1. EXER33.ASM

```
; Filename: EXER33.ASM
; Programmer Name: John Zillion C. Reyes
; 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 0DH, 0AH, 'Enter second number (0-9): $'
    msg3 DB 0DH, 0AH, '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
```

```
D:\>TLINK D:\TEST >>C:\36364.LOG

D:\>D:\TEST
Enter first number (0-9): 6
Enter second number (0-9): 3
The sum is: 9
Do you need to keep the DOSBox [Y,N]?
```

2. EXER34.ASM

```
; Filename: EXER34.ASM
; Programmer Name: John Zillion C. Reyes
; 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 'Enter second number: $'
    resultMsg db '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
```

```
D:\>TLINK D:\TEST >>C:\61970.LOG

D:\>D:\TEST
Enter first number: 5Enter second number: 2The result is: 3
Do you need to keep the DOSBox [Y,N]?
```

3. EXER35.ASM

```
; Filename: EXER35.ASM
; Programmer Name: John Zillion C. Reyes
; 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 'Enter second number (0-9): $'
    resultMsg db '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
```

```
:\>TLINK D:\TEST >>C:\72252.LOG

:\>D:\TEST
nter first number (0-9): 7Enter second number (0-9): 4The result is: L
o you need to keep the DOSBox [Y,N]?
```

4. EXER36.ASM

```
; Filename: EXER36.ASM
; Programmer Name: John Zillion C. Reyes
; 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 'Enter second number (1-9): $' ; Second number cannot be zero
    resultMsg db '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
```

```
D:\>TLINK D:\TEST >>C:\54194.LOG

D:\>D:\TEST
Enter first number (0-9): 8Enter second number (1-9): 4The result is: 2
Do you need to keep the DOSBox [Y,N]?
```

5. EXER37.ASM

```
; Filename: EXER37.ASM
; Programmer Name: John Zillion C. Reyes
; Date: October 18, 2024

.model small
.stack 100h
.data
    input db "Enter a character: $"
    succ db "You entered A.", 0Ah, "$"
    deny db "You entered not A.", 0Ah, "$"
.code
start:
    mov ax, @data
    mov ds ,ax

    mov dx, offset input
    mov ah, 09h
    int 21h

    mov ah, 01h
    int 21h
```

```
    mov bl, al

    mov ah, 02h
    mov dl, 0Ah
    int 21h

    cmp bl, 'A'
    je succPrint
    mov dx, offset deny
    jmp exitSucc
succPrint:
    mov dx, offset succ
exitSucc:
    mov ah, 09h
    int 21h

    int 27h
end start
```

```
D:\>TLINK D:\TEST >>C:\42282.LOG

D:\>D:\TEST
Enter a character: 2
You entered not A.

Do you need to keep the DOSBox [Y,N]?
```

EXER37.asm 6. EXE

6. EXER38.ASM

```
; Filename: EXER38.ASM
; Programmer Name: John Zillion C. Reyes
; Date: October 18, 2024

.model small
.stack 100h
.data
    input db "Enter a number: $"
    equally db "The number is equal to 5.", 0Ah, "$"
    lessy db "The number is less than 5.", 0Ah, "$"
    greedy db "The number is greater than 5.", 0Ah, "$"
.code
start:
    mov ax, @data
```

```
    mov ds ,ax

    mov dx, offset input
    mov ah, 09h
    int 21h

    mov ah, 01h
    int 21h

    mov bl, al

    mov ah, 02h
    mov dl, 0Ah
    int 21h

    sub bl, '0'

    cmp bl, 5
    je succPrint
    jg greedyPrint
    mov dx, offset lessy
    jmp exitSucc
greedyPrint:
    mov dx, offset greedy
    jmp exitSucc
succPrint:
    mov dx, offset equally
exitSucc:
    mov ah, 09h
    int 21h

    int 27h

end start
```

```
D:\>TLINK D:\TEST >>C:\75802.LOG

D:\>D:\TEST
Enter a number: 2
The number is less than 5.

Do you need to keep the DOSBox [Y,N]?_
                c:\Users\L12X17W33\Documents\CIT_TryHard\BSCS-2 1st\CS243\EXER39
```

7. EXER39.ASM

```
; Filename: EXER39.ASM
; Programmer Name: John Zillion C. Reyes
; Date: October 18, 2024

.model small
.stack 100
.data
    h1 db "MATH OPERATIONS", 0Ah, 0Ah, "$"
    hA db "1. Addition", 0Ah, "$"
    hS db "2. Subtraction", 0Ah, "$"
    hM db "3. Multiplication", 0Ah, "$"
    hD db "4. Division", 0Ah, "Enter your choice: $"

    inA0 db "Addition", 0Ah, "$"
    inA1 db "Enter first addend: $"
    inA2 db "Enter second addend: $"
    outA3 db "Sum: $"

    inS0 db "Subtraction", 0Ah, "$"
    inS1 db "Enter minuend: $"
    inS2 db "Enter subtrahend: $"
    outS3 db "Difference: $"

    inM0 db "Multiplication", 0Ah, "$"
    inM1 db "Enter multiplicand: $"
    inM2 db "Enter multiplier: $"
    outM3 db "Product: $"

    inD0 db "Division", 0Ah, "$"
    inD1 db "Enter dividend: $"
    inD2 db "Enter divisor: $"
    outD3 db "Quotient: $"

    outE db "Exit Program", 0Ah, "$"
    outN db "INVALID CHOICE!", 0Ah, "$"

    cls db "                                ", 0Dh ,
"                              ", 0Dh, "$"
.code

start:
    mov ax, @data
```

```
    mov ds, ax

    mov ah, 00h
    mov al, 03h
    int 10h

    lea dx, h1
    call printString

    lea dx, hA
    call printString

    lea dx, hS
    call printString


    lea dx, hM
    call printString


    lea dx, hD
    call printString

    mov ah, 01h
    int 21h

    call endLine
    call endLine

    cmp al, '1'
    jne nAdd
    call opAdd
    jmp exit
nAdd:
    cmp al, '2'
    jne nSub
    call opSub
    jmp exit
nSub:
    cmp al, '3'
    jne nMul
    call opMul
    jmp exit
nMul:
    cmp al, '4'
```

```
    call opDiv
exit:

    int 27h

opAdd:
    push dx
    push cx
    push bx
    push ax

    lea dx, inA0
    call printString
    lea di, inA1
    call inputNum
    mov bx, ax
    lea di, inA2
    call inputNum
    mov cx, ax


    add ax, bx
    lea dx, outA3
    call printString
    call printNum
    call endline

    pop ax
    pop bx
    pop cx
    pop dx
    ret
opSub:
    push dx
    push cx
    push bx
    push ax

    lea dx, inS0
    call printString
    lea di, inS1
    call inputNum
    mov bx, ax
    lea di, inS2
    call inputNum
```

```
    mov cx, ax


    sub bx, ax
    mov ax, bx
    lea dx, outS3
    call printString
    call printNum
    call endline

    pop ax
    pop bx
    pop cx
    pop dx
    ret
opMul:
    push dx
    push cx
    push bx
    push ax

    lea dx, inM0
    call printString
    lea di, inM1
    call inputNum
    mov bx, ax
    lea di, inM2
    call inputNum
    mov cx, ax


    mul bx
    lea dx, outM3
    call printString
    call printNum
    call endline

    pop ax
    pop bx
    pop cx
    pop dx
    ret
opDiv:
    push dx
    push cx
```

```
    push bx
    push ax

    lea dx, inD0
    call printString
    lea di, inD1
    call inputNum
    mov bx, ax
    lea di, inD2
    call inputNum
    mov cx, ax


    mov ax, bx
    mov dx, 0
    div cx
    lea dx, outD3
    call printString
    call printNum
    call endline

    pop ax
    pop bx
    pop cx
    pop dx
    ret

printString:
    push ax

    mov ah, 09h
    int 21h

    pop ax
    ret

endLine:
    push ax
    push dx

    mov ah, 02h
    mov dl, 0Ah
    int 21h

    pop dx
```

```
    pop ax
    ret

isOdd:
    push bx
    push dx
    mov dx, 0
    mov bx, 2
    div bx

    cmp dx, 1

    pop dx
    pop bx
    ret


inputNum:
    push bx
    push cx
    push dx
    push si
    mov cx, 0
    mov si, 10
    mov bx, 0
    mov dx, di
    call printString
inputLoop:
    mov ah, 7
    int 21h

    cmp al, 8
    je inputRem
    cmp al, '0'
    jl exitInputNum
    cmp al, '9'
    jg exitInputNum
    sub al, '0'
    mov cl, al
    mov ax, bx
    mul si

    add ax, cx
    mov bx, ax
```

```
    mov dx, offset cls
    call printString
    mov dx, di
    call printString
    mov ax, bx
    call printNum

    jmp inputLoop
inputRem:
    mov ax, bx
    mov dx, 0
    div si
    mov bx, ax

    mov dx, offset cls
    call printString
    mov dx, di
    call printString
    mov ax, bx
    call printNum

    jmp inputLoop

exitInputNum:
    call endLine
    mov ax, bx

    pop si
    pop dx
    pop cx
    pop bx
    ret


printNum:
    push ax
    push bx
    push cx
    push dx
    push si

    mov cx, 0

digitLoop:
    mov bx, 10
```

```
    mov dx, 0
    div bx

    mov bx, ax
    mov ax, cx
    call isOdd

    je ifAppend
    mov ah, dl
    mov al, 0
    jmp endifAppend
ifAppend:
    pop ax
    mov al, dl
endifAppend:
    push ax
    inc cx

    mov ax, bx

    cmp ax, 0
    jne digitLoop

printLoop:

    mov ax, cx
    call isOdd

    pop ax
    je ifPrint
    mov dl, al
    push ax
    jmp endifPrint
ifPrint:
    mov dl, ah
endifPrint:
    add dl, '0'
    mov ah, 02h
    int 21h

    loop printLoop

    pop si
    pop dx
    pop cx
```

```
    pop bx
    pop ax

    ret
end start
```