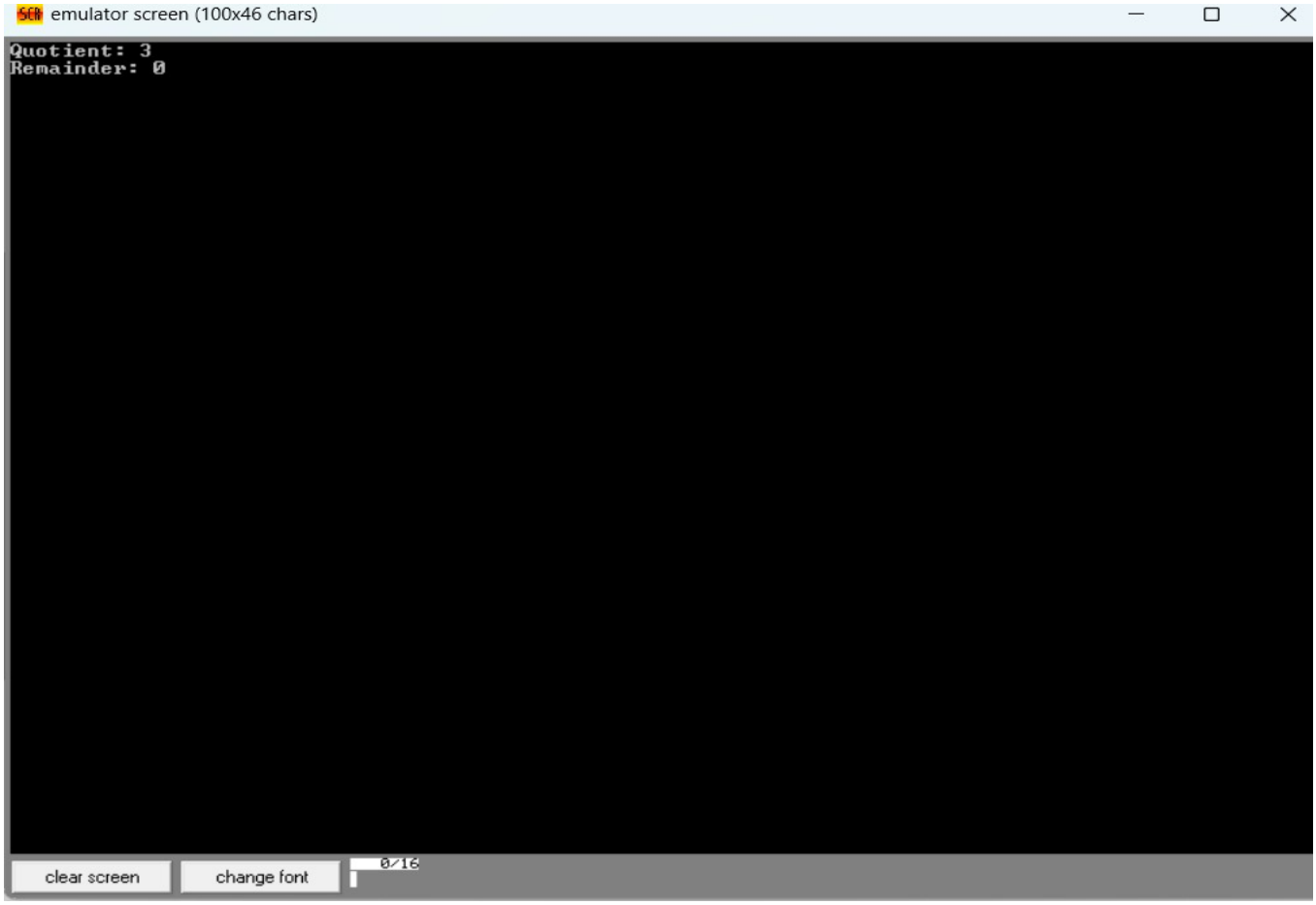
1.Write an assembly language program to perform division of 8-bit data.

Code:

```
org 100h
start:
   mov ax, cs      ; Copy CS to AX
   mov ds, ax      ; Set DS = CS

   mov al, [dividend]; Load dividend into AL
   mov bl, [divisor] ; Load divisor into BL
   xor ah, ah      ; Clear AH before division
   div bl          ; Divide AL by BL, result in AL (quotient), AH (remainder)

   mov [quotient], al ; Store quotient
   mov [remainder], ah ; Store remainder

   mov dx, offset msg_quotient ; Load offset of msg_quotient into DX
   mov ah, 09h     ; DOS function: Write string to standard output
   int 21h         ; DOS interrupt

   mov al, [quotient] ; Load quotient for printing
   add al, '0'     ; Convert quotient to ASCII
   mov dl, al      ; Prepare DL for printing
   mov ah, 02h     ; DOS function: Write character to standard output
   int 21h         ; DOS interrupt

   mov dl, 0Dh     ; Carriage return
   int 21h         ; Print carriage return
   mov dl, 0Ah     ; New line
   int 21h         ; Print new line

   mov dx, offset msg_remainder ; Load offset of msg_remainder
   mov ah, 09h     ; DOS function: Write string to standard output
   int 21h         ; DOS interrupt

   mov al, [remainder] ; Load remainder for printing
   add al, '0'     ; Convert remainder to ASCII
   mov dl, al      ; Prepare DL for printing
   mov ah, 02h     ; DOS function: Write character to standard output
   int 21h         ; DOS interrupt

   mov ah, 4Ch     ; DOS function: Terminate program
   int 21h         ; DOS interrupt

   dividend db 12   ; Define dividend
   divisor  db 4    ; Define divisor
   quotient db 0    ; Define quotient
```

```
    remainder db 0    ; Define remainder
    msg_quotient db 'Quotient: $'
    msg_remainder db 'Remainder: $'
```

Output:



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

Code:

```
org 100h
start:
    mov ax, cs      ; Copy CS to AX
    mov ds, ax      ; Set DS = CS

    xor ax, ax      ; Clear AX
    xor dx, dx      ; Clear DX
    mov ax, [dividend] ; Load dividend (16-bit) into AX
    mov bx, [divisor]  ; Load divisor (16-bit) into BX
    div bx          ; Divide DX:AX by BX, result in AX (quotient), DX (remainder)
```

```
    mov [quotient], ax ; Store quotient (16-bit)
    mov [remainder], dx ; Store remainder (16-bit)

    mov dx, offset msg_quotient ; Load offset of msg_quotient into DX
    mov ah, 09h      ; DOS function: Write string to standard output
    int 21h          ; DOS interrupt

    mov ax, [quotient] ; Load quotient for printing
    call PrintNum    ; Print number in AX

    mov dl, 0Dh      ; Carriage return
    int 21h          ; Print carriage return
    mov dl, 0Ah      ; New line
    int 21h          ; Print new line

    mov dx, offset msg_remainder ; Load offset of msg_remainder
    mov ah, 09h      ; DOS function: Write string to standard output
    int 21h          ; DOS interrupt

    mov dx, [remainder] ; Load remainder for printing
    call PrintNum    ; Print number in DX

    mov ah, 4Ch      ; DOS function: Terminate program
    int 21h          ; DOS interrupt

dividend dw 1234   ; Define dividend as 16-bit
divisor  dw 100    ; Define divisor as 16-bit
quotient dw 0      ; Define quotient as 16-bit
remainder dw 0     ; Define remainder as 16-bit
msg_quotient db 'Quotient: $'
msg_remainder db 'Remainder: $'

PrintNum:          ; Subroutine to print a number in AX
    xor cx, cx
    mov bx, 10     ; Base 10 for decimal numbers
    num_loop:
      xor dx, dx
      div bx     ; Divide AX by 10, result in AX, remainder in DX
      push dx    ; Push remainder onto stack
      inc cx     ; Increment counter
      test ax, ax ; Check if AX is zero
      jnz num_loop

    print_loop:
      pop dx     ; Pop remainder from stack
      add dl, '0' ; Convert to ASCII
      mov ah, 02h ; DOS function: Write character to standard output
```

```
    int 21h    ; DOS interrupt
    loop print_loop
  ret
```

output: