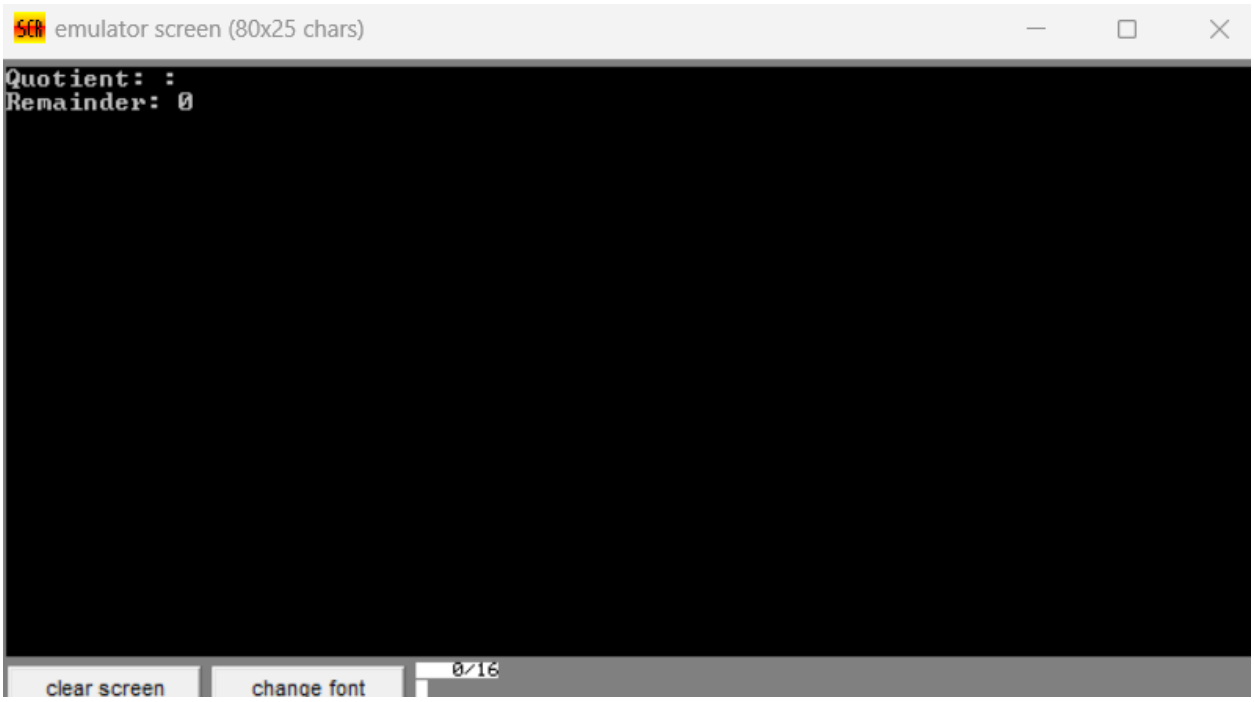
Write an assembly language program to perform division of 8-bit data.

CODE:

```
.model small
.stack 100h
.data
    dividend db 64h
    divisor db 0Ah
    quotient db ?
    remainder db ?
    msg1 db 'Quotient: $'
    msg2 db 0Dh, 0Ah, '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_value

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

    mov al, remainder
    call display_value

    mov ah, 4ch
    int 21h
main endp

display_value proc
    add al, 30h
    mov ah, 02h
    mov dl, al
    int 21h
    ret
display_value endp
end main
```



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

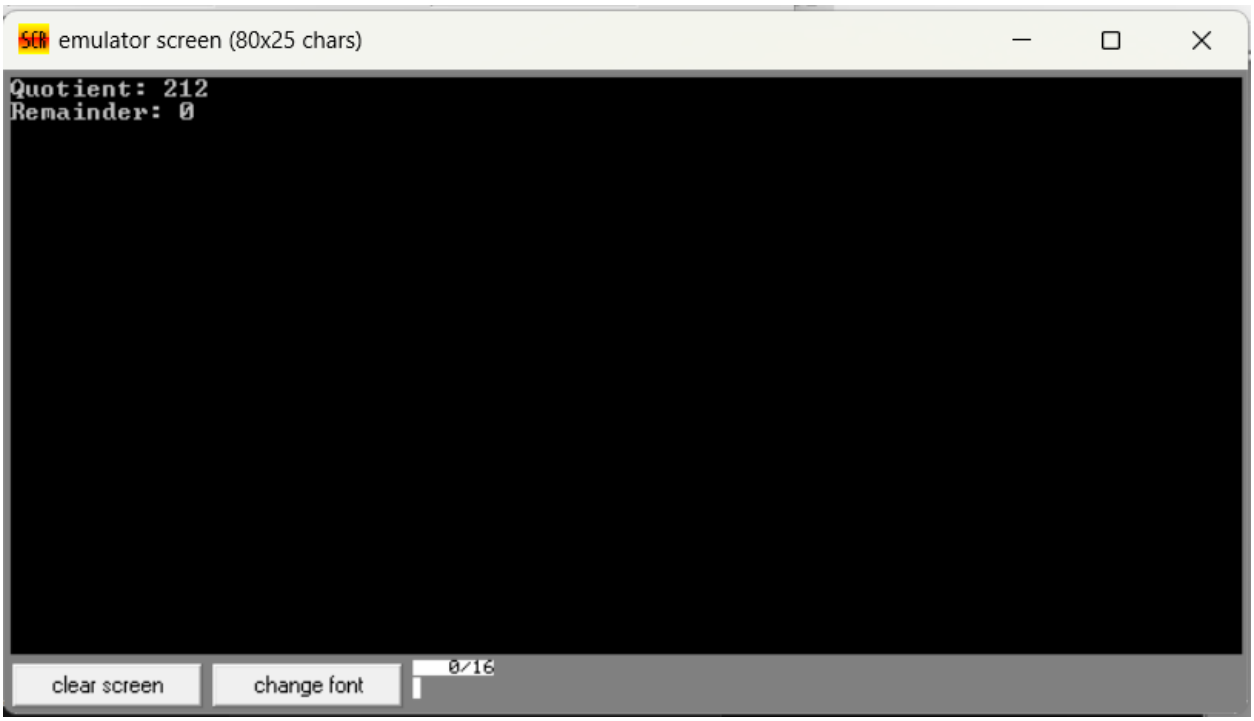
CODE:

```
.model small

.stack 100h

.data

    dividend dw 0848h
    divisor dw 000Ah
    quotient dw ?
    remainder dw ?
    msg1 db 'Quotient: $'
    msg2 db 0Dh, 0Ah, 'Remainder: $'
```

```
.code

main proc

    mov ax, @data
    mov ds, ax

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

    mov quotient, ax
    mov remainder, dx

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

    mov ax, quotient
    call display_value16

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

    mov ax, remainder
```

```
    call display_value16

    mov ah, 4ch
    int 21h

main endp

display_value16 proc

    push ax
    mov cx, 0

    cmp ax, 0
    je display_zero

convert_loop:
    xor dx, dx
    mov bx, 10
    div bx

    push dx
    inc cx
    test ax, ax
    jnz convert_loop

display_digits:
    pop dx
```

```
    add dl, 30h
    mov ah, 02h
    int 21h
    loop display_digits

    jmp done_display

display_zero:
    mov dl, '0'
    mov ah, 02h
    int 21h

done_display:
    pop ax
    ret

display_value16 endp

end main
```


emulator screen (80x25 chars)
Quotient: 212
Remainder: 0
clear screen
change font
0/16