**CHRISTIAN A NEMENO  
CS 243 ACTIVITY WEEK 11  
Mr. Roden J. Ugang  
DATE : 10/25/2024**

**1. EXER40.ASM**

; Filename: EXER40.ASM

; Programmer Name: CHRISTIAN A NEMENO

; Date: October 25, 2024

; Description: This assembly language program will prompt two integers and display their sum

.model small

.stack 100h

.data

    prompt1 db 'Enter first integer : $'

    prompt2 db 'Enter second integer : $'

    resultMsg db 'The sum of the two integers is: $'

    nL db 10, 13, '$'

    int1 dw ?

    int2 dw ?

    result dw ?

    buffer          db 6,?,6 dup('$')

.code

newLine proc

    mov ah, 09h

    lea dx, nL

    int 21h

    ret

newLine endp

printString proc

    mov ah, 09h

    int 21h

    ret

printString endp

setInput proc

    ; Read string input

    mov ah, 0Ah                   ; Set AH to 0Ah for buffered input

    lea dx, buffer                ; Load address of buffer into DX

    int 21h                       ; Call DOS interrupt to read input

    ; Convert string to number

    xor ax, ax                    ; Clear AX to prepare for conversion

    xor cx, cx                    ; Clear CX to use as a counter

    mov cl, [buffer+1]            ; Get length of input from buffer

    lea si, buffer+2              ; Point SI to the start of the string

    mov bx, 10                    ; Set BX to 10 for base 10 conversion

convert\_loop:                     ; Begin conversion loop

    mul bx                        ; AX = AX \* 10 (shift left by one decimal)

    mov dl, [si]                  ; Get next character from string

    sub dl, '0'                   ; Convert ASCII character to numerical value

    add ax, dx                    ; Add converted value to AX

    inc si                        ; Move to the next character

    loop convert\_loop             ; Repeat until all characters are processed

    ret                           ; Return from procedure

setInput endp

print proc                 ; Procedure to print a number

    ; Convert number to string

    mov cx, 0                     ; Initialize digit counter to 0

    mov bx, 10                    ; Set BX to 10 for division

divide\_loop:                      ; Begin divide loop

    xor dx, dx                    ; Clear DX before division

    div bx                        ; Divide AX by 10 (result in AX, remainder in DX)

    push dx                       ; Push remainder (digit) onto stack

    inc cx                        ; Increment digit counter

    test ax, ax                   ; Check if quotient (AX) is zero

    jnz divide\_loop               ; If not zero, continue dividing

print\_loop:                       ; Begin print loop

    pop dx                        ; Pop digit from stack

    add dl, '0'                   ; Convert digit to ASCII

    mov ah, 2                     ; Set AH to 2 for character output

    int 21h                       ; Call DOS interrupt to print character

    loop print\_loop               ; Repeat until all digits are printed

    ret                           ; Return from procedure

print endp                 ; End of print\_number procedure

main proc

    mov ax, @data

    mov ds, ax

    ; Input first integer

    lea dx, prompt1

    call printString

    call setInput

    mov int1, ax        ; Store first input

    call newLine

    ; Input second integer

    lea dx, prompt2

    call printString

    call setInput

    mov int2, ax         ; Store second input

    call newLine

    mov ax, int1

    add ax, int2

    mov result, ax

        ; Display result message

        lea dx, resultMsg

        call printString

        mov ax, result

        call print

        ; Exit program

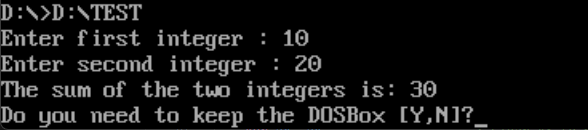
        mov ax, 4C00h

        int 21h

    main endp

    end main

**Screenshot**

****

**2. EXER41.ASM**

; Filename: EXER41.ASM

; Programmer Name: CHRISTIAN A NEMENO

; Date: OCTOBER 25, 2029

; Description: This assembly language program will prompt two integers and display their difference

.model small

.stack 100h

.data

    prompt1 db 'Enter first integer: $'

    prompt2 db 'Enter second integer: $'

    resultMsg db 'The difference of the two integers is: $'

    nL db 10, 13, '$'

    int1 dw ?

    int2 dw ?

    result dw ?

    buffer          db 6,?,6 dup('$')

.code

newLine proc

    mov ah, 09h

    lea dx, nL

    int 21h

    ret

newLine endp

printString proc

    mov ah, 09h

    int 21h

    ret

printString endp

setInput proc

      ; Read string input

    mov ah, 0Ah                   ; Set AH to 0Ah for buffered input

    lea dx, buffer                ; Load address of buffer into DX

    int 21h                       ; Call DOS interrupt to read input

    ; Convert string to number

    xor ax, ax                    ; Clear AX to prepare for conversion

    xor cx, cx                    ; Clear CX to use as a counter

    mov cl, [buffer+1]            ; Get length of input from buffer

    lea si, buffer+2              ; Point SI to the start of the string

    mov bx, 10                    ; Set BX to 10 for base 10 conversion

convert\_loop:                     ; Begin conversion loop

    mul bx                        ; AX = AX \* 10 (shift left by one decimal)

    mov dl, [si]                  ; Get next character from string

    sub dl, '0'                   ; Convert ASCII character to numerical value

    add ax, dx                    ; Add converted value to AX

    inc si                        ; Move to the next character

    loop convert\_loop             ; Repeat until all characters are processed

    ret                           ; Return from procedure

setInput endp

print proc                 ; Procedure to print a number

    ; Convert number to string

    mov cx, 0                     ; Initialize digit counter to 0

    mov bx, 10                    ; Set BX to 10 for division

divide\_loop:                      ; Begin divide loop

    xor dx, dx                    ; Clear DX before division

    div bx                        ; Divide AX by 10 (result in AX, remainder in DX)

    push dx                       ; Push remainder (digit) onto stack

    inc cx                        ; Increment digit counter

    test ax, ax                   ; Check if quotient (AX) is zero

    jnz divide\_loop               ; If not zero, continue dividing

print\_loop:                       ; Begin print loop

    pop dx                        ; Pop digit from stack

    add dl, '0'                   ; Convert digit to ASCII

    mov ah, 2                     ; Set AH to 2 for character output

    int 21h                       ; Call DOS interrupt to print character

    loop print\_loop               ; Repeat until all digits are printed

    ret                           ; Return from procedure

print endp                 ; End of print\_number procedure

main proc

    mov ax, @data

    mov ds, ax

    ; Input first integer

    lea dx, prompt1

    call printString

    call setInput

    mov int1, ax        ; Store first input

    call newLine

    ; Input second integer

    lea dx, prompt2

    call printString

    call setInput

    mov int2, ax         ; Store second input

    call newLine

    mov ax, int1

    sub ax, int2

    mov result, ax

        ; Display result message

        lea dx, resultMsg

        call printString

        mov ax, result

        call print

        ; Exit program

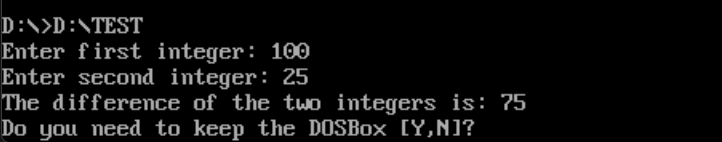
        mov ax, 4C00h

        int 21h

    main endp

    end main

**Screenshot:**

****

**3. EXER42.ASM**

; Filename: EXER42.ASM

; Programmer Name: CHRISTIAN A NEMENO

; Date: October 25, 2024

; Description: This assembly language program will prompt two integers and display their product

.model small

.stack 100h

.data

    prompt1 db 'Enter first integer: $'

    prompt2 db 'Enter second integer: $'

    resultMsg db 'The product of the two integers is: $'

    nL db 10, 13, '$'

    int1 dw ?

    int2 dw ?

    result dw ?

    buffer          db 6,?,6 dup('$')

.code

newLine proc

    mov ah, 09h

    lea dx, nL

    int 21h

    ret

newLine endp

printString proc

    mov ah, 09h

    int 21h

    ret

printString endp

setInput proc

      ; Read string input

    mov ah, 0Ah                   ; Set AH to 0Ah for buffered input

    lea dx, buffer                ; Load address of buffer into DX

    int 21h                       ; Call DOS interrupt to read input

    ; Convert string to number

    xor ax, ax                    ; Clear AX to prepare for conversion

    xor cx, cx                    ; Clear CX to use as a counter

    mov cl, [buffer+1]            ; Get length of input from buffer

    lea si, buffer+2              ; Point SI to the start of the string

    mov bx, 10                    ; Set BX to 10 for base 10 conversion

convert\_loop:                     ; Begin conversion loop

    mul bx                        ; AX = AX \* 10 (shift left by one decimal)

    mov dl, [si]                  ; Get next character from string

    sub dl, '0'                   ; Convert ASCII character to numerical value

    add ax, dx                    ; Add converted value to AX

    inc si                        ; Move to the next character

    loop convert\_loop             ; Repeat until all characters are processed

    ret                           ; Return from procedure

setInput endp

print proc                 ; Procedure to print a number

    ; Convert number to string

    mov cx, 0                     ; Initialize digit counter to 0

    mov bx, 10                    ; Set BX to 10 for division

divide\_loop:                      ; Begin divide loop

    xor dx, dx                    ; Clear DX before division

    div bx                        ; Divide AX by 10 (result in AX, remainder in DX)

    push dx                       ; Push remainder (digit) onto stack

    inc cx                        ; Increment digit counter

    test ax, ax                   ; Check if quotient (AX) is zero

    jnz divide\_loop               ; If not zero, continue dividing

print\_loop:                       ; Begin print loop

    pop dx                        ; Pop digit from stack

    add dl, '0'                   ; Convert digit to ASCII

    mov ah, 2                     ; Set AH to 2 for character output

    int 21h                       ; Call DOS interrupt to print character

    loop print\_loop               ; Repeat until all digits are printed

    ret                           ; Return from procedure

print endp                 ; End of print\_number procedure

main proc

    mov ax, @data

    mov ds, ax

    ; Input first integer

    lea dx, prompt1

    call printString

    call setInput

    mov int1, ax        ; Store first input

    call newLine

    ; Input second integer

    lea dx, prompt2

    call printString

    call setInput

    mov int2, ax         ; Store second input

    call newLine

    mov ax, int1

    mov bx, int2

    mul bx                  ; AX = int1 \* int2

        mov result, ax         ; Store the product in result

        ; Display result message

        lea dx, resultMsg

        call printString

        mov ax, result

        call print

        ; Exit program

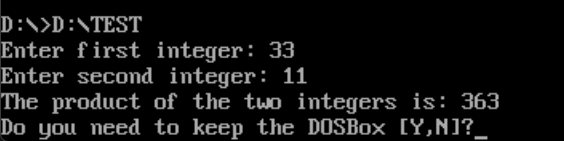
        mov ax, 4C00h

        int 21h

    main endp

    end main

**Screenshot:**

****

**4. EXER43.ASM**

; Filename: EXER43.ASM

; Programmer Name: CHRISTIAN A NEMENO

; Date: October 25, 2024

; Description: This assembly language program will prompt two integers and display their quotient

.model small

.stack 100h

.data

    prompt1 db 'Enter first integer: $'

    prompt2 db 'Enter second integer: $'

    resultMsg db 'The quotient of the two integers is: $'

    inv db 'Error! Division by zero $'

    nL db 10, 13, '$'

    int1 dw ?

    int2 dw ?

    result dw ?

    buffer          db 6,?,6 dup('$')

.code

newLine proc

    mov ah, 09h

    lea dx, nL

    int 21h

    ret

newLine endp

printString proc

    mov ah, 09h

    int 21h

    ret

printString endp

setInput proc

      ; Read string input

    mov ah, 0Ah                   ; Set AH to 0Ah for buffered input

    lea dx, buffer                ; Load address of buffer into DX

    int 21h                       ; Call DOS interrupt to read input

    ; Convert string to number

    xor ax, ax                    ; Clear AX to prepare for conversion

    xor cx, cx                    ; Clear CX to use as a counter

    mov cl, [buffer+1]            ; Get length of input from buffer

    lea si, buffer+2              ; Point SI to the start of the string

    mov bx, 10                    ; Set BX to 10 for base 10 conversion

convert\_loop:                     ; Begin conversion loop

    mul bx                        ; AX = AX \* 10 (shift left by one decimal)

    mov dl, [si]                  ; Get next character from string

    sub dl, '0'                   ; Convert ASCII character to numerical value

    add ax, dx                    ; Add converted value to AX

    inc si                        ; Move to the next character

    loop convert\_loop             ; Repeat until all characters are processed

    ret                           ; Return from procedure

setInput endp

print proc                 ; Procedure to print a number

    ; Convert number to string

    mov cx, 0                     ; Initialize digit counter to 0

    mov bx, 10                    ; Set BX to 10 for division

divide\_loop:                      ; Begin divide loop

    xor dx, dx                    ; Clear DX before division

    div bx                        ; Divide AX by 10 (result in AX, remainder in DX)

    push dx                       ; Push remainder (digit) onto stack

    inc cx                        ; Increment digit counter

    test ax, ax                   ; Check if quotient (AX) is zero

    jnz divide\_loop               ; If not zero, continue dividing

print\_loop:                       ; Begin print loop

    pop dx                        ; Pop digit from stack

    add dl, '0'                   ; Convert digit to ASCII

    mov ah, 2                     ; Set AH to 2 for character output

    int 21h                       ; Call DOS interrupt to print character

    loop print\_loop               ; Repeat until all digits are printed

    ret                           ; Return from procedure

print endp                 ; End of print\_number procedure

main proc

    mov ax, @data

    mov ds, ax

    ; Input first integer

    lea dx, prompt1

    call printString

    call setInput

    mov int1, ax        ; Store first input

    call newLine

    ; Input second integer

    lea dx, prompt2

    call printString

    call setInput

    mov int2, ax         ; Store second input

    call newLine

    mov ax, int1

    xor dx, dx              ; Clear DX for division

        mov bx, int2            ; Set divisor

        cmp bx, 0               ; Check if divisor is zero

        je divisionByZero       ; Jump if divisor is zero

        div bx                  ; AX = int1 / int2, quotient in AX, remainder in DX

        mov result, ax          ; Store quotient in result

        jmp Output

    divisionByZero:

        lea dx, inv

        call printString

        jmp Done

    Output:

        ; Display result message

        lea dx, resultMsg

        call printString

        mov ax, result

        call print

    Done:

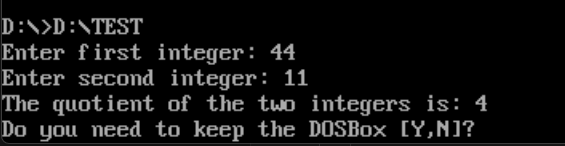
        ; Exit program

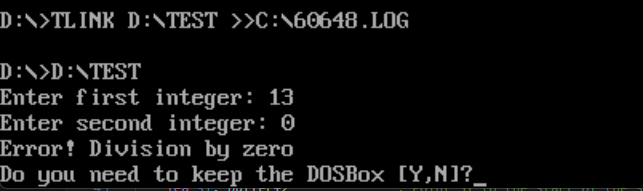
        mov ax, 4C00h

        int 21h

    main endp

    end main

**Screenshot:  
**

****

**5. EXER44.ASM**

; Filename: EXER44.ASM

; Programmer Name: CHRISTIAN A NEMENO

; Date: October 25, 2024

; Description: Create and call a function, sub-program, or procedure for the main menu and for each math operation.

.model small

.stack 500h

.data

    intro       db 'THE CALCULATOR',13,10

                DB 'Created by: CHRISTIAN A NEMENO',13,10

                db 'Date: October 25, 2024', 13,10,10

                db 'Main Menu',13,10,'$'

    a           db 'a - Addition',13,10,'$'

    s           db 's - Subtraction',13,10,'$'

    m           db 'm - Multiplication',13,10,'$'

    d           db 'd - Division',13,10,'$'

    e           db 'e - Exit',13,10,'$'

    choice      db 'Enter your choice: $'

    aPrompt     db 'Addition$'

    aPrompt1    db 13,10,'Enter first addend: $'

    aPrompt2    db 13,10,'Enter second addend: $'

    aOp         db ' + $'

    sPrompt     db 'Subtraction$'

    sPrompt1    db 13,10,'Enter minuend: $'

    sPrompt2    db 13,10,'Enter subtrahend: $'

    sOp         db ' - $'

    mPrompt     db 'Multiplication$'

    mPrompt1    db 13,10,'Enter multiplicand: $'

    mPrompt2    db 13,10,'Enter multiplier: $'

    mOp         db ' x $'

    dPrompt     db 'Division$'

    dPrompt1    db 13,10,'Enter dividend: $'

    dPrompt2    db 13,10,'Enter divisor: $'

    dOp         db ' / $'

    equals      db ' = $'

    eDisplay    db 'Exit Program$'

    invalid     db 'INVALID CHOICE!$'

    ending      db 13,10,'Thank you.$'

    negSign     db '-$'

    divZero     db 13,10,'Error: Division by zero is not allowed.', 13, 10,'$'

    input1   dw ?

    input2   dw ?

    sum      dw ?

    diff     dw ?

    prod     dw ?

    quo      dw ?

    buffer   db 6,?,6 dup('$')

.code

print proc

    mov ah, 09h

    int 21h

    ret

print endp

getNum PROC

    ; Read string input

    mov ah, 0Ah                   ; Set AH to 0Ah for buffered input

    lea dx, buffer                ; Load address of buffer into DX

    int 21h                       ; Call DOS interrupt to read input

    ; Convert string to number

    xor ax, ax                    ; Clear AX to prepare for conversion

    xor cx, cx                    ; Clear CX to use as a counter

    mov cl, [buffer+1]            ; Get length of input from buffer

    lea si, buffer+2              ; Point SI to the start of the string

    mov bx, 10                    ; Set BX to 10 for base 10 conversion

    convert\_loop:                     ; Begin conversion loop

        mul bx                        ; AX = AX \* 10 (shift left by one decimal)

        mov dl, [si]                  ; Get next character from string

        sub dl, '0'                   ; Convert ASCII character to numerical value

        add ax, dx                    ; Add converted value to AX

        inc si                        ; Move to the next character

        loop convert\_loop             ; Repeat until all characters are processed

        ret

getNum ENDP

converter proc

    mov cx, 0            ; Counter for digits

    mov bx, 10           ; Base for decimal conversion

    converter\_loop:

        xor dx, dx           ; Clear DX before dividing

        div bx               ; AX / BX, result in AX, remainder in DX

        push dx              ; Push remainder onto stack

        inc cx               ; Count the number of digits

        test ax, ax         ; Check if quotient is zero

        jne converter\_loop       ; Repeat if not

    display\_loop:

        pop dx               ; Pop the last remainder

        add dl, '0'          ; Convert to ASCII

        mov ah, 02h          ; Print character function

        int 21h              ; Interrupt to print character

        dec cx               ; Decrease the digit counter

        cmp cx, 0            ; Check if finished printing all digits

        jne display\_loop      ; Continue if not

    ret

converter endp

getChar PROC

    mov ah,01h

    int 21h

    ret

getChar ENDP

newLine PROC

    mov ah, 02h

    mov dl, 13

    int 21h

    mov ah, 02h

    mov dl, 10

    int 21h

    ret

newLine ENDP

addition proc

    call newLine

    call newLine

    mov ah,09h

    mov bl,20h ;green bg and black text

    mov cx,8

    int 10h

    lea dx, aPrompt

    call print

    lea dx, aPrompt1

    call print

    call getNum

    mov input1, ax

    lea dx, aPrompt2

    call print

    call getNum

    mov input2, ax

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    mov dx, input1

    add dx, input2

    mov sum, dx

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    call newLine

    mov ax, input1

    call converter

    lea dx, aOp

    call print

    mov ax, input2

    call converter

    lea dx, equals

    call print

    mov ax, sum

    call converter

    call newLine

    ret

addition endp

subtraction proc

    call newLine

    call newLine

    mov ah,09h

    mov bl,30h ;cyan bg and black text

    mov cx,11

    int 10h

    lea dx, sPrompt

    call print

    lea dx, sPrompt1

    call print

    call getNum

    mov input1, ax

    lea dx, sPrompt2

    call print

    call getNum

    mov input2, ax

    call newLine

    mov ax, input1

    call converter

    lea dx, sOp

    call print

    mov ax, input2

    call converter

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    mov ax, input1

    sub ax, input2

    mov diff, ax

    cmp ax, 0

    lea dx, equals

    call print

    jge display\_result

    xor ax,ax

    mov ax, input2

    sub ax, input1

    mov diff, ax

    lea dx, negSign

    call print

    display\_result:

        mov ax, diff

        call converter

    call newLine

    ret

subtraction endp

multiplication proc

    call newLine

    call newLine

    mov ah,09h

    mov bl,50h ;purple bg and black text

    mov cx,14

    int 10h

    lea dx, mPrompt

    call print

    lea dx, mPrompt1

    call print

    call getNum

    mov input1, ax

    lea dx, mPrompt2

    call print

    call getNum

    mov input2, ax

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    mov ax, input1

    mov dx, input2

    mul dx

    mov prod, ax

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    call newLine

    mov ax, input1

    call converter

    lea dx, mOp

    call print

    mov ax, input2

    call converter

    lea dx, equals

    call print

    mov ax, prod

    call converter

    call newLine

    ret

multiplication endp

checkAndHandleZero PROC

    cmp bx, 0

    jne continueDivision

    lea dx, divZero

    call print

    call endingDisplay

    jmp startLoop

    continueDivision:

        ret

checkAndHandleZero ENDP

division proc

    call newLine

    call newLine

    mov ah,09h

    mov bl,60h ;orange bg and black text

    mov cx,8

    int 10h

    lea dx, dPrompt

    call print

    lea dx, dPrompt1

    call print

    call getNum

    mov input1, ax

    lea dx, dPrompt2

    call print

    call getNum

    mov input2, ax

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    mov ax, input1

    mov bx, input2

    call checkAndHandleZero

    xor dx, dx

    div bx

    mov quo, ax

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    call newLine

    mov ax, input1

    call converter

    lea dx, dOp

    call print

    mov ax, input2

    call converter

    lea dx, equals

    call print

    mov ax, quo

    call converter

    call newLine

    ret

division endp

exitProg proc

    call newLine

    call newLine

    mov ah,09h

    mov bl,10h ;blue bg and black text

    mov cx,12

    int 10h

    lea dx, eDisplay

    call print

    call newLine

    call terminate

exitProg endp

invalidChoice proc

    call newLine

    call newLine

    mov ah,09h

    mov bl,0CEh ;red bg and blinking yellow text

    mov cx,15

    int 10h

    lea dx, invalid

    mov ah, 09h

    int 21h

    call newLine

    ret

invalidChoice endp

endingDisplay proc

    lea dx, ending

    call print

    call getChar

    ret

endingDisplay endp

terminate proc

    lea dx, ending

    call print

    call getChar

    mov ax, 4C00h

    int 21h

terminate endp

start:

    mov  ax, @data

    mov  ds, ax

    startLoop:

        mov ax, 3

        int 10h

        lea dx, intro

        call print

        mov ah,09h

        mov bl,20h ;green bg and black text

        mov cx,12

        int 10h

        lea dx, a

        call print

        mov ah,09h

        mov bl,30h ;cyan bg and black text

        mov cx,15

        int 10h

        lea dx, s

        call print

        mov ah,09h

        mov bl,50h ;purple bg and black text

        mov cx,18

        int 10h

        lea dx, m

        call print

        mov ah,09h

        mov bl,60h ;orange bg and black text

        mov cx,12

        int 10h

        lea dx, d

        call print

        mov ah,09h

        mov bl,10h ;blue bg and black text

        mov cx,8

        int 10h

        lea dx, e

        call print

        call newLine

        lea dx, choice

        call print

        mov ah, 01h

        int 21h

        cmp al,'a'

        je doAdd

        cmp al,'s'

        je doSub

        cmp al,'m'

        je doMult

        cmp al,'d'

        je doDiv

        cmp al,'e'

        je doExit

        jne doInvalid

        doAdd:

            call addition

            call endingDisplay

            jmp startLoop

        doSub:

            call subtraction

            call endingDisplay

            jmp startLoop

        doMult:

            call multiplication

            call endingDisplay

            jmp startLoop

        doDiv:

            call division

            call endingDisplay

            jmp startLoop

        doExit:

            call exitProg

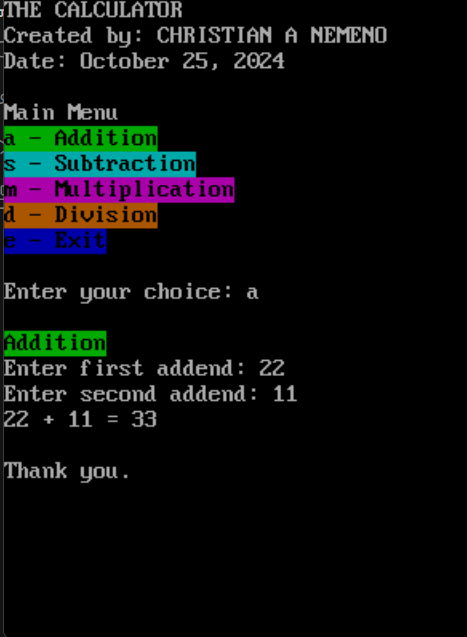
        doInvalid:

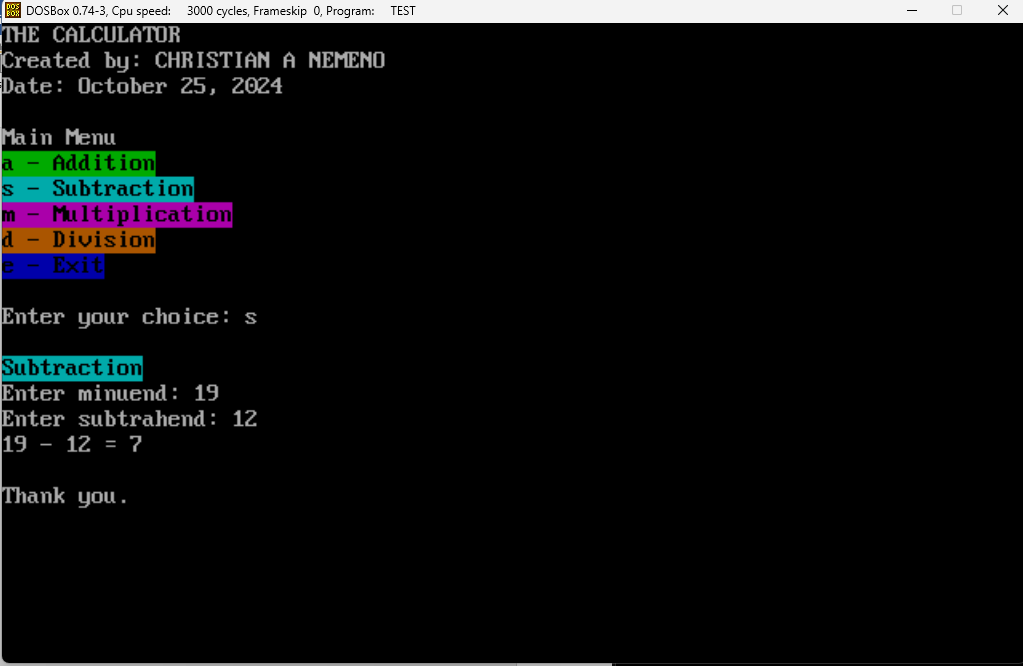
            call invalidChoice

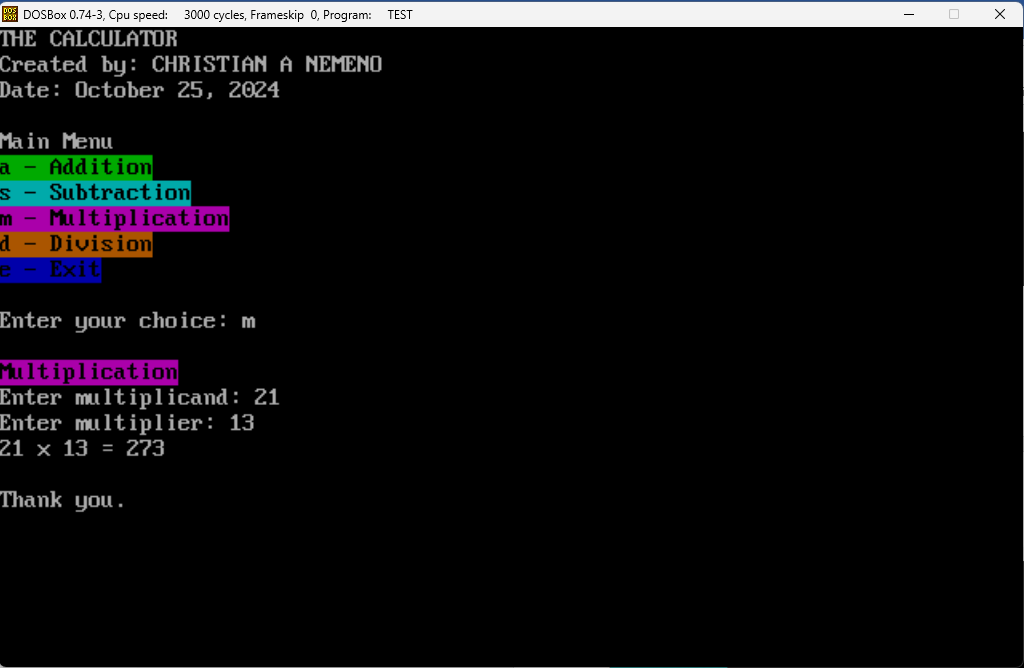
            call endingDisplay

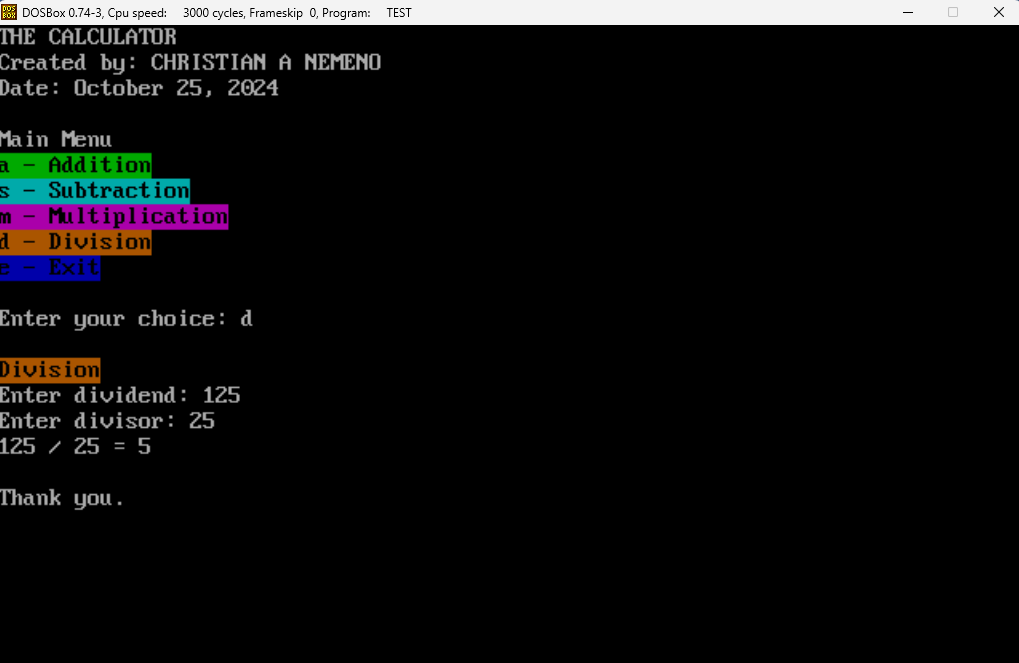
            jmp startLoop

end start

**Screenshot:  
**

****

****

****

****

