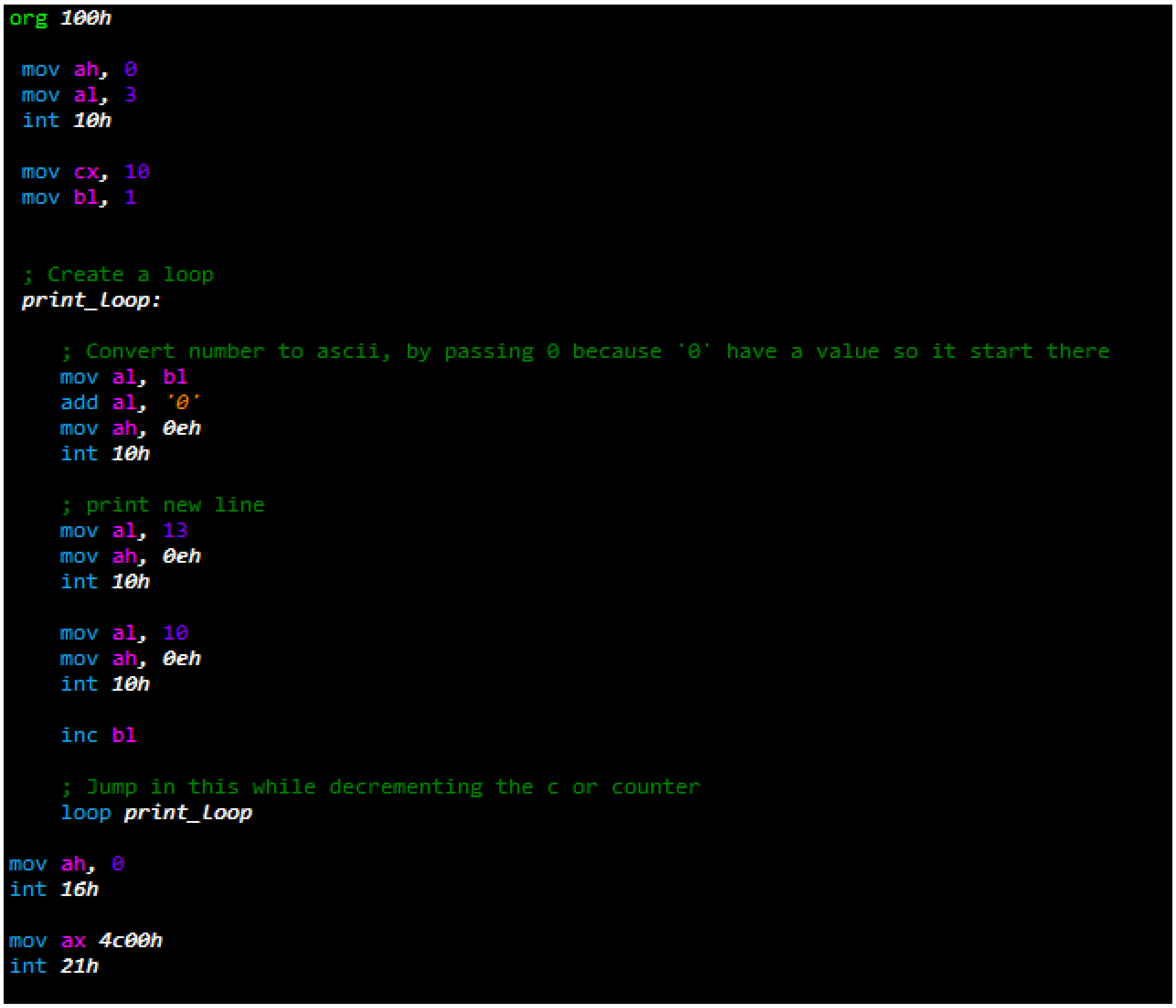
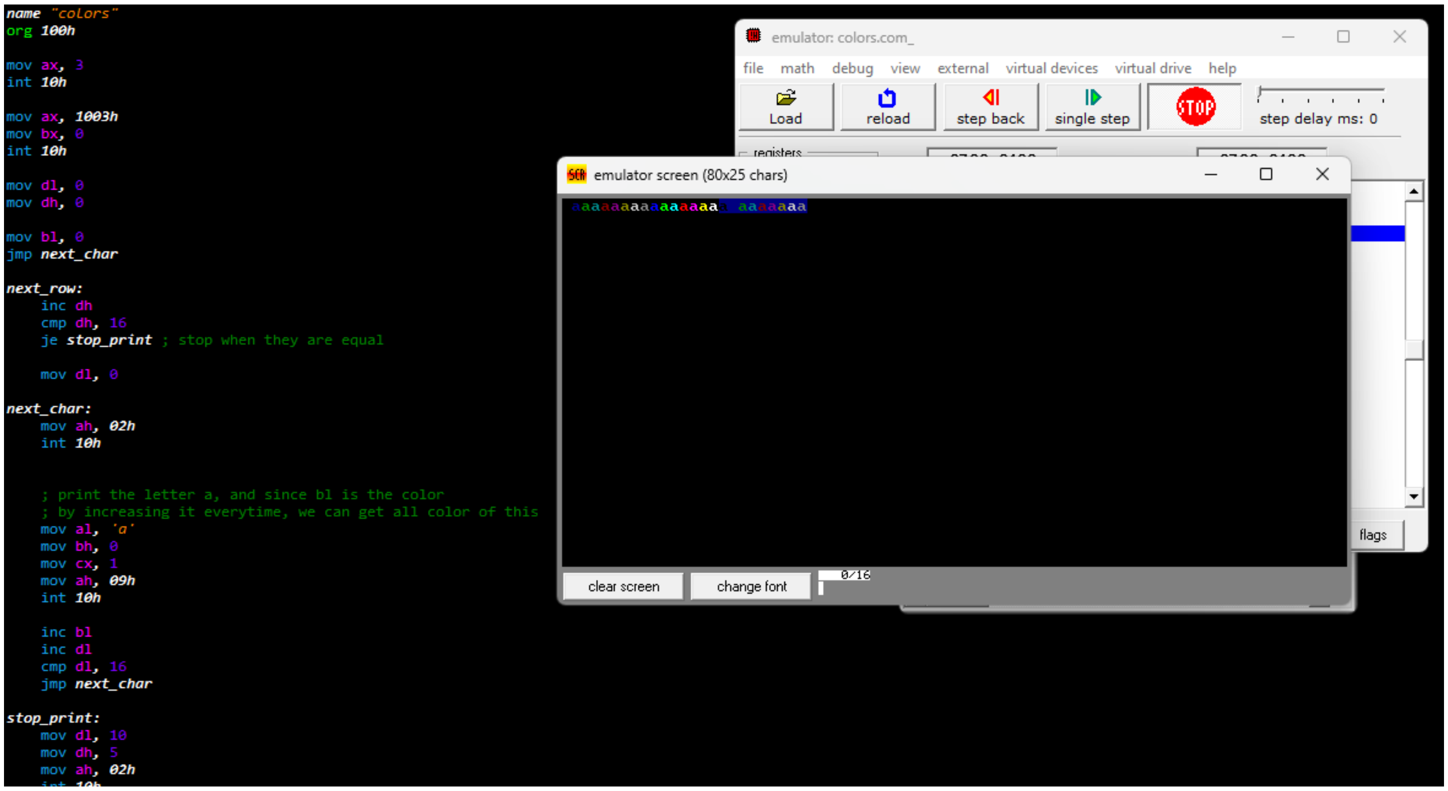
ACT 1



```
org 100h

 mov ah, 0
 mov al, 3
 int 10h

 mov cx, 10
 mov bl, 1


 ; Create a loop
 print_Loop:

    ; Convert number to ascii, by passing 0 because '0' have a value so it start there
    mov al, bl
    add al, '0'
    mov ah, 0eh
    int 10h

    ; print new line
    mov al, 13
    mov ah, 0eh
    int 10h

    mov al, 10
    mov ah, 0eh
    int 10h

    inc bl

    ; Jump in this while decrementing the c or counter
    loop print_Loop

mov ah, 0
int 16h

mov ax 4c00h
int 21h
```

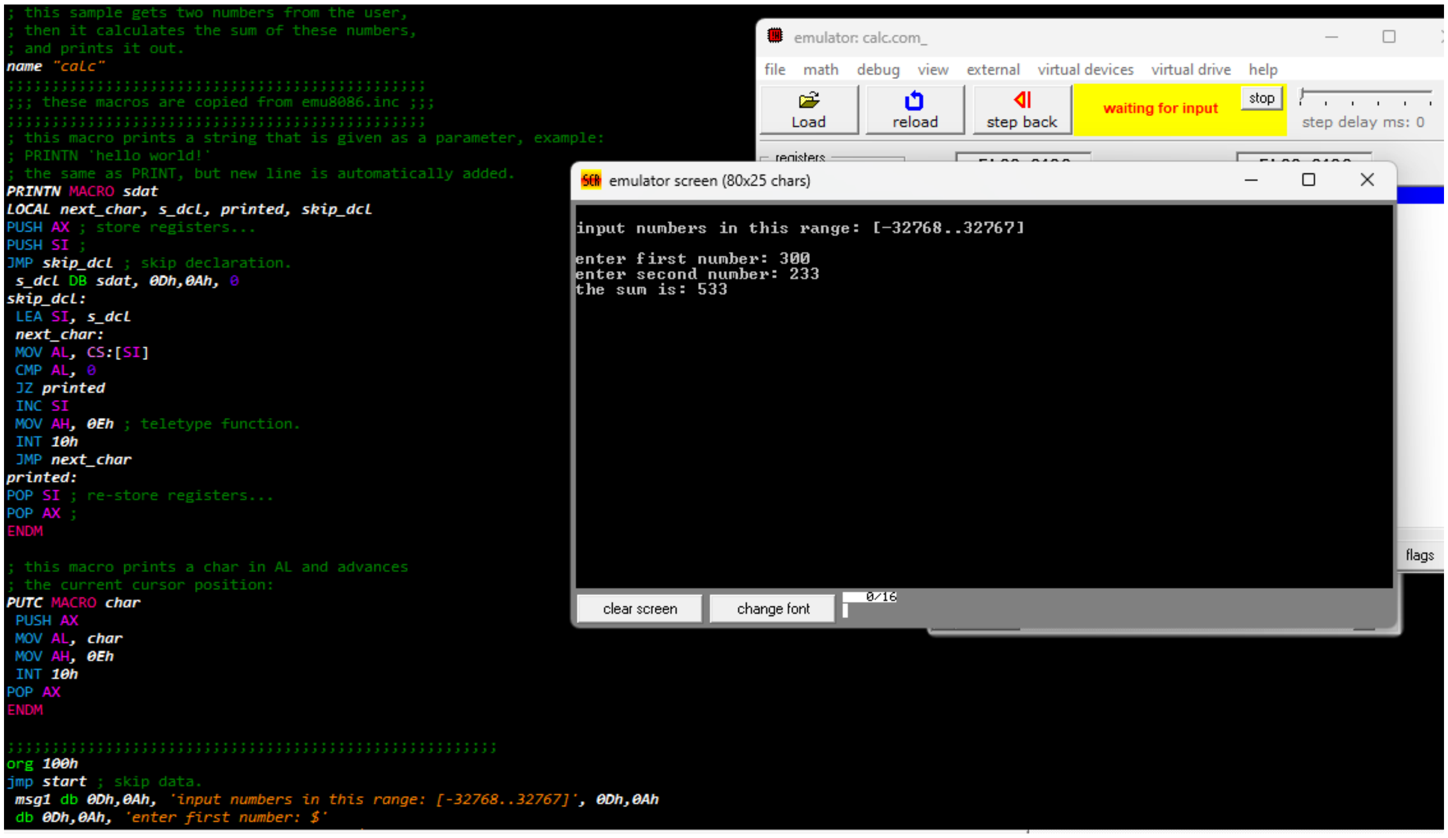
ACT 2



```
name "colors"
org 100h

mov ax, 3
int 10h

mov ax, 1003h
mov bx, 0
int 10h

mov dl, 0
mov dh, 0

mov bl, 0
jmp next_char

next_row:
    inc dh
    cmp dh, 16
    je stop_print ; stop when they are equal

    mov dl, 0

next_char:
    mov ah, 02h
    int 10h


    ; print the letter a, and since bl is the color
    ; by increasing it everytime, we can get all color of this
    mov al, 'a'
    mov bh, 0
    mov cx, 1
    mov ah, 09h
    int 10h

    inc bl
    inc dl
    cmp dl, 16
    jmp next_char

stop_print:
    mov dl, 10
    mov dh, 5
    mov ah, 02h
    int 10h
```

```
aaaaaaaaaaaaaaa aaaaaaa
```

ACT 3



```
; this sample gets two numbers from the user,
; then it calculates the sum of these numbers,
; and prints it out.
name "calc"
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; these macros are copied from emu8086.inc ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; this macro prints a string that is given as a parameter, example:
; PRINTN 'hello world!'
; the same as PRINT, but new line is automatically added.
PRINTN MACRO sdat
LOCAL next_char, s_dcl, printed, skip_dcl
PUSH AX ; store registers...
PUSH SI ;
JMP skip_dcl ; skip declaration.
 s_dcl DB sdat, 0Dh,0Ah, 0
skip_dcl:
 LEA SI, s_dcl
 next_char:
 MOV AL, CS:[SI]
 CMP AL, 0
 JZ printed
 INC SI
 MOV AH, 0Eh ; teletype function.
 INT 10h
 JMP next_char
printed:
POP SI ; re-store registers...
POP AX ;
ENDM

; this macro prints a char in AL and advances
; the current cursor position:
PUTC MACRO char
 PUSH AX
 MOV AL, char
 MOV AH, 0Eh
 INT 10h
POP AX
ENDM

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
org 100h
jmp start ; skip data.
 msg1 db 0Dh,0Ah, 'input numbers in this range: [-32768..32767]', 0Dh,0Ah
 db 0Dh,0Ah, 'enter first number: $'
```

```
input numbers in this range: [-32768..32767]

enter first number: 300
enter second number: 233
the sum is: 533
```