI]
   INC SI
   INC CX
   JMP LOOP1

   LABEL1:
   MOV SI, OFFSET STR1
         LOOP2:
         CMP CX,0
         JE EXIT
         POP DX
         XOR DH, DH
         MOV [SI], DX
         INC SI
         DEC CX
         JMP LOOP2
   EXIT:
   MOV [SI],'$ '
   RET
REVERSE ENDP
```

END MAIN

Output:





2. String Comparison –

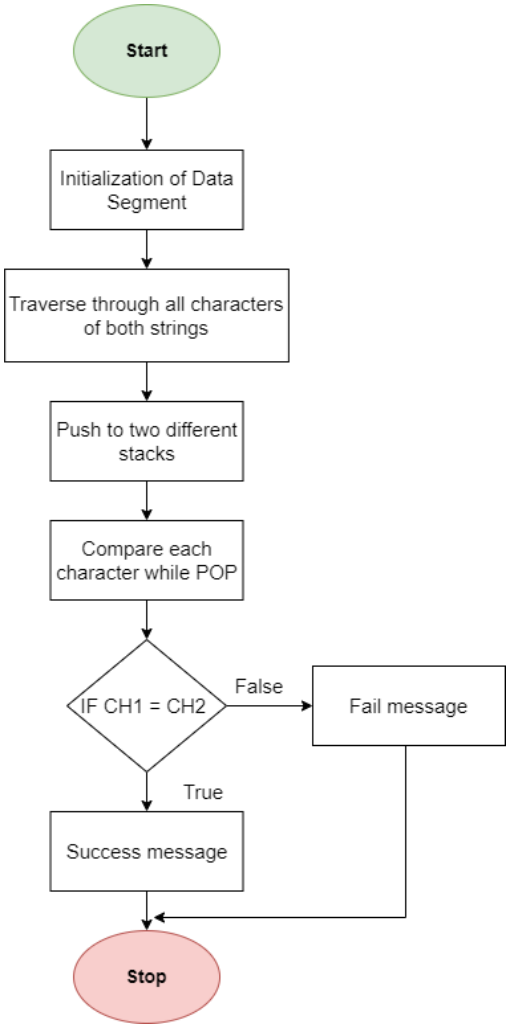
Input:

User input of two strings for compare both, if same result will be Strings are same. Use of interrupt AH = 09h with OFFSET.

Algorithm:

1. Create a string
2. Traverse through the strings
3. Push the characters in the different stacks.
4. Compare each character.
5. Load the starting address of the string
6. POP the top character of the stack until count is not equal to zero and compare the characters.
7. JMP to success message if every character matches otherwise fail message will be printed. Continue until the count is greater than zero.
8. Print the string by calling the interrupt with 9H in AH. The string must be terminated by '$' sign.

Flowchart:



Program:

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
CR EQU 13
LF EQU 10

M1 DB CR,LF,"Enter First String: $"
M2 DB CR,LF,LF,"Enter Second String: $"

STR1 DB 0BH,12 DUP(?)
STR2 DB 0BH,12 DUP(?)

SM DB CR,LF,LF,"Same Strings $"
FM DB CR,LF,LF,"Different Strings $"
DATA ENDS

CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV ES,AX
   LEA DX,M1
   MOV AH,09
   INT 21H

   MOV DX,OFFSET STR1
   MOV AH,0AH
   INT 21H

   LEA DX,M2
   MOV AH,09
   INT 21H

   MOV DX,OFFSET STR2
   MOV AH,0AH
   INT 21H

   MOV SI, OFFSET STR1
   MOV DI, OFFSET STR2
   CLD
   MOV CX,6H
   REPE CMPSB
   JZ SUCCESS
   LEA DX,FM
   JMP DISPLAY
   SUCCESS:
      LEA DX,SM
```

```
    DISPLAY:
        MOV AH,09
        INT 21H
    MOV AH,4CH
    INT 21H

CODE ENDS
END START
```

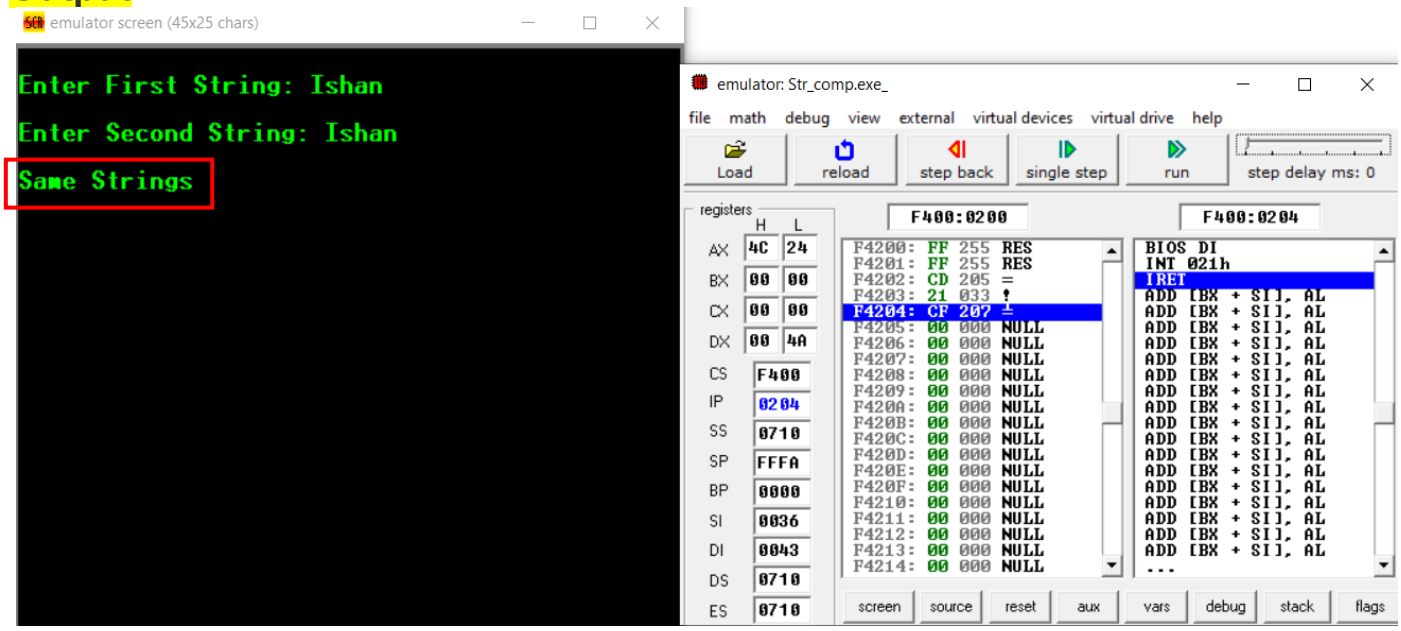
edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 6\Programs\Str_comp.asm

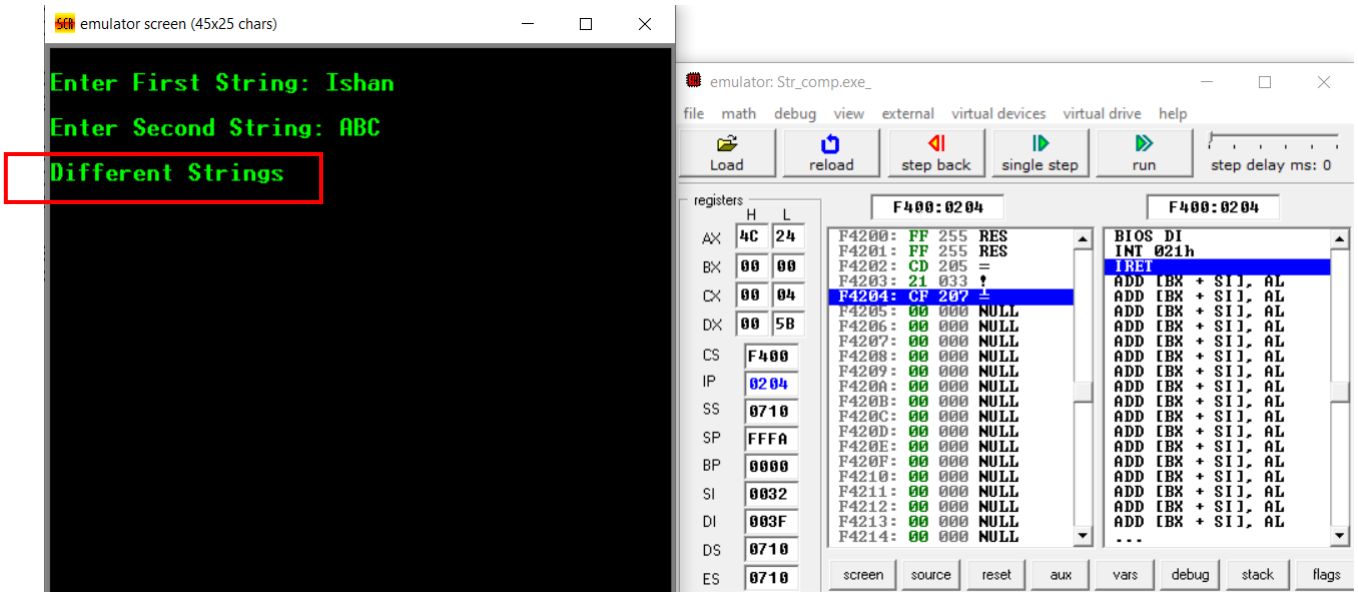
file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
CR EQU 13
LF EQU 10

M1 DB CR,LF,"Enter First String: $"
M2 DB CR,LF,LF,"Enter Second String: $"

STR1 DB 0BH,12 DUP(?)
STR2 DB 0BH,12 DUP(?)

SM DB CR,LF,LF,"Same Strings $"
FM DB CR,LF,LF,"Different Strings $"
DATA ENDS

CODE SEGMENT
START:
    MOV AX,DATA
    MOV DS,AX
    MOV ES,AX
    LEA DX,M1
    MOV AH,09
    INT 21H

    MOV DX,OFFSET STR1
    MOV AH,0AH
    INT 21H

    LEA DX,M2
    MOV AH,09
    INT 21H

    MOV DX,OFFSET STR2
    MOV AH,0AH
    INT 21H
```

Output:





3. String concatenation –

Input:

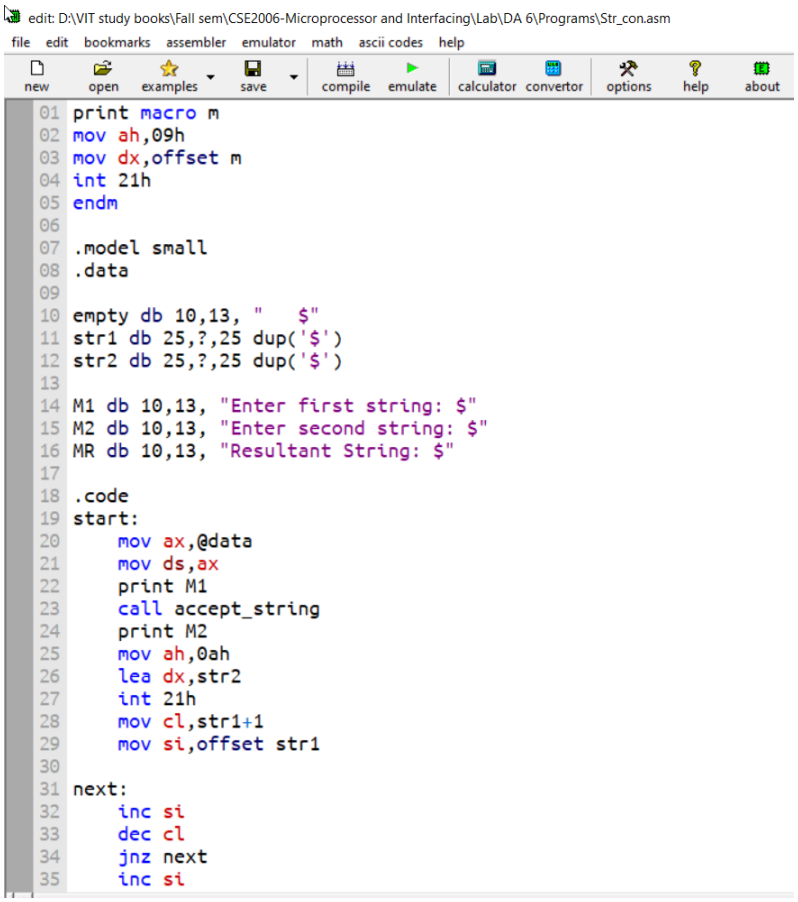
User input of two strings and result will be combined single string as output. Used interrupt is INT21H / AH=09h to read string with OFFSET in DL with the help of AH=0Ah as buffered input of string.

Program:

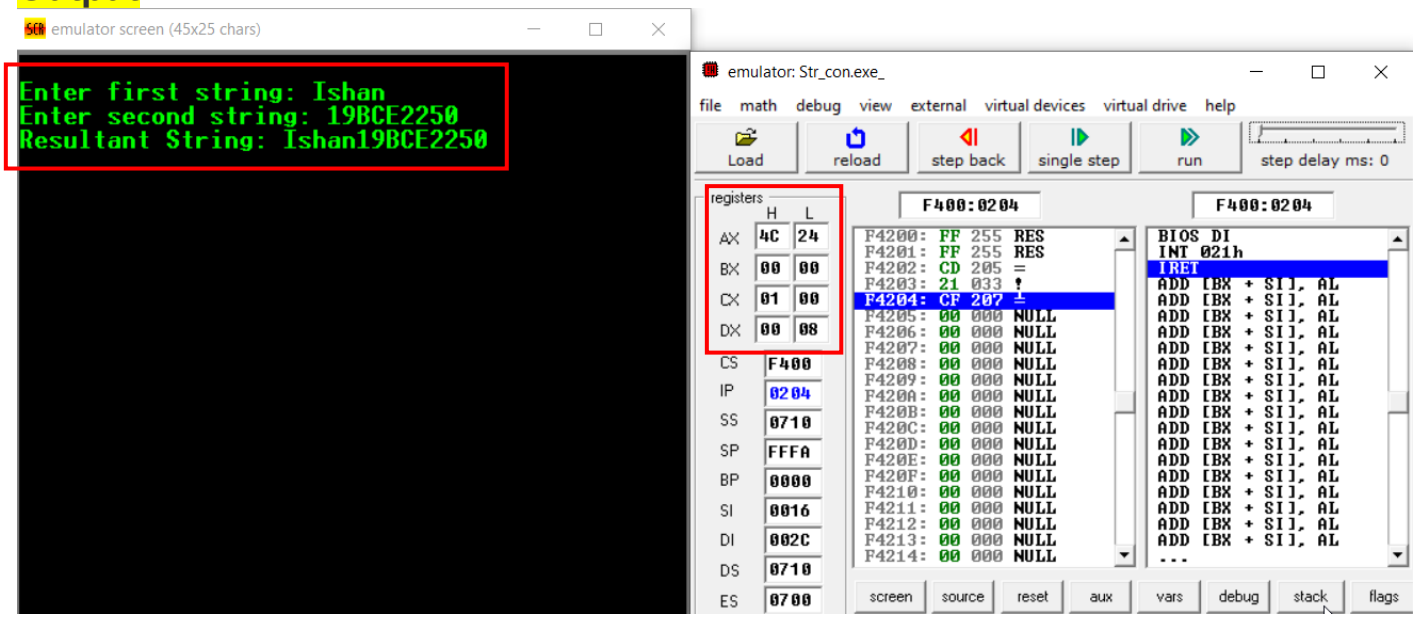
```
print macro m
mov ah,09h
mov dx,offset m
int 21h
endm

.model small
.data

empty db 10,13, "   $"
str1 db 25,?,25 dup('$')
str2 db 25,?,25 dup('$')

M1 db 10,13, "Enter first string: $"
M2 db 10,13, "Enter second string: $"
MR db 10,13, "Resultant String: $"

.code
start:
   mov ax,@data
   mov ds,ax
   print M1
    call accept_string
    print M2
    mov ah,0ah
```

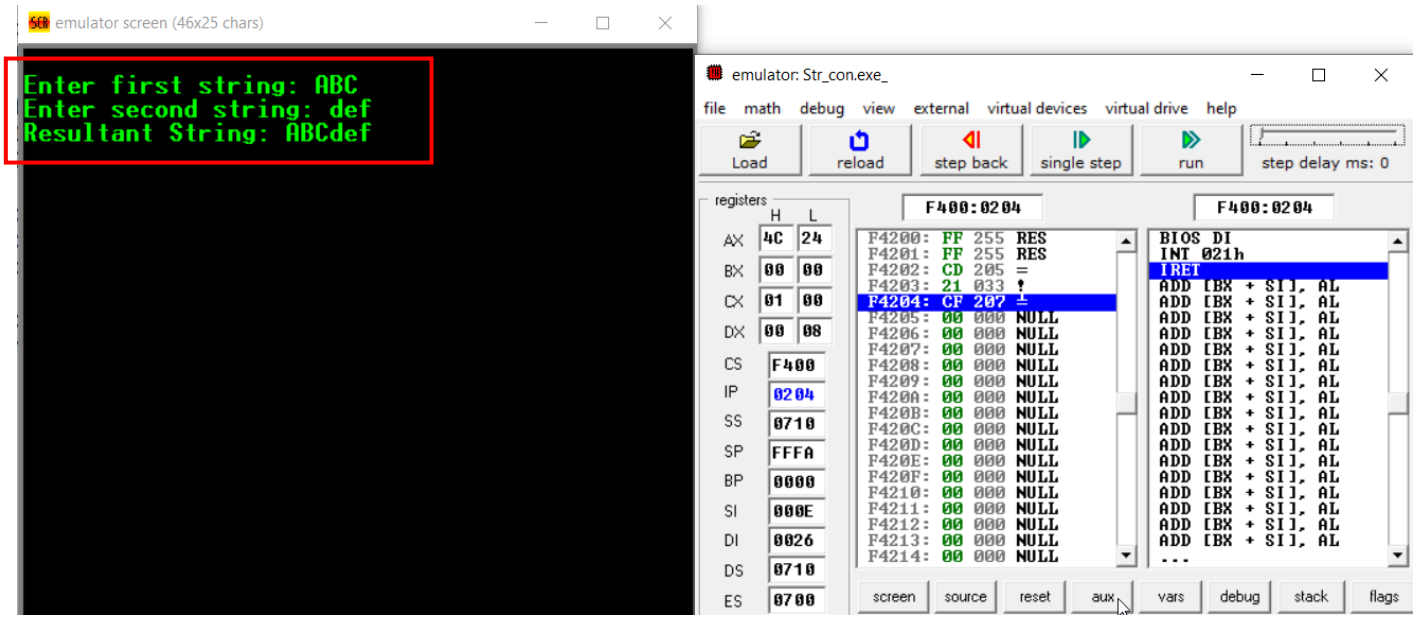
```
    lea dx,str2
    int 21h
    mov cl,str1+1
    mov si,offset str1

next:
    inc si
    dec cl
    jnz next
    inc si
    inc si
    mov di,offset str2
    inc di
    inc di
    mov cl,str2+1

move_next:
    mov al,[di]
    mov [si],al
    inc si
    inc di
    dec cl
    jnz move_next
    print MR
    print str1+2

exit:
    mov ah,4ch
    int 21h
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
mov dx,offset str1
int 21h
ret

accept_string endp
end start
end
```



Output:





4. String Palindrome –

Input:

User input of string and result will be String is palindrome or not message as output.

Used interrupt is INT21H / AH=09h to read string with OFFSET in DL with the help of AH=0Ah as buffered input of string.

Program:

```
org 100h
jmp start

m1:
  STR1 db 'ishan'
  STR_SIZE = $ - m1
  db 0Dh,0Ah,'$'
```

```
start:
   mov ah, 9
   mov dx, offset STR1
   int 21h

lea di, STR1
mov si, di
add si, STR_SIZE
dec si
mov cx, STR_SIZE
cmp cx, 1
je Is_Palindrome
shr cx, 1

next_char:
    mov al, [di]
    mov bl, [si]
    cmp al, bl
    jne Not_Palindrome
    inc di
    dec si
loop next_char

Is_Palindrome:

   mov ah, 9
   mov dx, offset MSG1
   int 21h
jmp stop

Not_Palindrome:
   mov ah, 9
   mov dx, offset MSG2
   int 21h

stop:
   mov ah, 0
   int 16h

ret
MSG1 db "  String Is Palindrome $"
MSG2 db "  String Is Not a Palindrome !$"
```

edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 6\Programs\Str_pallin.asm

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
org 100h
jmp start

m1:
   STR1 db 'ishan'
   STR_SIZE = $ - m1
   db 0Dh,0Ah,'$'

start:
   mov ah, 9
   mov dx, offset STR1
   int 21h

lea di, STR1
mov si, di
add si, STR_SIZE
dec si
mov cx, STR_SIZE
cmp cx, 1
je Is_Palindrome
shr cx, 1

next_char:
    mov al, [di]
    mov bl, [si]
    cmp al, bl
    jne Not_Palindrome
    inc di
    dec si
loop next_char

Is_Palindrome:

   mov ah, 9
   mov dx, offset M1
```

Output:

emulator screen (37x25 chars)

```
ishan
  String Is Not a Palindrome !
```

emulator: Str_pallin.com_

file math debug view external virtual devices virtual drive help

Load reload step back waiting for input stop step delay ms: 0

registers

	H	L
AX	00	24
BX	00	6E
CX	00	02
DX	01	5D
CS	F400	
IP	01C0	
SS	0700	
SP	FFF8	
BP	0000	
SI	0106	
DI	0102	
DS	0700	
ES	0700	

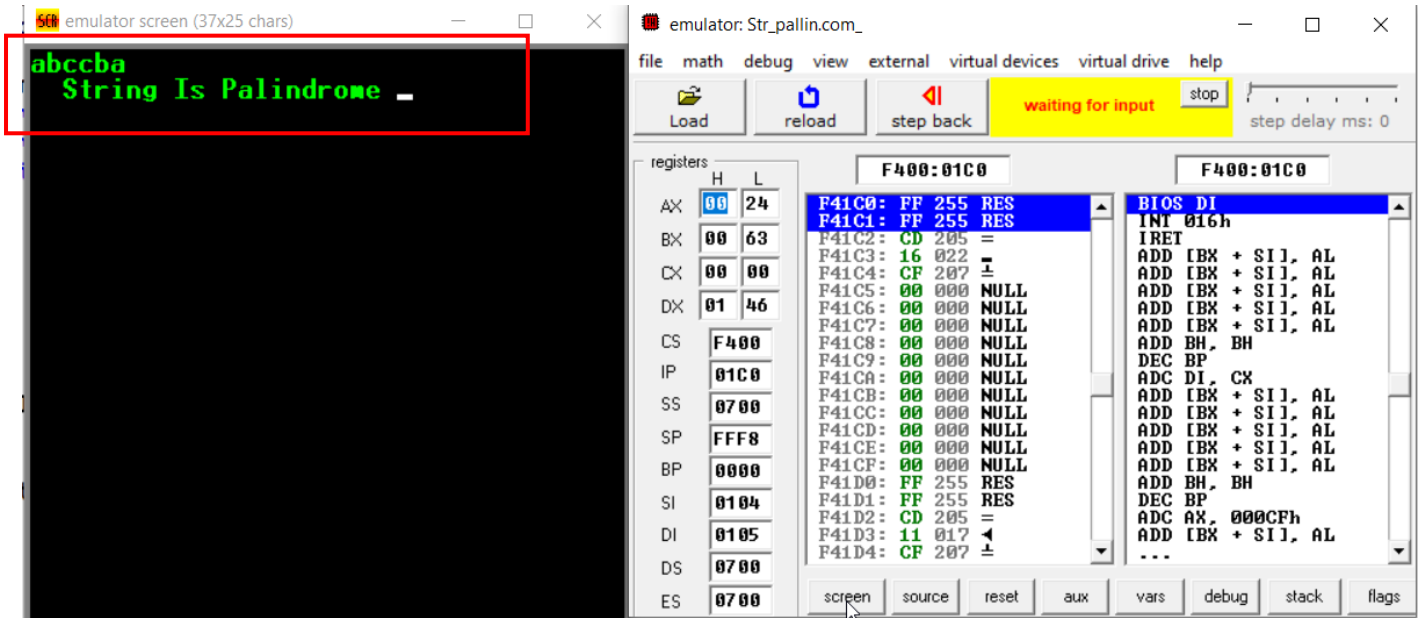
F400:01C0

```
F41C0: FF 255 RES
F41C1: FF 255 RES
F41C2: CD 205 =
F41C3: 16 022 ▬
F41C4: CF 207 ╧
F41C5: 00 000 NULL
F41C6: 00 000 NULL
F41C7: 00 000 NULL
F41C8: 00 000 NULL
F41C9: 00 000 NULL
F41CA: 00 000 NULL
F41CB: 00 000 NULL
F41CC: 00 000 NULL
F41CD: 00 000 NULL
F41CE: 00 000 NULL
F41CF: 00 000 NULL
F41D0: FF 255 RES
F41D1: FF 255 RES
F41D2: CD 205 =
F41D3: 11 017 ◄
F41D4: CF 207 ╧
```

F400:01C0

```
BIOS DI
INT 016h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
ADC DI, CX
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
ADC AX, 000CFh
ADD [BX + SI], AL
...
```

screen source reset aux vars debug stack flags



5. String Password Checking –

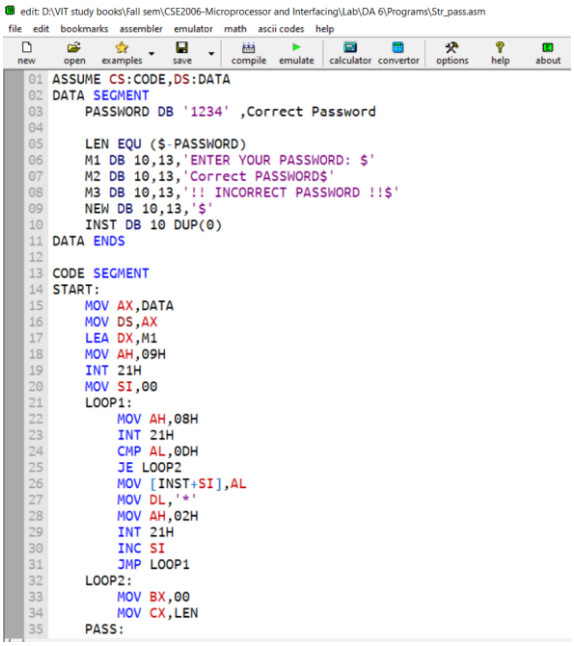
Input:

User input of string and result will be password is correct or not by checking with default saved password string.
Used interrupt is INT21H / AH=09h to read string with OFFSET in DL with the help of AH=0Ah as buffered input of string.

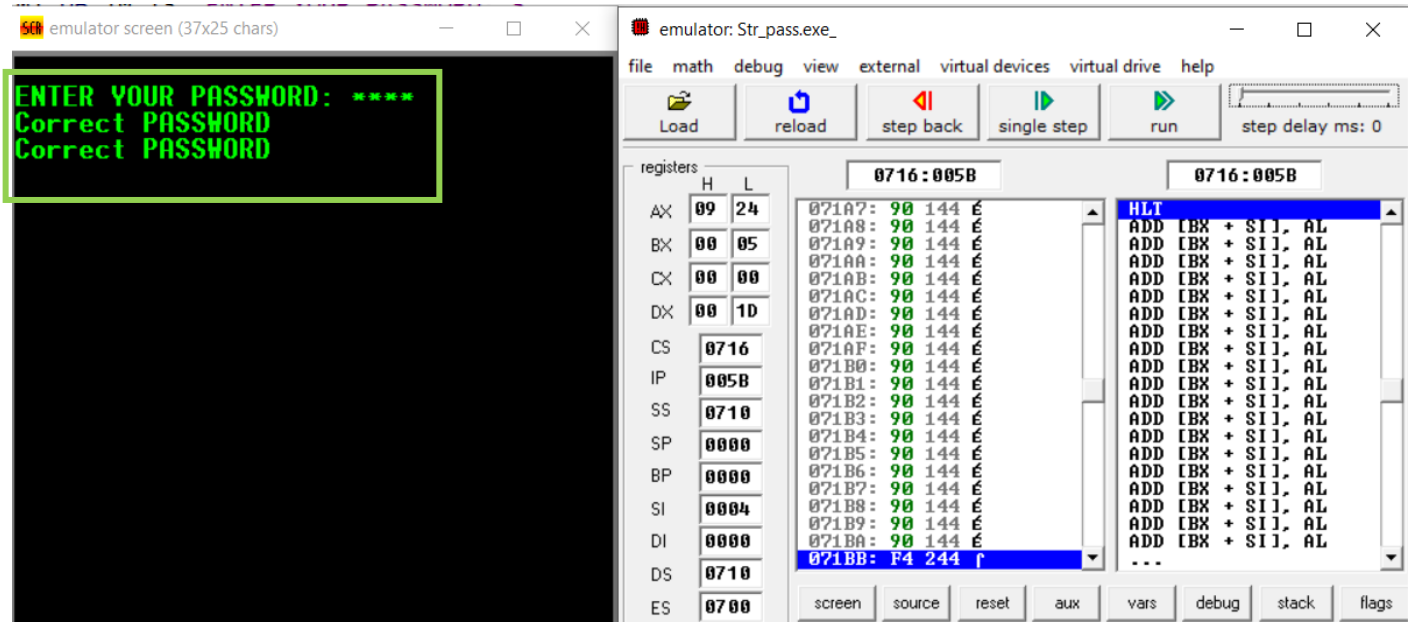
Program:

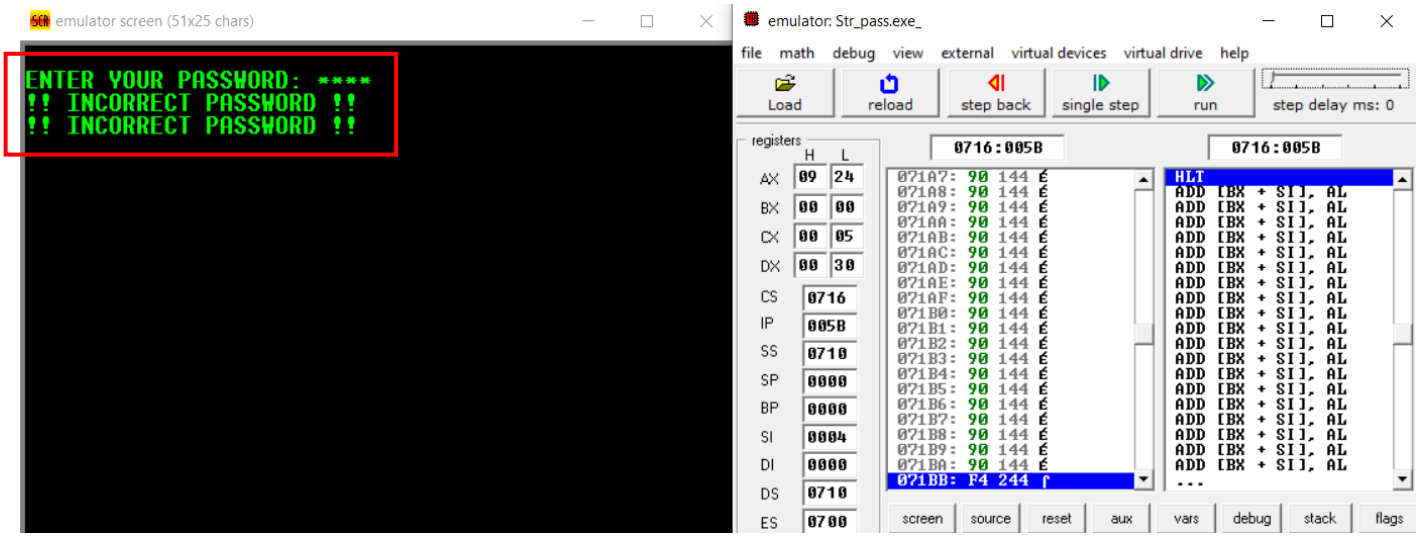
```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   PASSWORD DB '1234' ,Correct Password

   LEN EQU ($-PASSWORD)
   M1 DB 10,13,'ENTER YOUR PASSWORD: $'
   M2 DB 10,13,'Correct PASSWORD$'
   M3 DB 10,13,'!! INCORRECT PASSWORD !!$'
   NEW DB 10,13,'$'
   INST DB 10 DUP(0)
DATA ENDS

CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   LEA DX,M1
   MOV AH,09H
   INT 21H
   MOV SI,00
   LOOP1:
      MOV AH,08H
      INT 21H
```

```
        CMP AL,0DH
        JE LOOP2
        MOV [INST+SI],AL
        MOV DL,'*'
        MOV AH,02H
        INT 21H
        INC SI
        JMP LOOP1
    LOOP2:
        MOV BX,00
        MOV CX,LEN
    PASS:
        MOV AL,[INST+BX]
        MOV DL,[PASSWORD+BX]
        CMP AL,DL
        JNE FAIL
        INC BX
        LOOP PASS
        LEA DX,M2
        MOV AH,09H
        INT 21H
        JMP FINISH
     FAIL:
        LEA DX,M3
        MOV AH,09H
        INT 21H
        FINISH:
INT 21H
CODE ENDS
END START
END
```



Output:


emulator screen (37x25 chars)
ENTER YOUR PASSWORD: ****
Correct PASSWORD
Correct PASSWORD
emulator: Str_pass.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX 09 24
BX 00 05
CX 00 00
DX 00 1D
CS 0716
IP 005B
SS 0710
SP 0000
BP 0000
SI 0004
DI 0000
DS 0710
ES 0700
0716:005B
071A7: 90 144 É
071A8: 90 144 É
071A9: 90 144 É
071AA: 90 144 É
071AB: 90 144 É
071AC: 90 144 É
071AD: 90 144 É
071AE: 90 144 É
071AF: 90 144 É
071B0: 90 144 É
071B1: 90 144 É
071B2: 90 144 É
071B3: 90 144 É
071B4: 90 144 É
071B5: 90 144 É
071B6: 90 144 É
071B7: 90 144 É
071B8: 90 144 É
071B9: 90 144 É
071BA: 90 144 É
071BB: F4 244 ⌠
HLT
ADD [BX + SI], AL
screen source reset aux vars debug stack flags


emulator screen (51x25 chars)
ENTER YOUR PASSWORD: ****
!! INCORRECT PASSWORD !!
!! INCORRECT PASSWORD !!
emulator: Str_pass.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX 09 24
BX 00 00
CX 00 05
DX 00 30
CS 0716
IP 005B
SS 0710
SP 0000
BP 0000
SI 0004
DI 0000
DS 0710
ES 0700
0716:005B
HLT
ADD [BX + SI], AL
screen source reset aux vars debug stack flags