CHRISTIAN A. NEMENO
OCTOBER 18, 2024
COURSE: Computer Organization and Architecture
Intstructor: Mr. Roden J. Ugang
CS243 Week 10 Lab Exercises

1.
```
; Filename: EXER33.ASM

; Programmer Name: Christian A. Nemeno
; Date: OCTOBER 18, 2024
; Description: This assembly language program will input
; two single-digit numbers, add the two numbers,
; and display the sum of the two numbers.
.MODEL SMALL
.STACK 100H
.DATA
num1 DB ?
num2 DB ?
sum DB ?
msg1 DB 'Enter first number (0-9): $'
msg2 DB 13, 10, 'Enter second number (0-9): $'
msg3 DB 13, 10, 'The sum is: $'
.CODE
MAIN PROC
; Initialize data segment
MOV AX, @DATA
MOV DS, AX
; Input first number
LEA DX, msg1
MOV AH, 09H
INT 21H
; Read character input
MOV AH, 01H
INT 21H
SUB AL, '0' ; Convert ASCII to number
MOV num1, AL
; Input second number
LEA DX, msg2
MOV AH, 09H
INT 21H
MOV AH, 01H
INT 21H
SUB AL, '0' ; Convert ASCII to number
MOV num2, AL
```

```
; Calculate sum
MOV AL, num1
ADD AL, num2
MOV sum, AL
; Display result
LEA DX, msg3
MOV AH, 09H
INT 21H
; Convert sum to ASCII
ADD sum, '0'
MOV DL, sum
MOV AH, 02H
INT 21H
; Exit program
MOV AX, 4C00H
INT 21H
MAIN ENDP
END MAIN
```

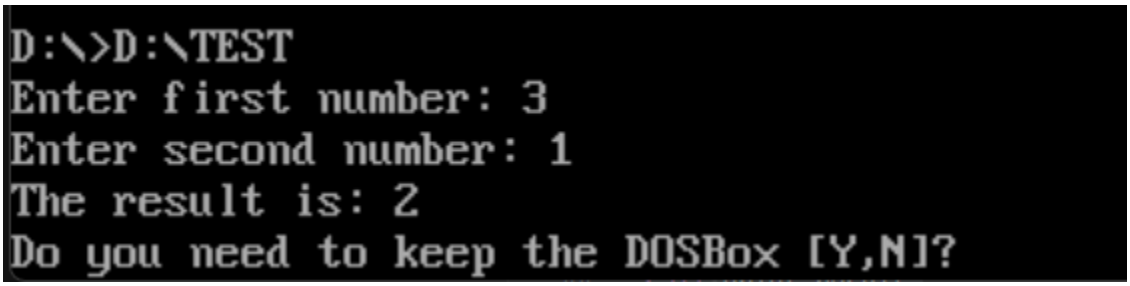
Screen shot run:

```
D:\>TLINK D:\test
Turbo Link  Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>D:\test
Enter first number (0-9): 4
Enter second number (0-9): 2
The sum is: 6
D:\>_
```

```
2. ; Filename: EXER34.ASM

; Programmer Name: CHRISTIAN A. NEMENO
; Date: OCTOBER 18, 2024
; Description: This assembly language program will input two
;single-digit numbers, subtract the two numbers,
; and display the difference of the two numbers.
.model small
.stack 100h
.data
msg1 db 'Enter first number: $'
msg2 db 13,10,'Enter second number: $'
resultMsg db 13,10,'The result is: $'
num1 db ?
num2 db ?
result db ?
.code
start:
; Set up the data segment
mov ax, @data
mov ds, ax
; Prompt for the first number
mov ah, 09h
lea dx, msg1
int 21h
; Read first number
call read_number
mov num1, al
; Prompt for the second number
mov ah, 09h
lea dx, msg2
int 21h
; Read second number
call read_number
mov num2, al
; Subtract the second number from the first
mov al, num1
sub al, num2
mov result, al
; Display the result
mov ah, 09h
lea dx, resultMsg
int 21h
; Convert result to ASCII and print
call print_result
```

```
; Exit program
mov ax, 4C00h
int 21h
; Read a number from keyboard (assumes single digit input)
read_number proc
mov ah, 01h ; Function to read a character
int 21h
sub al, '0' ; Convert ASCII to integer
ret
read_number endp
; Print the result (single digit)
print_result proc
add result, '0' ; Convert result to ASCII
mov ah, 0Eh ; BIOS teletype output function
mov al, result
int 10h
ret
print_result endp
end start
```

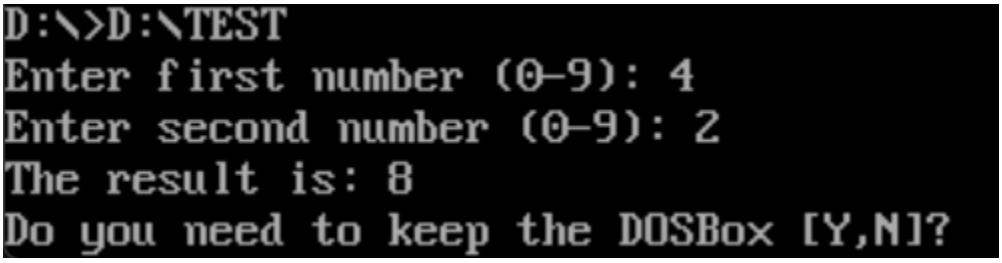
Screenshot run:



3.
```
; Filename: EXER35.ASM

; Programmer Name: CHRISTIAN A. NEMENO
; Date: OCTOBER 18, 2024
; Description: This assembly language program will input two
;single-digit numbers, multiply the two numbers,
; and display the product of the two numbers.
.model small
.stack 100h
.data
msg1 db 'Enter first number (0-9): $'
msg2 db 13,10,'Enter second number (0-9): $'
resultMsg db 13,10,'The result is: $'
num1 db ?
num2 db ?
result db ?
.code
start:
; Set up the data segment
mov ax, @data
mov ds, ax
; Prompt for the first number
mov ah, 09h
lea dx, msg1
int 21h
; Read first number
call read_number
mov num1, al
; Prompt for the second number
mov ah, 09h
lea dx, msg2
int 21h
; Read second number
call read_number
mov num2, al
; Multiply the two numbers
mov al, num1
mov bl, num2
mul bl ; AL = AL * BL, result in AX
mov result, al ; Store the lower byte of the result
; Display the result
mov ah, 09h
lea dx, resultMsg
int 21h
; Convert result to ASCII and print
```

```
call print_result
; Exit program
mov ax, 4C00h
int 21h
; Read a number from keyboard (assumes single digit input)
read_number proc
mov ah, 01h ; Function to read a character
int 21h
sub al, '0' ; Convert ASCII to integer
ret
read_number endp
; Print the result (single digit)
print_result proc
add result, '0' ; Convert result to ASCII
mov ah, 0Eh ; BIOS teletype output function
mov al, result
int 10h
ret
print_result endp
end start
```

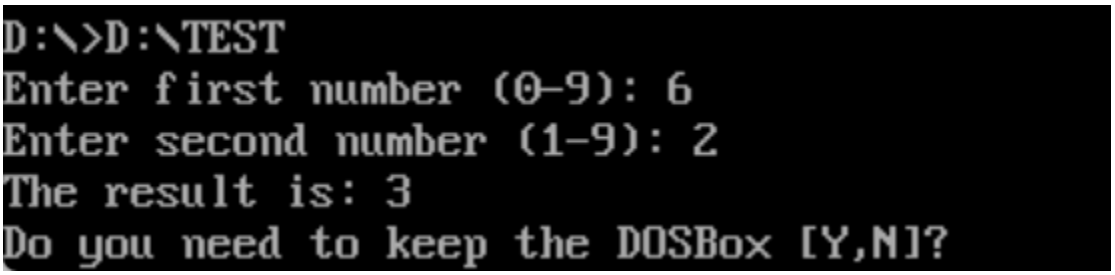
Screenshot run:



```
4. ; Filename: EXER36.ASM

; Programmer Name: CHRISTIAN A. NEMENO
; Date: OCTOBER 18, 2024
; Description: This assembly language program will input two
;single-digit numbers, divide the two numbers,
; and display the quotient of the two numbers.
.model small
.stack 100h
.data
msg1 db 'Enter first number (0-9): $'
msg2 db 13,10,'Enter second number (1-9): $' ; Second number
;cannot be zero
resultMsg db 13,10,'The result is: $'
num1 db ?
num2 db ?
result db ?
.code
start:
; Set up the data segment
mov ax, @data
mov ds, ax
; Prompt for the first number
mov ah, 09h
lea dx, msg1
int 21h
; Read first number
call read_number
mov num1, al
; Prompt for the second number
mov ah, 09h
lea dx, msg2
int 21h
; Read second number
call read_number
mov num2, al
; Check for division by zero
cmp num2, 0
je div_by_zero
; Divide the two numbers
mov al, num1
xor ah, ah ; Clear AH for the division
mov bl, num2
div bl ; AL = AL / BL, quotient in AL, remainder in AH
mov result, al ; Store the quotient
```

```
; Display the result
mov ah, 09h
lea dx, resultMsg
int 21h
; Convert result to ASCII and print
call print_result
; Exit program
mov ax, 4C00h
int 21h
div_by_zero:
; Handle division by zero (optional: you can display a
;message)
mov ah, 09h
lea dx, msg2 ; Reuse msg2 for simplicity
int 21h
; Exit program
mov ax, 4C00h
int 21h
; Read a number from keyboard (assumes single digit input)
read_number proc
mov ah, 01h ; Function to read a character
int 21h
sub al, '0' ; Convert ASCII to integer
ret
read_number endp
; Print the result (single digit)
print_result proc
add result, '0' ; Convert result to ASCII
mov ah, 0Eh ; BIOS teletype output function
mov al, result
int 10h
ret
print_result endp
end start
```

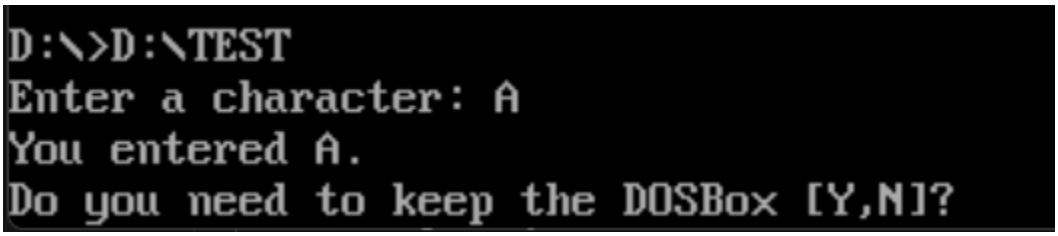
Screenshot run:



5.
```
; Filename: EXER37.ASM

; Programmer Name: CHRISTIAN A. NEMENO
; Date: October 18, 2024
; Description: Create a program that inputs a character. If the character
; is the capital letter A, display message “You entered A.”,
; else display “You entered not A.”


.model small
.stack 100h
.data
    prompt      db      'Enter a character: $'
    msgA        db      13, 10,'You entered A.$'
    msgNotA     db      13, 10,'You entered not A.$'
    inputChar   db ?

.code
main:
    ; Set up the data segment
    mov ax, @data
    mov ds, ax

    ; Display prompt
    mov dx, offset prompt
    mov ah, 09h
    int 21h

    ; Read a character from the keyboard
    mov ah, 01h
    int 21h
    mov inputChar, al  ; Store the character in inputChar

    ; Compare the character with 'A'
    cmp inputChar, 'A'
    je isA             ; If equal, jump to isA

notA:
    ; Display "You entered not A."
    mov dx, offset msgNotA
    mov ah, 09h
    int 21h
    jmp endProgram

isA:
    ; Display "You entered A."
```

```
    mov dx, offset msgA
    mov ah, 09h
    int 21h

endProgram:
    ; Exit the program
    mov ax, 4C00h
    int 21h
end main
```

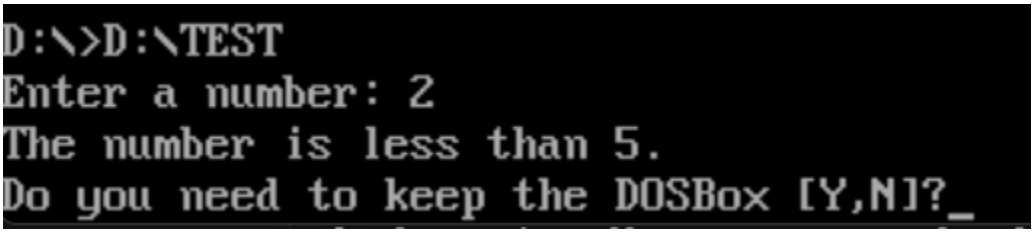
Screenshot run:

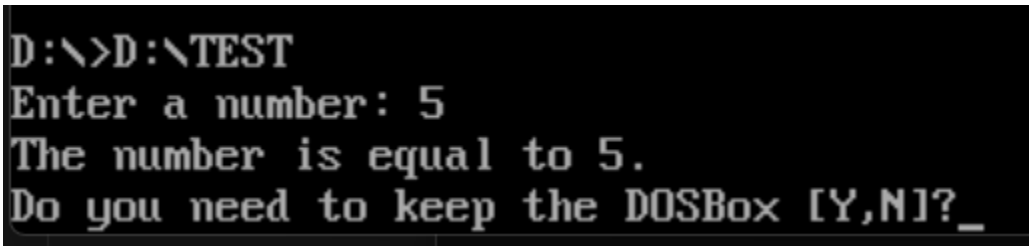


6.
```
; Filename: EXER38.ASM

; Programmer Name: CHRISTIAN A. NEMENO
; Date: October 18, 2024
; Description: Create a program that inputs a number. Display the following
messages depending on the value of the number entered

.model small
.stack 100h
.data
    prompt      db 'Enter a number: $'
    msgEqual    db 13, 10,'The number is equal to 5.$'
    msgLess     db 13, 10,'The number is less than 5.$'
    msgGreater  db 13, 10,'The number is greater than 5.$'
    num    db ?

.code
main:
    ; Set up the data segment
    mov ax, @data
    mov ds, ax

    ; Display prompt
    mov dx, offset prompt
    mov ah, 09h
    int 21h

    ; Read a number from the keyboard
    mov ah, 01h
    int 21h           ; Read a character
    sub al, '0'       ; Convert ASCII to integer
    mov num, al  ; Store the number

    ; Compare the number with 5
    cmp num, 5
    je isEqual        ; Jump if equal to 5
    jl isLess         ; Jump if less than 5

isGreater:
    ; Display "The number is greater than 5."
    mov dx, offset msgGreater
    mov ah, 09h
    int 21h
    jmp endProgram
```

```
isEqual:
    ; Display "The number is equal to 5."
    mov dx, offset msgEqual
    mov ah, 09h
    int 21h
    jmp endProgram

isLess:
    ; Display "The number is less than 5."
    mov dx, offset msgLess
    mov ah, 09h
    int 21h

endProgram:
    ; Exit the program
    mov ax, 4C00h
    int 21h
end main
```

Screenshot run:





```
7. ;Filename: EXER39.ASM

;Programmer name: Christian A Nemeno
;Date: OCTOBER 18, 2024
;Description: Create a program that displays a menu for Addition, Subtraction,
Multiplication, and Division.

.model small
.stack 500h
.data
    menu        db 'MATH OPERATIONS'  ,13,10
                db '1. Addition'      ,13,10
                db '2. Subtraction'   ,13,10
                db '3. Multiplication',13,10
                db '4. Division'      ,13,10,'$'

    choice      db 'Enter your choice: $'

    aPrompt     db 'Addition$'
    aPrompt1    db 13,10,'Enter first number: $'
    aPrompt2    db 13,10,'Enter second number: $'
    aDisplay3   db 13,10,'Sum: $'

    sPrompt     db 'Subtraction$'
    sPrompt1    db 13,10,'Enter first number: $'
    sPrompt2    db 13,10,'Enter second number: $'
    sDisplay3   db 13,10,'Difference: $'

    mPrompt     db 'Multiplication$'
    mPrompt1    db 13,10,'Enter first number: $'
    mPrompt2    db 13,10,'Enter second number: $'
    mDisplay3   db 13,10,'Product: $'


    dPrompt     db 'Division$'
    dPrompt1    db 13,10,'Enter first number: $'
    dPrompt2    db 13,10,'Enter second number: $'
    dDisplay3   db 13,10,'Quotient: $'

    eDisplay    db 'Exit Program$'
    invalid     db 'INVALID CHOICE!$'
    ending      db 13,10,'Press Enter to continue.$'
    negSign     db '-$'
    divZero     db 'Error: Division by zero is not allowed.$', 13, 10
```

```
    input1   dw ?
    input2   dw ?
    sum      dw ?
    diff     dw ?
    prod     dw ?
    quo      dw ?

.code

print proc
    mov ah, 09h
    int 21h
    ret
print endp


getNum PROC
    ; Read a single digit from keyboard and store in AX
    mov ah, 01h
    int 21h
    sub al, '0'          ; Convert from ASCII to integer
    mov ah, 0
    ret
getNum ENDP

getChar PROC
    mov ah,01h
    int 21h
    ret
getChar ENDP

converter proc
    push ax
    push bx
    push cx
    push dx

    mov cx, 0            ; Counter for digits
    mov bx, 10           ; Base for decimal conversion

    converter_loop1:
        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
```

```
        cmp ax, 0            ; Check if quotient is zero
        jne converter_loop1      ; Repeat if not

    converter_loop2:
        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 converter_loop2      ; Continue if not

    pop dx
    pop cx
    pop bx
    pop ax
    ret
converter 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
    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

    lea dx, aDisplay3
    call print
    mov ax, sum
    call converter

    call newLine
    ret
addition endp

subtraction proc
    call newLine
    call newLine

    mov ah,09h
    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


    mov ax, input1
    sub ax, input2
    mov diff, ax
    cmp ax, 0
    lea dx, sDisplay3
    call print
    jge display_result

    neg ax
    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
    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

    lea dx, mDisplay3
    call print
    mov ax, prod
    call converter

    call newLine
    ret
multiplication endp

checkAndHandleZero PROC
    cmp bx, 0
    jne continueDivision
```

```
    lea dx, divZero
    call print

    continueDivision:
        ret
checkAndHandleZero ENDP

division proc
    call newLine
    call newLine

    mov ah,09h
    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

    lea dx, dDisplay3
    call print
    mov ax, quo
    call converter

    call newLine
    ret
division 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

        mov ah,09h
        lea dx, menu
        call print

        call newLine

        lea dx, choice
        call print
```

```
        mov ah, 01h
        int 21h

        cmp al,'1'
        je doAdd
        cmp al,'2'
        je doSub
        cmp al,'3'
        je doMult
        cmp al,'4'
        je doDiv
        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
        doInvalid:
            call invalidChoice
            call endingDisplay
            jmp startLoop

end start
```

Screenshot run:

```
MATH OPERATIONS
1. Addition
2. Subtraction
3. Multiplication
4. Division

Enter your choice: 6

INVALID CHOICE!

Press Enter to continue._
```

```
MATH OPERATIONS
1. Addition
2. Subtraction
3. Multiplication
4. Division

Enter your choice: 4

Division
Enter first number: 8
Enter second number: 4
Quotient: 2

Press Enter to continue._
```