

Assembly Level Program 1a – Binary Search

Write an Assembly Level Program to search a key element in a list of 'n' 16-bit numbers using the Binary Search Algorithm.

Program

```
.model SMALL
```

```
.data
```

```
    ARRAY    dW    1234h, 2345h, 3456h, 4567h, 5678h, 6789h
```

```
    LEN      dW    ($-ARRAY)/2
```

```
    KEY      dW    6789h
```

```
    STR1    dB    10, 13, 'Element Found at Position '
```

```
    POS     dB    ?, 10, 13, '$'
```

```
    STR2    dB    10, 13, 'Element Not Found!$'
```

```
.code
```

```
    MOV AX, @DATA
```

```
    MOV DS, AX
```

```
    MOV AX, 00h
```

```
    MOV CX, LEN
```

```
    MOV DX, KEY
```

Search:

```
    CMP CX, AX
```

```
    JB NotFound
```

```
    MOV BX, CX
```

```
    ADD BX, AX
```

SHR BX, 01h ; Divides by 2

MOV SI, BX

SHL SI, 01h ; Multiplies with 2

CMP ARRAY[SI], DX

JB newLow

JE Found

CMP BX, 00h

JE NotFound

DEC BX

MOV CX, BX

JMP Search

newLow:

INC BX

MOV AX, BX

JMP Search

Found:

ADD BL, '1'

MOV POS, BL

LEA DX, STR1

JMP Exit

NotFound:

LEA DX, STR2

Exit:

MOV AH, 09h

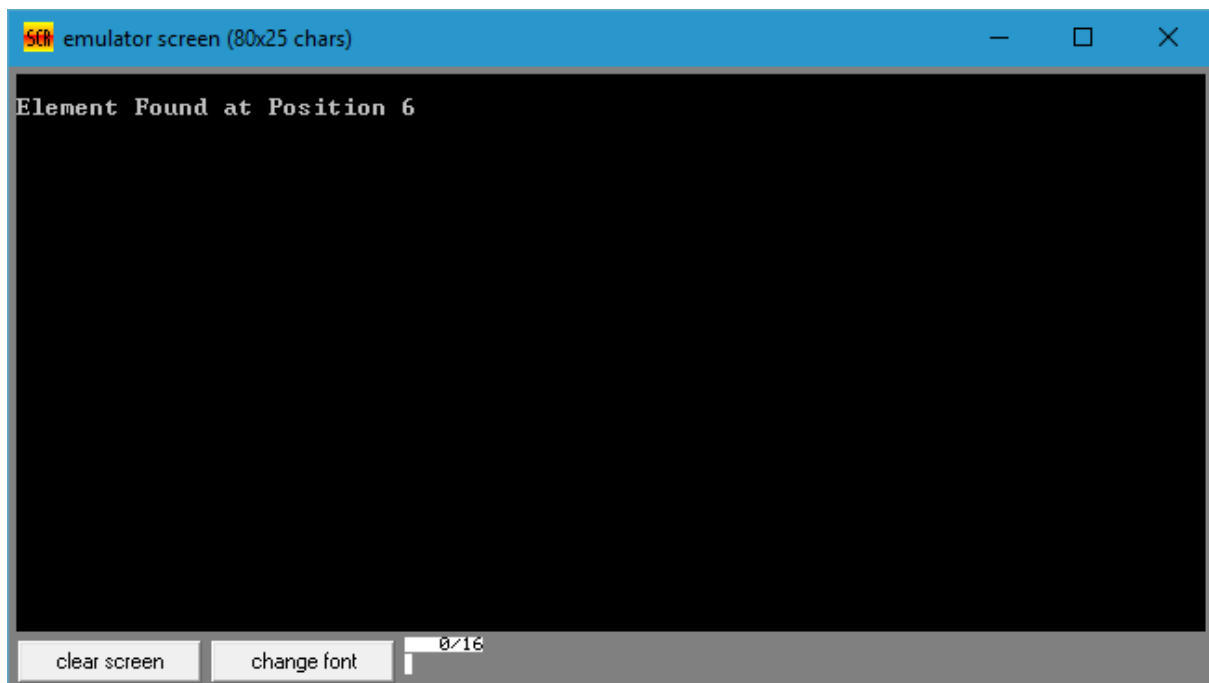
INT 21h

MOV AH, 4Ch

INT 21h

END

Output



Assembly Level Program 2a – Reading & Printing String

Write 2 ALP modules stored in two different files; one module is to read a character from the keyboard and the other one is to display a character. Use the above two modules to read a string of characters from the keyboard terminated by the carriage return and print the string on the display in the next line.

PrintCharacter.inc

PRINTCH MACRO CHAR

MOV DL, CHAR

MOV AH, 02h

INT 21h

ENDM

ReadCharacter.inc

READCH MACRO

MOV AH, 01h

INT 21h

ENDM

Program

.model SMALL

Include ReadCharacter.inc

Include PrintCharacter.inc

.data

LOC dB 100 DUP(0)

STR1 dB 10, 13, 'Enter a String: \$'

STR2 dB 10, 13, 'Entered String is: \$'

.code

MOV AX, @DATA

MOV DS, AX

MOV CX, 00h

LEA DX, STR1

MOV AH, 09h

INT 21h

LEA SI, LOC

Read:

READCH

CMP AL, 0Dh

JE Display

MOV [SI], AL

INC SI

INC CL

JMP Read

Display:

LEA DX, STR2

MOV AH, 09h

INT 21h

LEA SI, LOC

Print:

PRINTCH [SI]

INC SI

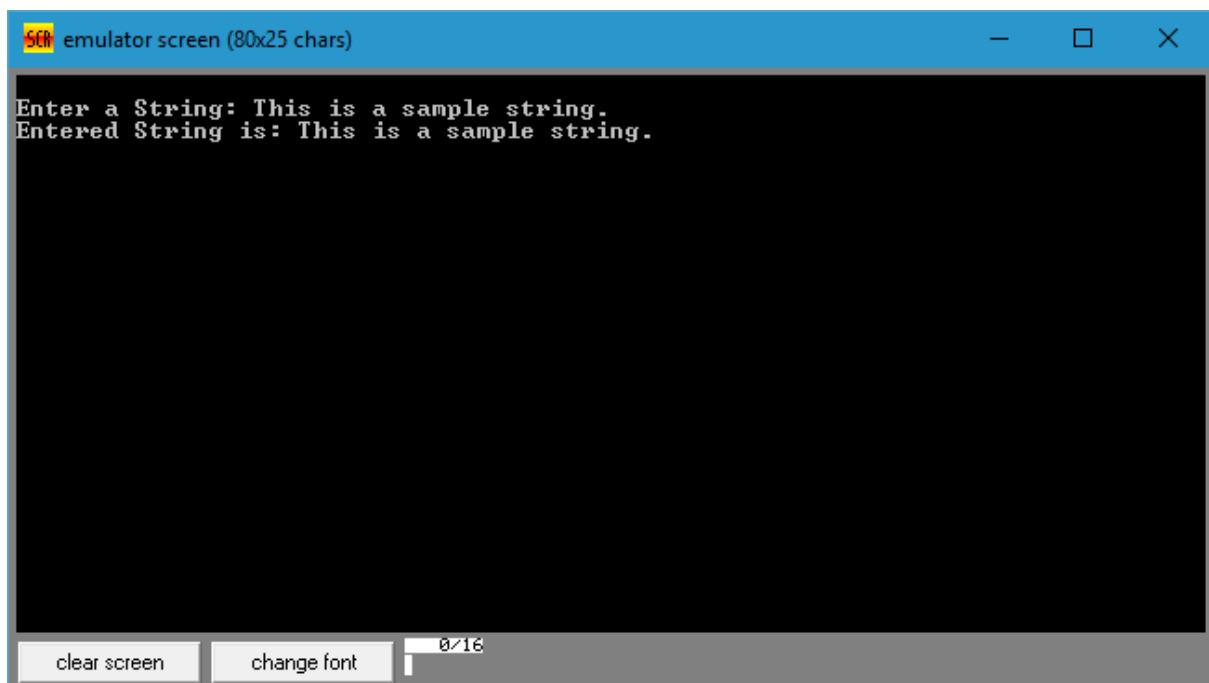
Loop Print

MOV AH, 4Ch

INT 21h

END

Output



Assembly Level Program 3a – Bubble Sort

Write an Assembly Level Program to sort a given set of 'n' numbers in ascending and descending orders using the Bubble Sort algorithm.

Program

```
.model SMALL
```

```
.data
```

```
    ARRAY      dB      05h, 07h, 06h, 04h, 10h, 09h
    LEN         dB      $-ARRAY
```

```
.code
```

```
    MOV AX, @DATA
    MOV DS, AX
```

```
    MOV CX, 00h
    MOV CL, LEN
    DEC CL
```

```
OuterLoop:
```

```
    MOV BX, CX
    LEA SI, ARRAY
```

```
InnerLoop:
```

```
    MOV AL, [SI]
    INC SI
    CMP [SI], AL
    JBE NoSwap

    XCHG [SI], AL
```

MOV [SI-1], AL

NoSwap:

DEC BX

JNZ InnerLoop

LOOP OuterLoop

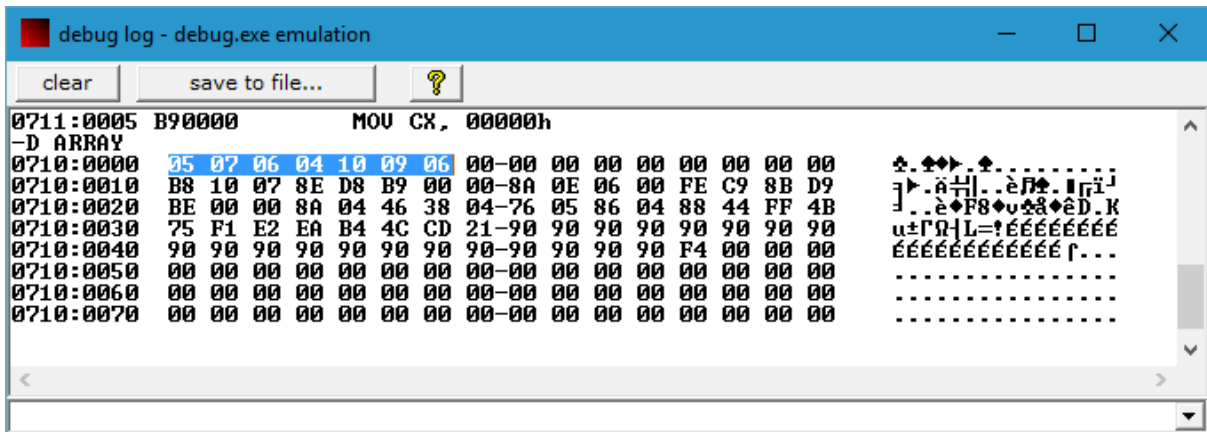
Exit:

MOV AH, 4Ch

INT 21h

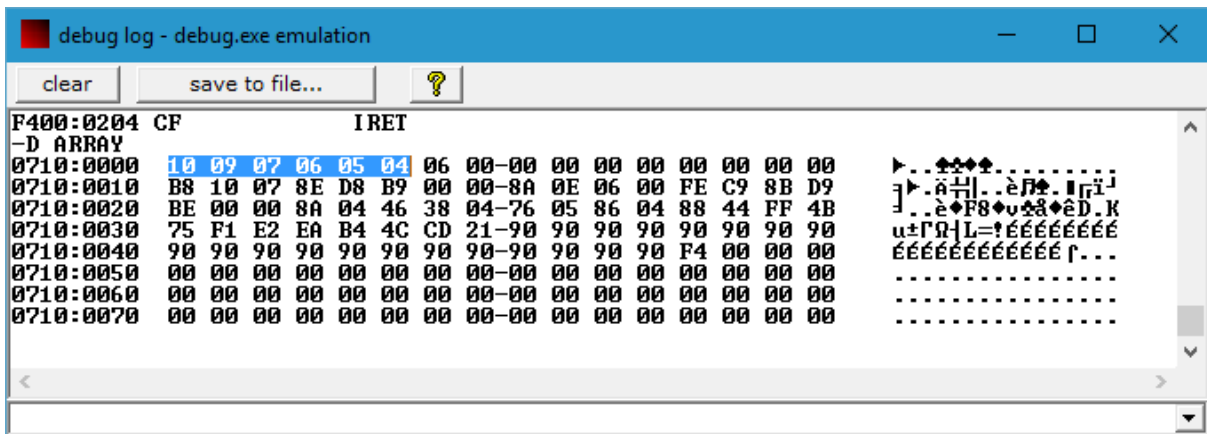
END

Output



```

debug log - debug.exe emulation
clear save to file... ?
0711:0005 B90000      MOV CX, 000000h
-D ARRAY
0710:0000 05 07 06 04 10 09 06 00-00 00 00 00 00 00 00 00 00  05 07 06 04 10 09 06
0710:0010 B8 10 07 8E D8 B9 00 00-8A 0E 06 00 FE C9 8B D9  10 07 8E D8 B9 00
0710:0020 BE 00 00 8A 04 46 38 04-76 05 86 04 88 44 FF 4B  00 00 8A 04 46 38
0710:0030 75 F1 E2 EA B4 4C CD 21-90 90 90 90 90 90 90 90  75 F1 E2 EA B4 4C
0710:0040 90 90 90 90 90 90 90 90-90 90 90 90 F4 00 00 00  90 90 90 90 90 90
0710:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  00 00 00 00 00 00
0710:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  00 00 00 00 00 00
0710:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  00 00 00 00 00 00
  
```



```

debug log - debug.exe emulation
clear save to file... ?
F400:0204 CF      IRET
-D ARRAY
0710:0000 10 09 07 06 05 04 06 00-00 00 00 00 00 00 00 00 00  10 09 07 06 05 04
0710:0010 B8 10 07 8E D8 B9 00 00-8A 0E 06 00 FE C9 8B D9  10 07 8E D8 B9 00
0710:0020 BE 00 00 8A 04 46 38 04-76 05 86 04 88 44 FF 4B  00 00 8A 04 46 38
0710:0030 75 F1 E2 EA B4 4C CD 21-90 90 90 90 90 90 90 90  75 F1 E2 EA B4 4C
0710:0040 90 90 90 90 90 90 90 90-90 90 90 90 F4 00 00 00  90 90 90 90 90 90
0710:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  00 00 00 00 00 00
0710:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  00 00 00 00 00 00
0710:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  00 00 00 00 00 00
  
```


Assembly Level Program 4a – Bubble Sort

Write an Assembly Level Program to read an alphanumeric character and display its equivalent ASCII code at the center of the screen.

Program

```
.model SMALL
```

```
CLRSCR MACRO
```

```
    MOV AH, 00h
```

```
    MOV AL, 03h
```

```
    INT 10h
```

```
ENDM
```

```
SETCURSOR MACRO
```

```
    MOV AH, 02h
```

```
    MOV BH, 00h
```

```
    MOV DH, 12d
```

```
    MOV DL, 39d
```

```
    INT 10H
```

```
ENDM
```

```
.data
```

```
    MSG1 dB    10, 13, 'Enter an alphanumeric character: $'
```

```
.code
```

```
    MOV AX, @DATA
```

```
    MOV DS, AX
```

```
    CLRSCR
```

```
    ; Print Message in Data Segment
```

LEA DX, MSG1

MOV AH, 09h

INT 21h

; Read Character from User

MOV AH, 01h

INT 21h

MOV AH, 00h

MOV BX, 10d

PUSH BX

Conversion:

MOV DX, 00h

DIV BX

PUSH DX

CMP AX, 00h

JNE Conversion

SETCURSOR

Display:

POP DX

CMP DX, 10

JE Exit

ADD DL, 30h

```
MOV AH, 02h
```

```
INT 21h
```

```
JMP Display
```

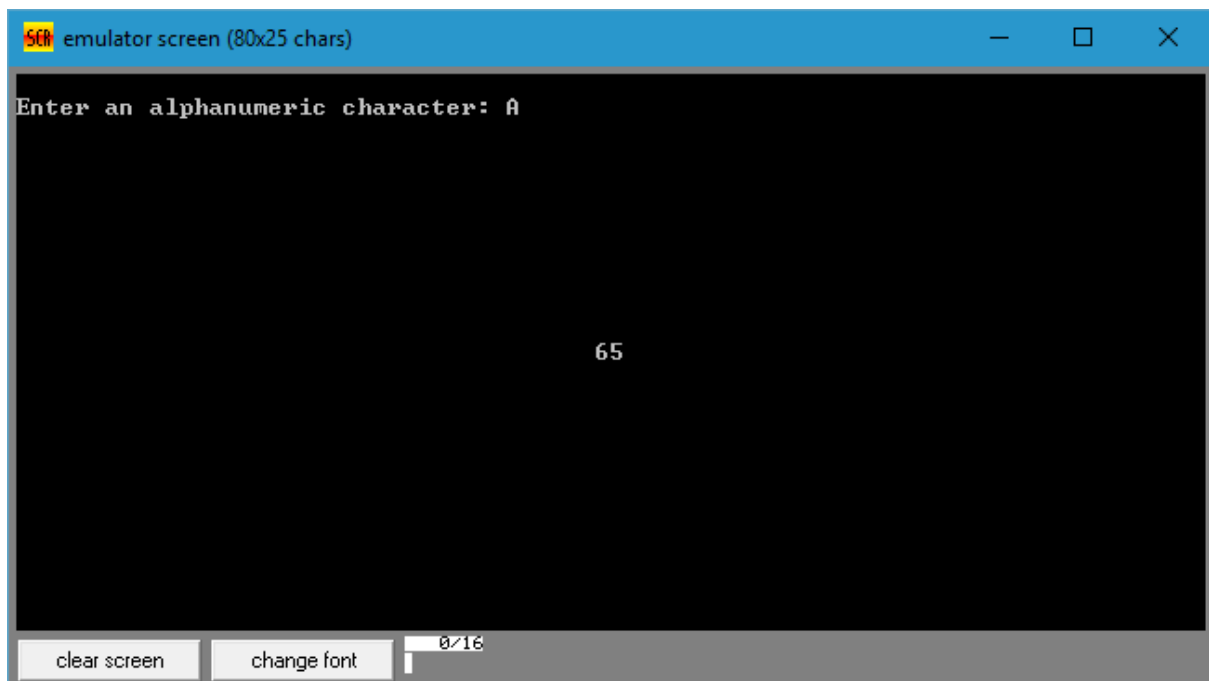
Exit:

```
MOV AH, 4Ch
```

```
INT 21h
```

```
END
```

Output



Assembly Level Program 5a – Palindrome Checker

Write an ALP to reverse a given string and check whether it is a palindrome or not.

Program

```
.model SMALL
```

```
PRINTSTR MACRO MSG
```

```
    LEA DX, MSG
```

```
    MOV AH, 09h
```

```
    INT 21h
```

```
ENDM
```

```
READSTR    MACRO BUF
```

```
    LEA DX, BUF
```

```
    MOV AH, 0Ah
```

```
    INT 21h
```

```
ENDM
```

```
.data
```

```
    BUF1  dB    20d
```

```
    LEN1  dB    ?
```

```
    STR1  dB    20d    DUP(0)
```

```
    RSTR  dB    20d    DUP(0)
```

```
    MSG1  dB    10, 13, 'Enter a String: $'
```

```
    MSG2  dB    10, 13, 'String is a Palindrome!$'
```

```
    MSG3  dB    10, 13, 'String is not a Palindrome!$'
```

```
.code
```

MOV AX, @DATA

MOV DS, AX

MOV ES, AX

PRINTSTR MSG1

READSTR BUF1

LEA SI, STR1

DEC SI

MOV CX, 00h

MOV CL, LEN1

ADD SI, CX

MOV DI, SI

LEA SI, RSTR

CopyString:

MOV AL, [DI]

MOV [SI], AL

INC SI

DEC DI

LOOP CopyString

LEA SI, STR1

LEA DI, RSTR

MOV CX, 00h

MOV CL, LEN1

CLD ; Clear Direction Flag

REPE CMPSB

JE Palindrome

PRINTSTR MSG3

JMP Exit

Palindrome:

PRINTSTR MSG2

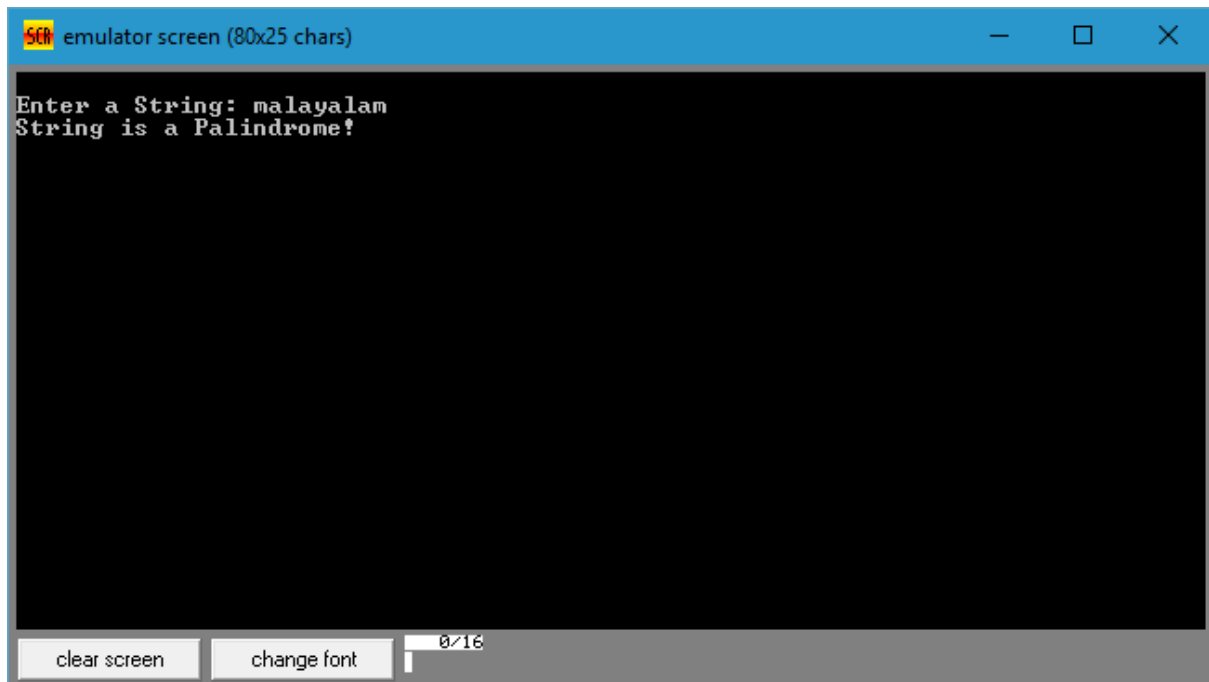
Exit:

MOV AH, 4Ch

INT 21h

END

Output



Assembly Level Program 6a – String Equality

Write an ALP to read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriated messages. Also display the length of the stored strings.

Program

```
.model SMALL
```

```
PRINTSTR MACRO MSG
```

```
    LEA DX, MSG
```

```
    MOV AH, 09h
```

```
    INT 21h
```

```
ENDM
```

```
READSTR MACRO BUF
```

```
    LEA DX, BUF
```

```
    MOV AH, 0Ah
```

```
    INT 21h
```

```
ENDM
```

```
.data
```

```
    BUF1  dB    20d
```

```
    LEN1  dB    ?
```

```
    STR1  dB    20d DUP(0)
```

```
    BUF2  dB    20d
```

```
    LEN2  dB    ?
```

```
    STR2  dB    20d DUP(0)
```

```
    MSG1  dB    10, 13, 'Enter String 1: $'
```

MSG2 dB 10, 13, 'Enter String 2: \$'

MSG3 dB 10, 13, 'Length of String 1: \$'

MSG4 dB 10, 13, 'Length of String 2: \$'

MSG5 dB 10, 13, 'Strings are Equal!\$'

MSG6 dB 10, 13, 'Strings are Not Equal!\$'

.code

MOV AX, @DATA

MOV DS, AX

MOV ES, AX

PRINTSTR MSG1

READSTR BUF1

PRINTSTR MSG2

READSTR BUF2

MOV CL, LEN1

CMP CL, LEN2

JNE NotEqual

LEA SI, STR1

LEA DI, STR2

MOV CH, 00h

MOV CL, LEN1

CLD ; Clear Direction Flag

REPE CMPSB ; Compare String Byte-by-Byte

JE Equal

NotEqual:

PRINTSTR MSG6

JMP Exit

Equal:

PRINTSTR MSG5

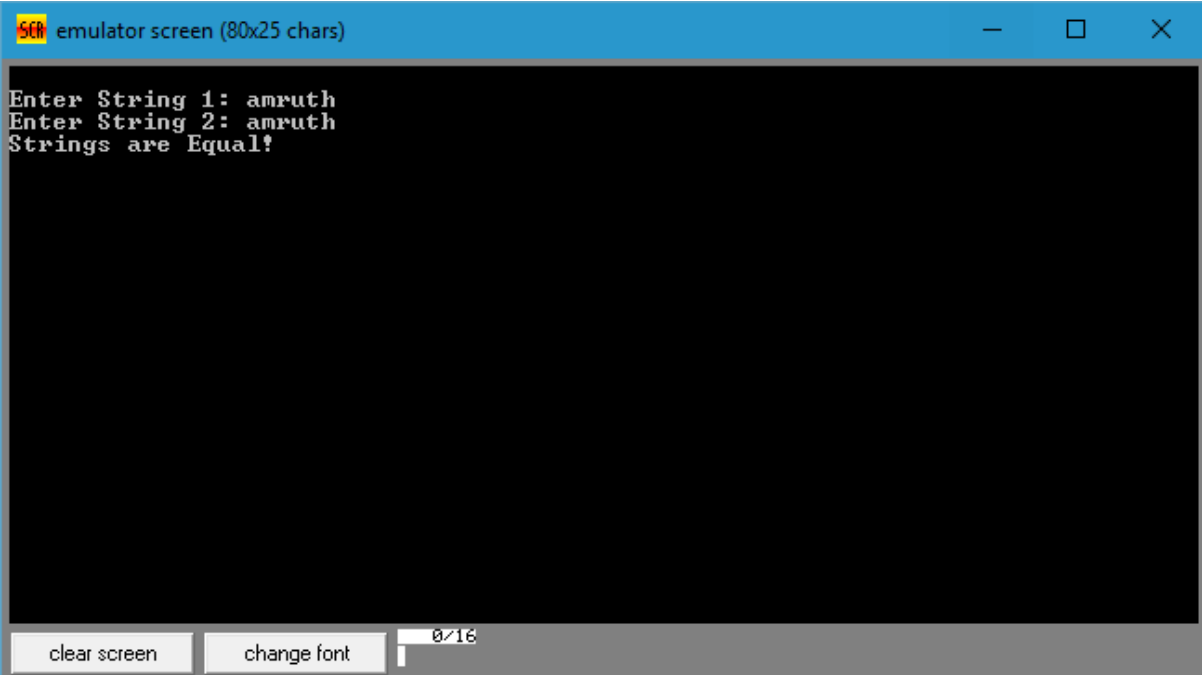
Exit:

MOV AH, 4Ch

INT 21h

END

Output



Assembly Level Program 7a – What Is Your Name?

Write an Assembly Level Program to read your name from the keyboard and display it at a specified location on the screen in front of the message What is your name?

You must clear the entire screen before display.

Program

```
.model SMALL
```

```
READCH MACRO LOC
```

```
    MOV AH, 01h
```

```
    INT 21h
```

```
    MOV LOC, AL
```

```
ENDM
```

```
CLRSCR MACRO
```

```
    MOV AH, 00h
```

```
    MOV AL, 03h
```

```
    INT 10h
```

```
ENDM
```

```
SETCURSOR MACRO
```

```
    MOV AH, 02h
```

```
    MOV BH, 00h
```

```
    MOV DH, 2
```

```
    MOV DL, 20
```

```
    INT 10h
```

```
ENDM
```

```
.data
```

```
    MSG1 dB    10, 13, 'Enter your name: $'
```

MSG2 dB 10, 13, 'What is your name? \$'

ARRAY dB 40h DUP(?)

.code

MOV AX, @DATA

MOV DS, AX

MOV SI, 00h

LEA DX, MSG1

MOV AH, 09h

INT 21h

ReadName:

READCH ARRAY[SI]

INC SI

CMP AL, 13

JNZ ReadName

MOV ARRAY[SI], '\$'

CLRSCR

SETCURSOR

LEA DX, MSG2

MOV AH, 09h

INT 21h

MOV SI, 00h

DisplayName:

```
LEA DX, ARRAY[SI]
```

```
MOV AH, 09h
```

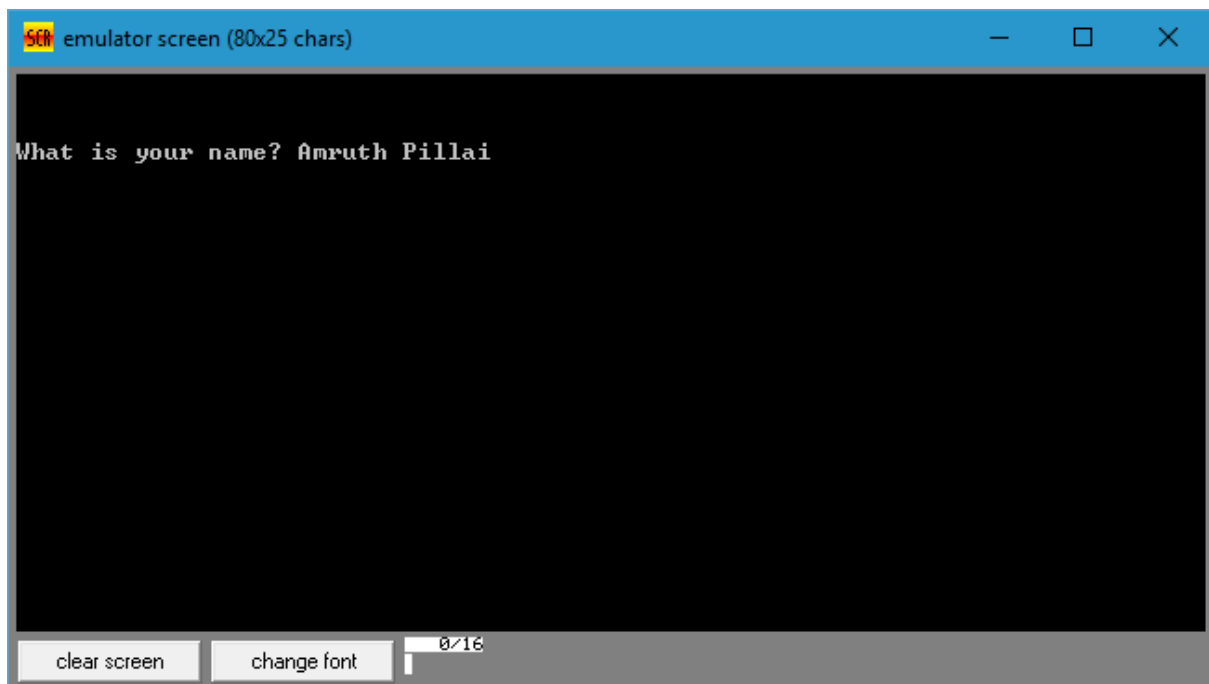
```
INT 21h
```

```
MOV AH, 4Ch
```

```
INT 21h
```

```
END
```

Output



Assembly Level Program 8a – Calculate NCR

Write an Assembly Level Program to compute nCr using recursive procedure. Assume that 'n' and 'r' are non-negative integers.

Program

```
.model SMALL
```

```
.data
```

```
    N          dB      05d
    R          dB      02d
    NCR        dW      ?
```

```
.code
```

```
    MOV AX, @DATA
```

```
    MOV DS, AX
```

```
    MOV AX, 00h
```

```
    MOV AL, N
```

```
    MOV BL, R
```

```
    MOV NCR, 00h
```

```
    CALL NCRProcedure
```

```
    MOV AH, 4Ch
```

```
    INT 21h
```

```
NCRProcedure PROC
```

```
    CMP AX, BX
```

```
    JE IncrementBy1
```

```
    CMP BX, 00h
```

JE IncrementBy1

CMP BX, 01h

JE IncrementByN

DEC AX

CMP AX, BX

JE IncrementBoth

PUSH AX

PUSH BX

CALL NCRProcedure

POP BX

POP AX

DEC BX

PUSH AX

PUSH BX

CALL NCRProcedure

POP BX

POP AX

RET

IncrementBy1:

INC NCR

RET

IncrementByN:

ADD NCR, AX

RET

IncrementBoth:

ADD NCR, AX

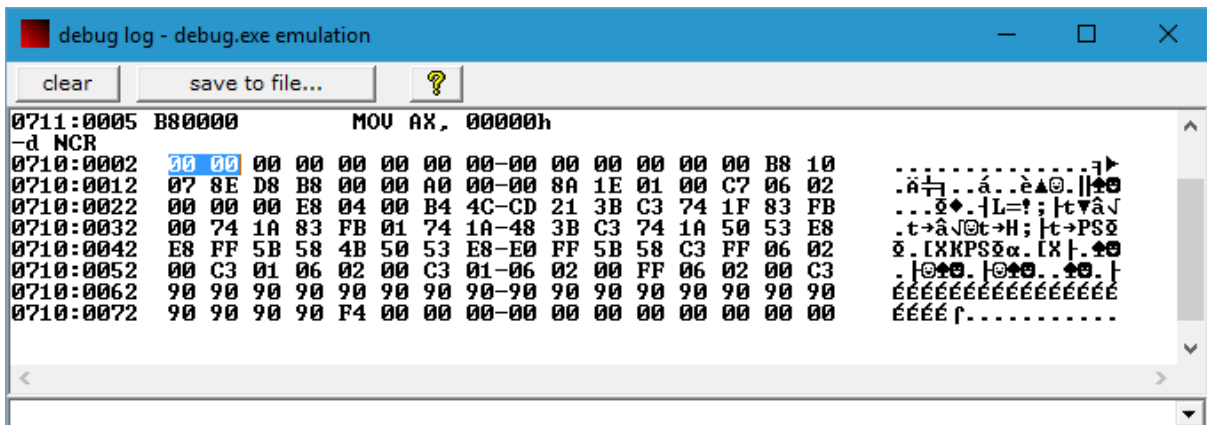
INC NCR

RET

NCRProcedure ENDP

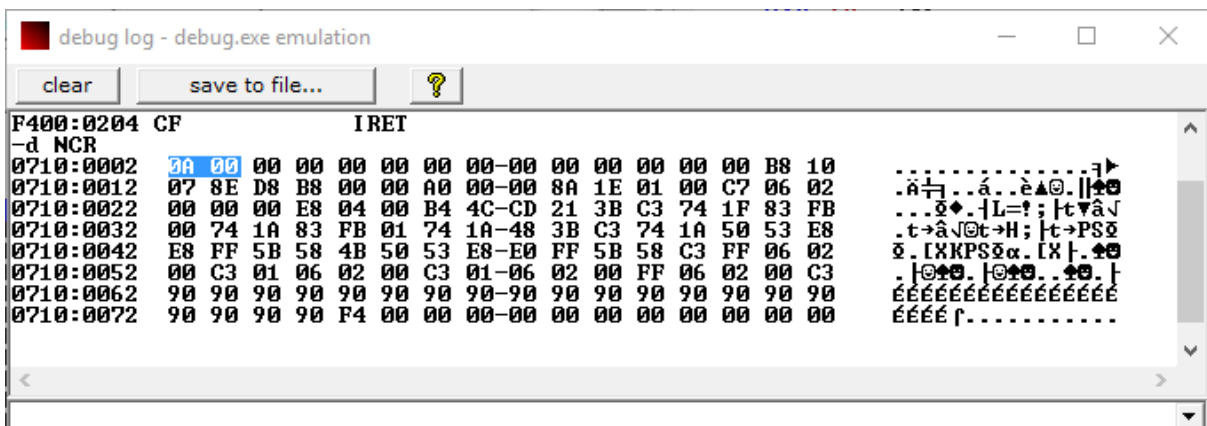
END

Output



```

debug log - debug.exe emulation
clear save to file... ?
0711:0005 B80000 MOU AX, 00000h
-d NCR
0710:0002 00 00 00 00 00 00 00-00 00 00 00 00 00 B8 10 .....
0710:0012 07 8E D8 B8 00 00 A0 00-00 8A 1E 01 00 C7 06 02 .A+...a...eA@.||
0710:0022 00 00 00 E8 04 00 B4 4C-CD 21 3B C3 74 1F 83 FB ..d+.}L=;!;|tva√
0710:0032 00 74 1A 83 FB 01 74 1A-48 3B C3 74 1A 50 53 E8 .t→a√@t→H;|t→PSd
0710:0042 E8 FF 5B 58 4B 50 53 E8-E0 FF 5B 58 C3 FF 06 02 d.[XKPSdα.[X|..d
0710:0052 00 C3 01 06 02 00 C3 01-06 02 00 FF 06 02 00 C3 .|@+@.|@+@..+@.|
0710:0062 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90 éééééééééééééééé
0710:0072 90 90 90 90 F4 00 00 00-00 00 00 00 00 00 00 éééééééééééééééé
  
```



```

debug log - debug.exe emulation
clear save to file... ?
F400:0204 CF IRET
-d NCR
0710:0002 0A 00 00 00 00 00 00-00 00 00 00 00 00 B8 10 .....
0710:0012 07 8E D8 B8 00 00 A0 00-00 8A 1E 01 00 C7 06 02 .A+...a...eA@.||
0710:0022 00 00 00 E8 04 00 B4 4C-CD 21 3B C3 74 1F 83 FB ..d+.}L=;!;|tva√
0710:0032 00 74 1A 83 FB 01 74 1A-48 3B C3 74 1A 50 53 E8 .t→a√@t→H;|t→PSd
0710:0042 E8 FF 5B 58 4B 50 53 E8-E0 FF 5B 58 C3 FF 06 02 d.[XKPSdα.[X|..d
0710:0052 00 C3 01 06 02 00 C3 01-06 02 00 FF 06 02 00 C3 .|@+@.|@+@..+@.|
0710:0062 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90 éééééééééééééééé
0710:0072 90 90 90 90 F4 00 00 00-00 00 00 00 00 00 00 éééééééééééééééé
  
```

Assembly Level Program 9a – System Time

Write an Assembly Level Program to read the current time from the system and display it in the standard format on the screen.

Program

```
.model SMALL
```

```
DISPLAY MACRO
```

```
    AAM ; BCD Adjustment After Multiplication
```

```
    MOV BX, AX
```

```
    MOV DL, BH
```

```
    ADD DL, 30h
```

```
    MOV AH, 02h
```

```
    INT 21h
```

```
    MOV DL, BL
```

```
    ADD DL, 30h
```

```
    MOV AH, 02h
```

```
    INT 21h
```

```
ENDM
```

```
COLON MACRO
```

```
    MOV DL, ':'
```

```
    MOV AH, 02h
```

```
    INT 21h
```

```
ENDM
```

```
.data
```


MSG1 dB 10, 13, 'The Current System Time is \$'

.code

MOV AX, @DATA

MOV DS, AX

LEA DX, MSG1

MOV AH, 09h

INT 21h

; Interrupt to Fetch System Time

; CH - Hours | CL - Minutes | DH - Seconds | DL - Milliseconds

MOV AH, 2Ch

INT 21h

MOV AL, CH

DISPLAY

COLON

MOV AL, CL

DISPLAY

COLON

MOV AL, DH

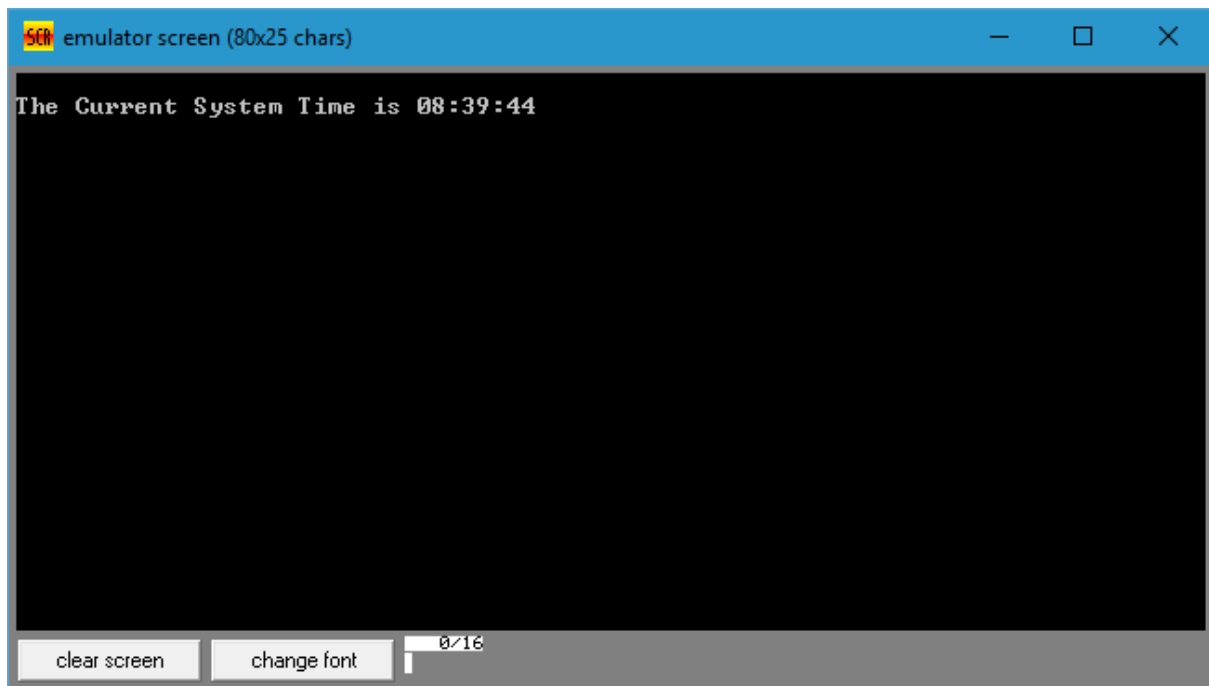
DISPLAY

MOV AH, 4Ch

INT 21h

END

Output



Assembly Level Program 10a – Decimal Up Counter

Write an Assembly Level Program to simulate a Decimal Up Counter to display 00 to 99.

Program

```
.model SMALL
```

```
.code
```

```
    MOV AL, 30h
```

```
FirstLoop:
```

```
    MOV DL, AL
```

```
    MOV AH, 02h
```

```
    INT 21h
```

```
    PUSH AX
```

```
    MOV BL, 30h
```

```
SecondLoop:
```

```
    MOV DL, BL
```

```
    MOV AH, 02h
```

```
    INT 21h
```

```
    INC BL
```

```
; Set Cursor to 2nd Column
```

```
    MOV AH, 02h
```

```
    MOV DL, 01h
```

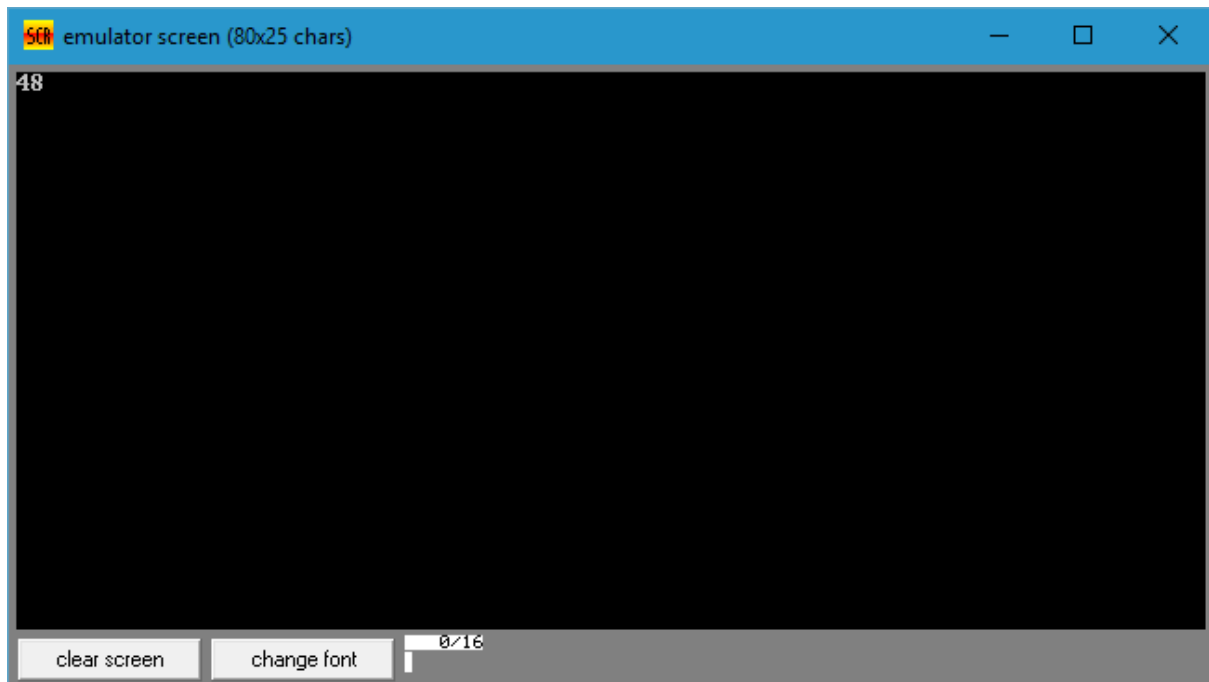
```
    INT 10h
```

```
    CMP BL, 039h
```

```
    JLE SecondLoop
```

```
; Set Cursor to 1st Column  
MOV AH, 02h  
MOV DL, 00h  
INT 10h  
  
POP AX  
INC AL  
  
CMP AL, 039h  
JLE FirstLoop  
  
MOV AH, 4Ch  
INT 21h  
END
```

Output



Assembly Level Program 11a – Cursor Movement

Write an Assembly Level Program to read a pair of input co-ordinates in BCD and move the cursor to the specified location on the screen.

Program

```
.MODEL SMALL
```

```
.DATA
```

```
    XMSG DB 13,10,'ENTER VALUE OF X CO-ORDINATES:','$'
```

```
    X DB ?
```

```
    YMSG DB 13,10,'ENTER VALUE OF Y CO-ORDINATES:','$'
```

```
    Y DB ?
```

```
.CODE
```

```
    MOV AX,@DATA
```

```
    MOV DS,AX
```

```
    MOV DX,OFFSET XMSG ;TO READ BCD CO=ORDINATES
```

```
    CALL READ_BCD
```

```
    MOV X,BH
```

```
    MOV DX,OFFSET YMSG
```

```
    CALL READ_BCD
```

```
    MOV Y,BH
```

```
    MOV AH,02H ;TO SET CURSOR POSITION
```

```
    MOV DH,X
```

```
    MOV DL,Y
```

```
    MOV BH,0
```

```
    INT 10H
```

```
    MOV DL,'-'
```

```
    MOV AH,06H
```

INT 21H

MOV AH,4CH

INT 21H

READ_BCD PROC

MOV AH,09H

INT 21H

MOV AH,01H ;FIRST DIGIT

INT 21H

MOV BH,AL

MOV AH,01H ;SECOND DIGIT

INT 21H

MOV BL,AL

MOV CL,4H

SUB BH,30H ;TO CONVERT FROM ASCII TO BCD

SUB BL,30H

SHL BH,CL

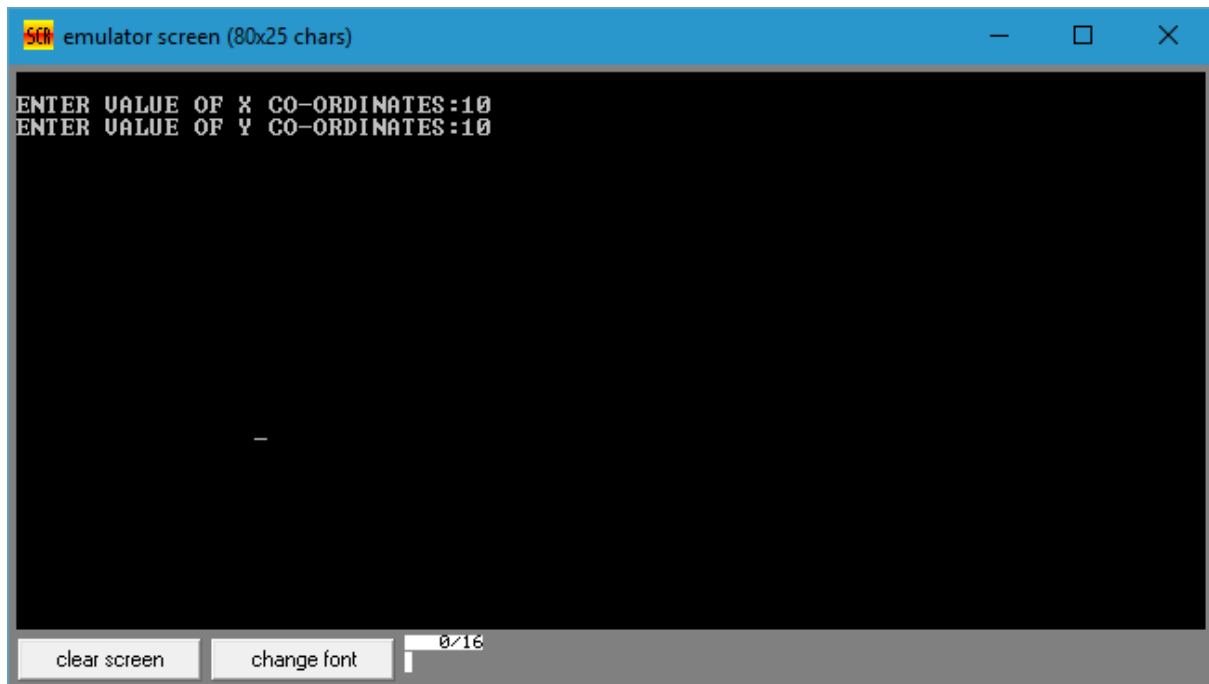
OR BH,BL

RET

READ_BCD ENDP

END

Output



Assembly Level Program 12a – File Handling

Write an Assembly Level Program to create a file (input file) and to delete an existing file.

CreateFile.asm

.model SMALL

.data

```
FNAME      dB      'SampleFile.txt', 00h
SUCCESS     dB      10, 13, 'File has been created successfully!$'
FAILURE     dB      10, 13, 'An Error Occurred during File Creation!$'
```

.code

```
MOV AX, @DATA
MOV DS, AX
```

```
MOV CX, 20h
```

```
; Interrupt to Create a File
```

```
LEA DX, FNAME
```

```
MOV AH, 3Ch
```

```
INT 21h
```

```
JC ErrorOccurred
```

```
LEA DX, SUCCESS
```

```
MOV AH, 09h
```

```
INT 21h
```

```
JMP Exit
```

ErrorOccurred:


```
LEA DX, FAILURE
```

```
MOV AH, 09h
```

```
INT 21h
```

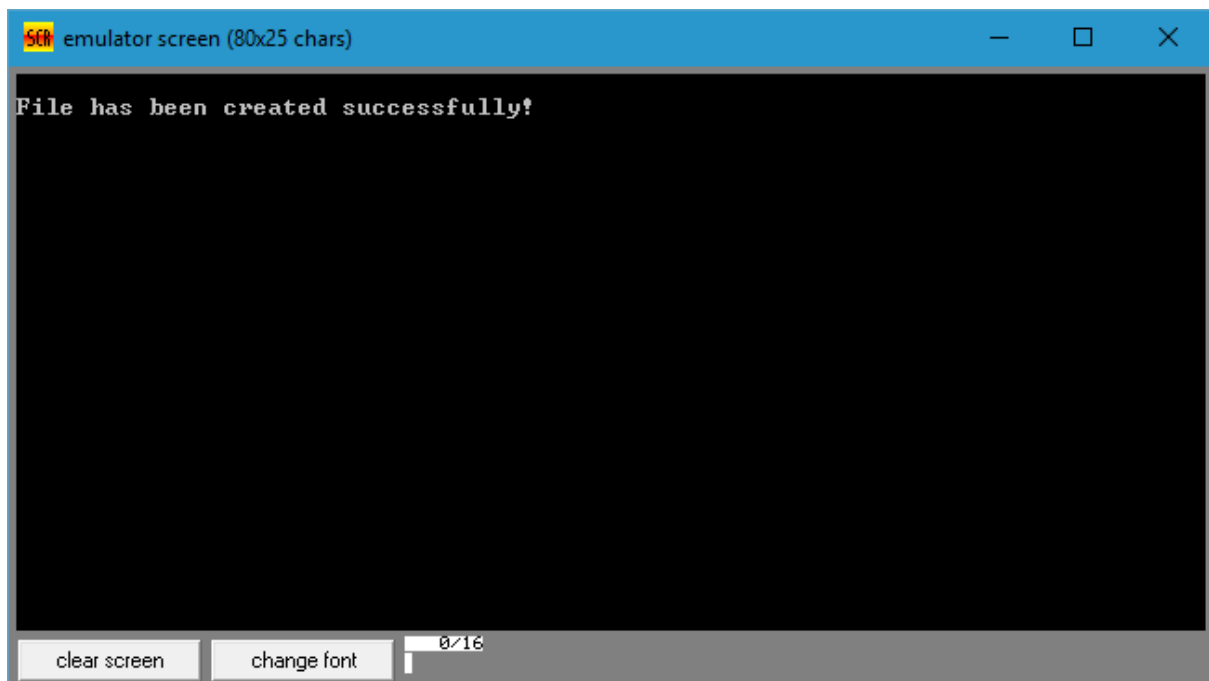
Exit:

```
MOV AH, 4Ch
```

```
INT 21h
```

```
END
```

Output



DeleteFile.am

.model SMALL

.data

FNAME	dB	'SampleFile.txt', 00h
SUCCESS	dB	10, 13, 'File has been deleted successfully!\$'
FAILURE	dB	10, 13, 'An Error Occurred during File Deletion!\$'

.code

MOV AX, @DATA
MOV DS, AX

MOV CX, 20h

; Interrupt to Delete a File

LEA DX, FNAME

MOV AH, 41h

INT 21h

JC ErrorOccurred

LEA DX, SUCCESS

MOV AH, 09h

INT 21h

JMP Exit

ErrorOccurred:

LEA DX, FAILURE

MOV AH, 09h

INT 21h

Exit:

```
MOV AH, 4Ch
```

```
INT 21h
```

```
END
```

Output

