

COAL Lab Report – Abdul Hannan Khan

Lab 2:

Task 1: ascii_characters

Code:

```
.model small
.stack 100h
.data
.code

main proc
    mov ah, 2
    mov cx, 256
    mov dl, 0

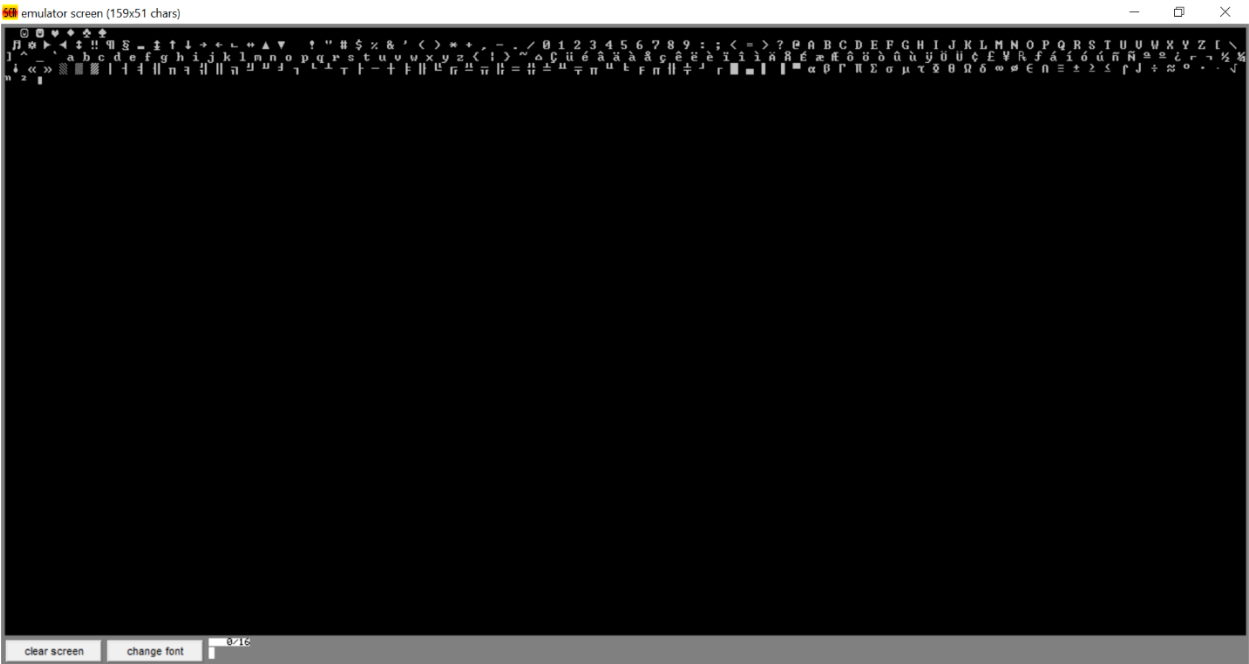
print:
    int 21h

    mov bl, dl
    mov dl, 32
    int 21h
    mov dl, bl

    inc dl
    dec cx
    jnz print

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

Output:



Task 2: arithmetic_operations

Code:

```
.model small
.stack 100h
.data
    msg1 db 'Enter first number: $'
    msg2 db 13, 10, 'Enter second number: $'
    addMsg db 13, 10, 'Addition: $'
    subMsg db 13, 10, 'Subtraction: $'
    mulMsg db 13, 10, 'Multiplication: $'
    newline db 13, 10, '$'
    num1 db ?
    num2 db ?
    result db ?
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, msg1
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov num1, al

    mov ah, 9
    lea dx, msg2
    int 21h

    mov ah, 1
```

```

    int 21h
    sub al, 30h
    mov num2, al

    mov ah, 9
    lea dx, newline
    int 21h

    call Addition
    call Subtraction
    call Multiplication

    mov ah, 4ch
    int 21h
main endp

Addition proc
    mov ah, 9
    lea dx, addMsg
    int 21h

    mov al, num1
    add al, num2
    mov result, al

    call DisplayNumber
    ret
Addition endp

Subtraction proc
    mov ah, 9
    lea dx, subMsg
    int 21h

    mov al, num1

```

```
    cmp al, num2
    jl negative_result

    sub al, num2
    mov result, al
    call DisplayNumber
    jmp subtraction_done
```

negative_result:

```
    mov dl, '-'
    mov ah, 2
    int 21h
```

```
    mov al, num2
    sub al, num1
    mov result, al
    call DisplayNumber
```

subtraction_done:

```
    ret
```

Subtraction endp

Multiplication proc

```
    mov ah, 9
    lea dx, mulMsg
    int 21h
```

```
    mov al, num1
    mul num2
    mov result, al
```

```
    call DisplayNumber
    ret
```

Multiplication endp

DisplayNumber proc

```
mov al, result
mov ah, 0
mov bl, 10
div bl
```

```
mov bh, ah
```

```
cmp al, 0
je skipFirstDigit
mov dl, al
add dl, 30h
mov ah, 2
int 21h
```

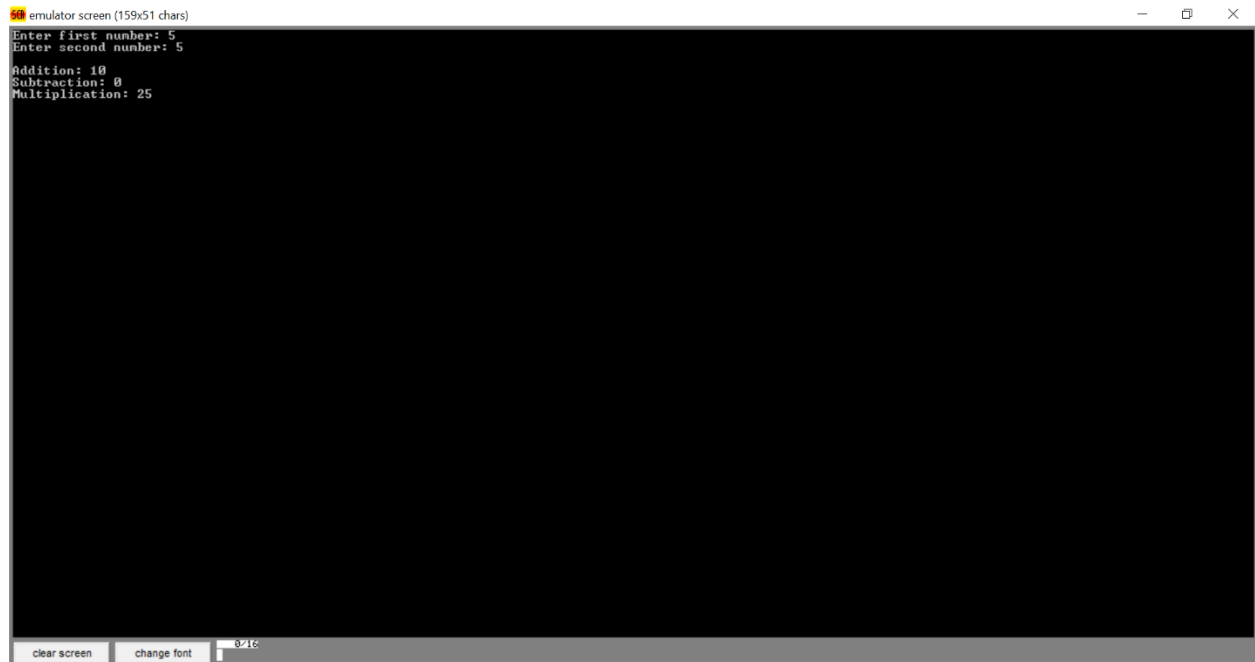
```
skipFirstDigit:
```

```
mov dl, bh
add dl, 30h
mov ah, 2
int 21h
ret
```

```
DisplayNumber endp
```

```
end main
```

Output:



The image shows a screenshot of a terminal window titled "emulator screen (159x51 chars)". The window has a black background with white text. The text displays the results of arithmetic operations: "Addition: 10", "Subtraction: 0", and "Multiplication: 25". The window includes standard OS controls (minimize, maximize, close) in the top right corner and a status bar at the bottom with buttons for "clear screen" and "change font", along with a font size indicator set to "0/16".

```
emulator screen (159x51 chars)
Enter first number: 5
Enter second number: 5

Addition: 10
Subtraction: 0
Multiplication: 25
```

clear screen change font 0/16

Task 3: multiplication_table

Code:

```
.model small
.stack 100h
.data
    msg db 'Enter a number: $'
    newline db 13, 10, '$'
    num db ?
```

```
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, msg
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov num, al

    mov ah, 9
    lea dx, newline
    int 21h

    mov cl, 1
```

```
printTable:
    mov ah, 2
    mov dl, 13
    int 21h
    mov dl, 10
```

```
int 21h

mov dl, num
add dl, 30h
int 21h

mov dl, ' '
int 21h
mov dl, 'x'
int 21h
mov dl, ' '
int 21h

mov dl, cl
add dl, 30h
int 21h

mov dl, ' '
int 21h
mov dl, '='
int 21h
mov dl, ' '
int 21h

mov al, num
mov bl, cl
mul bl

mov bl, 10
div bl

mov bh, ah

cmp al, 0
je skip_tens
```

```
    mov dl, al
    add dl, 30h
    mov ah, 2
    int 21h
```

```
skip_tens:
```

```
    mov dl, bh
    add dl, 30h
    mov ah, 2
    int 21h
```

```
    inc cl
    cmp cl, 10
    jbe printTable
```

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

```
main endp
end main
```

Output:



The screenshot shows a terminal window titled "emulator screen (159x51 chars)". The prompt "Enter a number: 5" is displayed. Below it, a multiplication table for the number 5 is shown, with the number 5 repeated on the left of each line. The table lists products from 5x1 to 5x10. At the bottom of the window, there are two buttons: "clear screen" and "change font", followed by a font size dropdown menu currently set to "0/16".

```
emulator screen (159x51 chars)
Enter a number: 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

clear screen change font 0/16

Lab 3:

Task 1: odd_numbers

Code:

```
.model small
.stack 100h
.data
    newline db 13, 10, '$'
.code
main proc
    mov ax, @data
    mov ds, ax

    mov cx, 1

print_odd:
    mov ax, cx
    mov bl, 2
    div bl

    cmp ah, 0
    je skip_print

    mov ax, cx
    mov bl, 10
    div bl

    mov bh, ah

    cmp al, 0
    je print_single
    mov dl, al
```

```
    add dl, 30h
    mov ah, 2
    int 21h
```

```
print_single:
```

```
    mov dl, bh
    add dl, 30h
    mov ah, 2
    int 21h
```

```
    mov dl, ' '
    int 21h
```

```
skip_print:
```

```
    inc cx
    cmp cx, 100
    jle print_odd
```

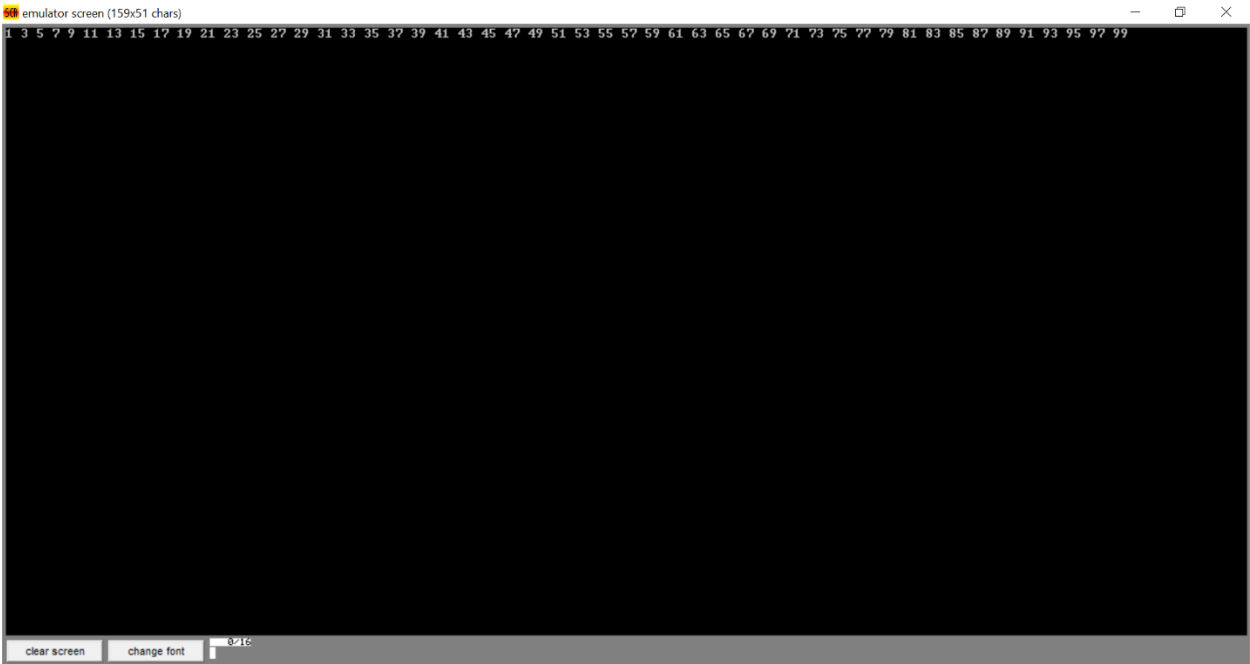
```
    mov ah, 9
    lea dx, newline
    int 21h
```

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

```
main endp
```

```
end main
```

Output:



Task 2: condition_loop

Code:

```
.model small
.stack 100h
.data
    msg1 db 'Enter x: $'
    msg2 db 13, 10, 'Enter y: $'
    newline db 13, 10, '$'
    zero db '0$'
    x db ?
    y db ?
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, msg1
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov x, al

    mov ah, 9
    lea dx, msg2
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov y, al
```



```
mov ah, 9
lea dx, newline
int 21h
```

```
mov al, x
cmp al, 4
jg else_part
```

```
mov al, y
cmp al, 9
jg else_part
```

```
mov cl, 0
```

```
loop_start:
    cmp cl, x
    jge end_loop
```

```
mov dl, x
add dl, 30h
mov ah, 2
int 21h
```

```
inc cl
jmp loop_start
```

```
else_part:
    mov ah, 9
    lea dx, zero
    int 21h
```

```
end_loop:
    mov ah, 9
    lea dx, newline
```

```
int 21h
```

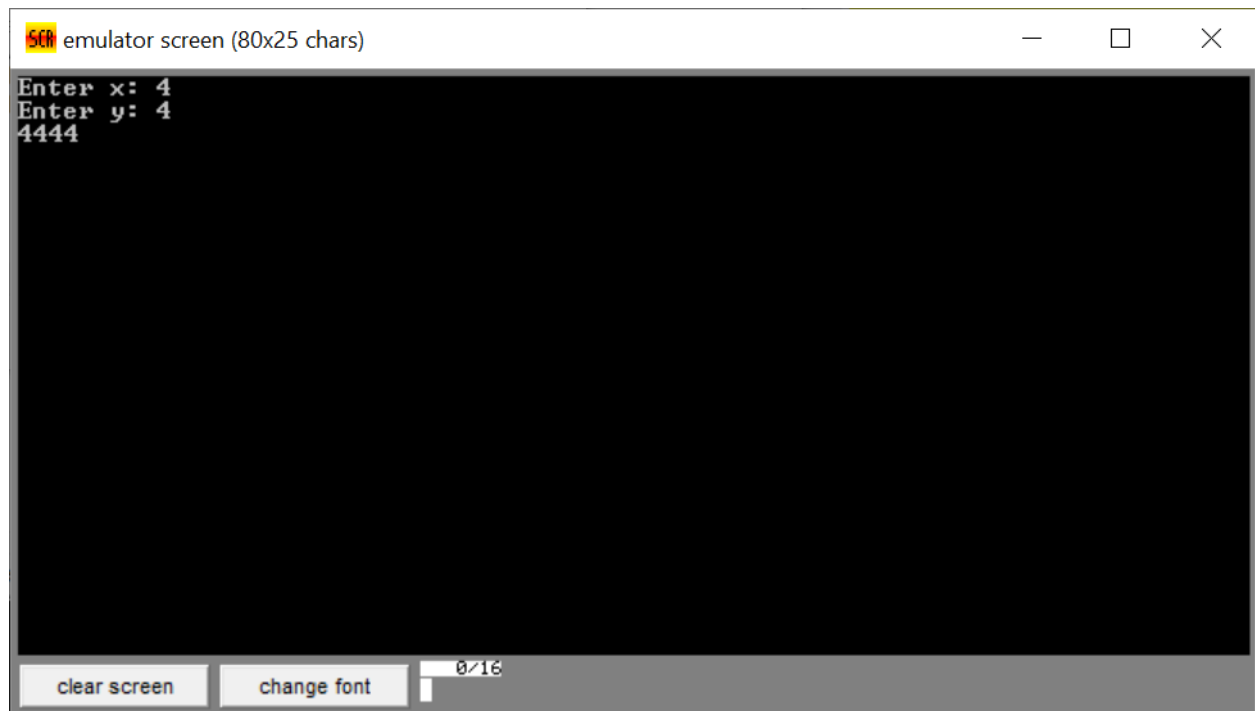
```
mov ah, 4ch
```

```
int 21h
```

```
main endp
```

```
end main
```

Output:



Task 3: largest_number

Code:

```
.model small
.stack 100h
.data
    msg1 db 'Enter x: $'
    msg2 db 13, 10, 'Enter y: $'
    newline db 13, 10, '$'
    zero db '0$'
    x db ?
    y db ?
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, msg1
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov x, al

    mov ah, 9
    lea dx, msg2
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov y, al
```

```
mov ah, 9
lea dx, newline
int 21h
```

```
mov al, x
cmp al, 4
jg else_part
```

```
mov al, y
cmp al, 9
jg else_part
```

```
mov cl, 0
```

```
loop_start:
    cmp cl, x
    jge end_loop
```

```
mov dl, x
add dl, 30h
mov ah, 2
int 21h
```

```
inc cl
jmp loop_start
```

```
else_part:
    mov ah, 9
    lea dx, zero
    int 21h
```

```
end_loop:
    mov ah, 9
    lea dx, newline
```

```
int 21h
```

```
mov ah, 4ch
```

```
int 21h
```

```
main endp
```

```
end main
```

Output:



Lab 4:

Task 1: addition

Code:

```
.model small
.stack 100h
.data
    msg1 db 'Enter first number: $'
    msg2 db 13, 10, 'Enter second number: $'
    resultMsg db 13, 10, 'Addition Result: $'
    newline db 13, 10, '$'
    num1 db ?
    num2 db ?
    result db ?
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, msg1
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov num1, al

    mov ah, 9
    lea dx, msg2
    int 21h
```



```
mov ah, 1
int 21h
sub al, 30h
mov num2, al

mov ah, 9
lea dx, resultMsg
int 21h
```

```
mov al, num1
add al, num2
mov result, al
```

```
mov al, result
mov ah, 0
mov bl, 10
div bl
```

```
mov bh, ah
```

```
cmp al, 0
je single_digit
```

```
mov dl, al
add dl, 30h
mov ah, 2
int 21h
```

single_digit:

```
mov dl, bh
add dl, 30h
mov ah, 2
int 21h
```

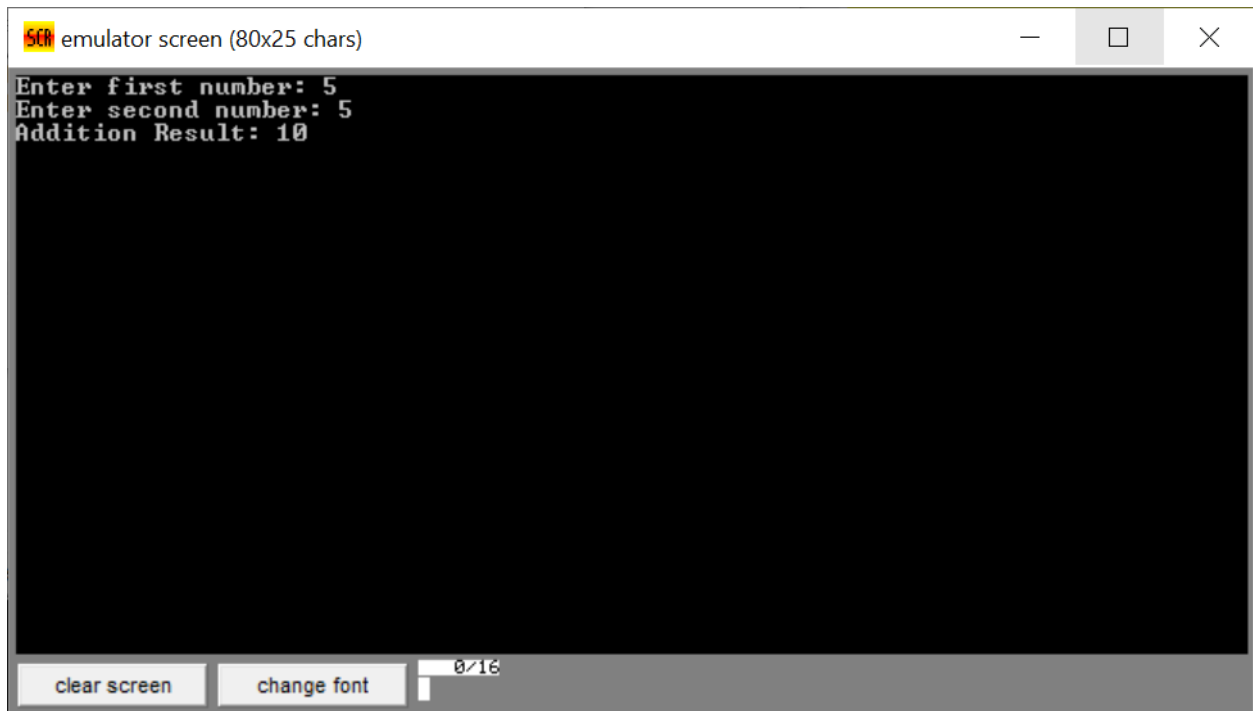
```
mov ah, 9
```

```
    lea dx, newline  
    int 21h
```

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

```
main endp  
end main
```

Output:



```
emulator screen (80x25 chars)
Enter first number: 5
Enter second number: 5
Addition Result: 10
```

clear screen change font 0/16

Task 2: multiplication

Code:

```
.model small
.stack 100h
.data
    msg1 db 'Enter first number: $'
    msg2 db 13, 10, 'Enter second number: $'
    resultMsg db 13, 10, 'Multiplication Result:
$'
    newline db 13, 10, '$'
    num1 db ?
    num2 db ?
    result db ?
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, msg1
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov num1, al

    mov ah, 9
    lea dx, msg2
    int 21h

    mov ah, 1
    int 21h
```

```
sub al, 30h
mov num2, al

mov ah, 9
lea dx, resultMsg
int 21h
```

```
mov al, num1
mul num2
mov result, al
```

```
mov al, result
mov ah, 0
mov bl, 10
div bl
```

```
mov bh, ah
```

```
cmp al, 0
je single_digit
```

```
mov dl, al
add dl, 30h
mov ah, 2
int 21h
```

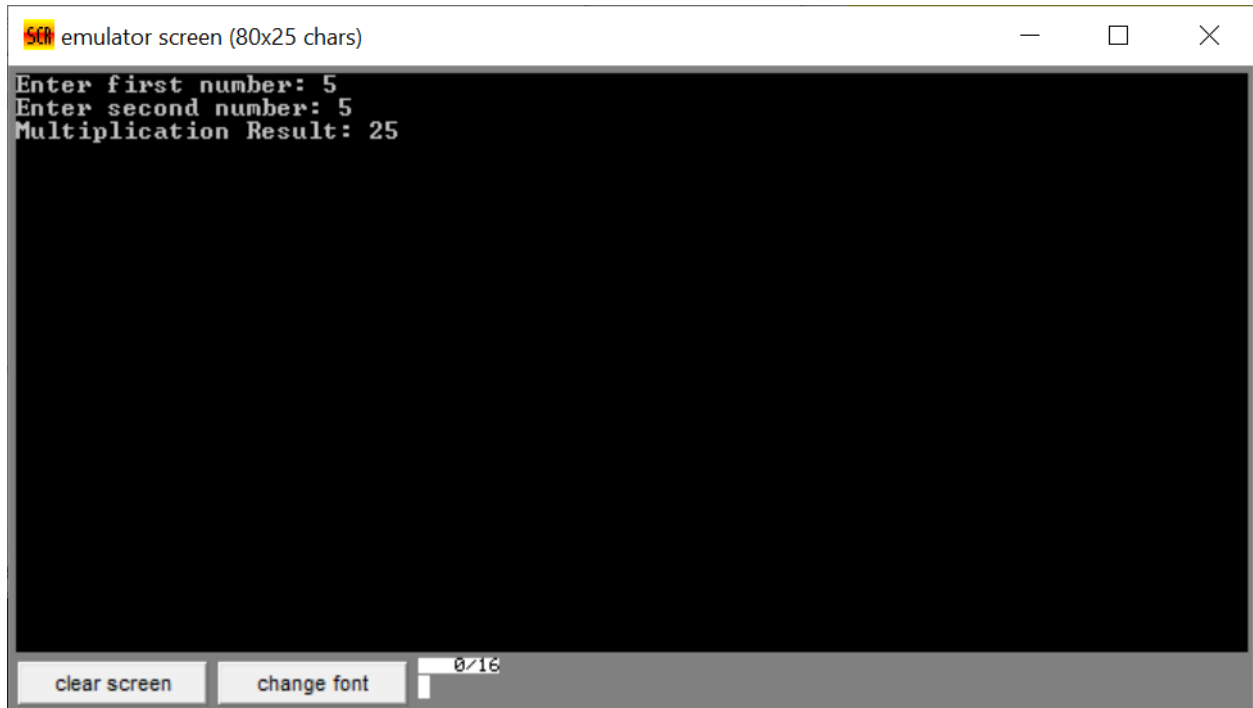
single_digit:

```
mov dl, bh
add dl, 30h
mov ah, 2
int 21h
```

```
mov ah, 9
lea dx, newline
int 21h
```

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

Output:



The image shows a window titled "emulator screen (80x25 chars)" with a standard macOS-style title bar (minimize, maximize, close buttons). The main area is a black terminal window with white text. The text displayed is: "Enter first number: 5", "Enter second number: 5", and "Multiplication Result: 25". At the bottom of the window, there is a grey bar containing two buttons: "clear screen" and "change font". To the right of these buttons is a small white box containing the text "0/16".

```
emulator screen (80x25 chars)
Enter first number: 5
Enter second number: 5
Multiplication Result: 25

clear screen  change font  0/16
```

Lab 6:

Task 1: case_converter

Code:

```
.model small
.stack 100h
.data
    msg db 'MASTER$'
.code
main proc
    mov ax, @data
    mov ds, ax
    lea si, msg
    mov cx, 6

label:
    mov al, [si]
    OR al, 00100000b
    mov dl, al
    mov ah, 2
    int 21h
    inc si
    loop label

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


Output:



Lab 7:

Task 1: extract_month_bits

Code:

```
.model small
.stack 100h
.data
    month db ?
.code
main proc
    mov ax, @data
    mov ds, ax

    mov dx, 266Ah
    mov ax, dx
    shr ax, 5
    and al, 00001111b
    mov month, al

    add month, al

    add month, 30h
    mov dl, month
    mov ah, 2
    int 21h

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

Output:



Task 2: month_extractor

Code:

```
.model smapp
.stack 100h
.data
    date dw 2A2Ah
    month db ?
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, date
    and ax, 01E0h
    mov cl, 5
    shr ax, cl
    mov month, al

    add month, 30h
    mov dl, month
    mov ah, 2
    int 21h

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

Output:

