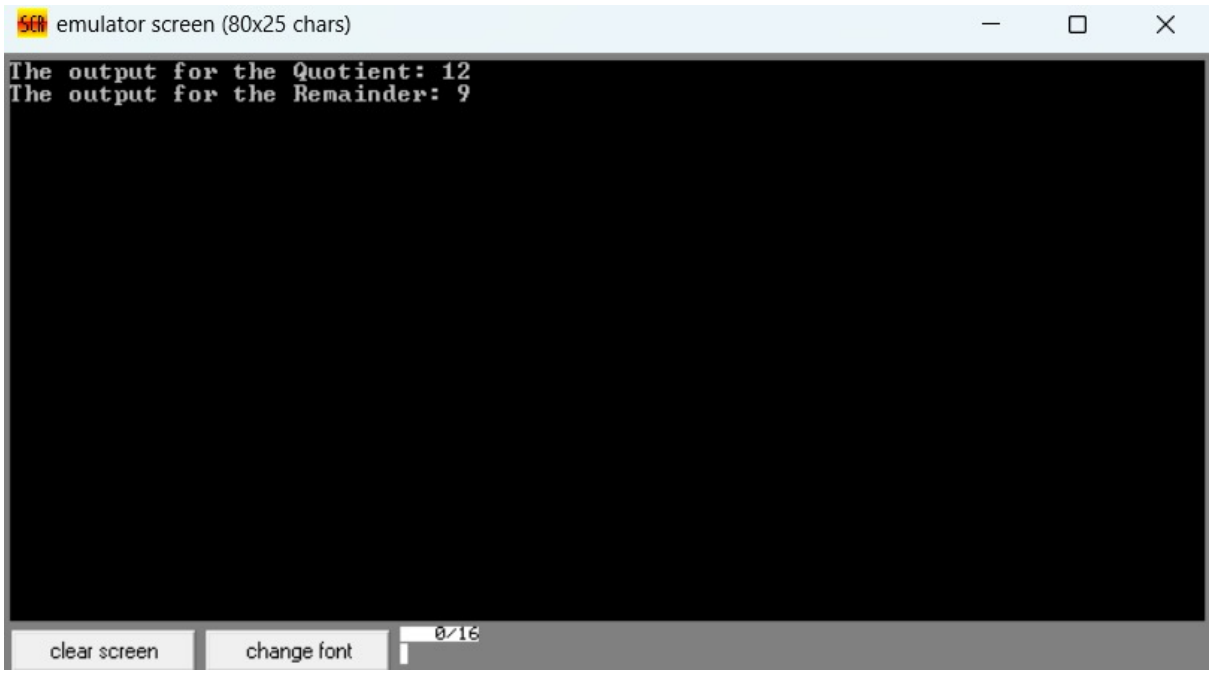
Write an assembly language program to perform division of 8-bit data.

**CODE**

```
. .model small
.stack 100h

.data
   dividend db 0A5h
   divisor db 0Ch
   quotient db ?
   remainder db ?
   msg1 db 'The output for the Quotient: $'
   msg2 db 0Dh, 0Ah, 'The output for the Remainder: $'

.code
main proc
   mov ax, @data
   mov ds, ax
   mov al, dividend
   mov bl, divisor
   xor ah, ah
   div bl
   mov quotient, al
   mov remainder, ah

   mov ah, 09h
   lea dx, msg1
   int 21h

   mov al, quotient
   call display_number

   mov ah, 09h
   lea dx, msg2
   int 21h

   mov al, remainder
   call display_number

   mov ah, 4ch
   int 21h
main endp

display_number proc
   cmp al, 10
   jb single_digit

   mov ah, 0
   mov bl, 0Ah
   div bl
```

```
    add al, 30h
    mov dl, al
    mov ah, 02h
    int 21h

    mov al, ah
    add al, 30h
    mov dl, al
    mov ah, 02h
    int 21h
    ret

single_digit:
    add al, 30h
    mov dl, al
    mov ah, 02h
    int 21h
    ret
display_number endp

end main
```



2. Write a program in assembly language to perform division of 16-bit data.

**CODE**

```
.model small
.stack 100h
.data
   dividend dw 1A3Bh
   divisor dw 0025h
   quotient dw ?
   remainder dw ?
   msg1 db 'Quotient: $'
   msg2 db 0Dh, 0Ah, 'Remainder: $'
.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, dividend
   mov bx, divisor
   xor cx, cx
   xor dx, dx

division_loop:
   cmp ax, bx
   jb division_done
   sub ax, bx
   inc cx
   jmp division_loop

division_done:
   mov quotient, cx
   mov remainder, ax

   mov ah, 09h
   lea dx, msg1
   int 21h

   mov ax, quotient
   call display_number_16

   mov ah, 09h
   lea dx, msg2
   int 21h

   mov ax, remainder
   call display_number_16

   mov ah, 4ch
   int 21h
main endp
```

```
display_number_16 proc
   push ax
   push bx
   push dx

   mov bx, 10
   xor cx, cx

convert_loop:
   xor dx, dx
   div bx
   push dx
   inc cx
   test ax, ax
   jnz convert_loop

print_digits:
   pop dx
   add dl, '0'
   mov ah, 02h
   int 21h
   loop print_digits

   pop dx
   pop bx
   pop ax
   ret
display_number_16 endp
end main
```