Christian A Nemeno   
CS243 F3 LABORATORY HANDS-ON EXERCISES

1. **EXER25.ASM**

; Filename: EXER25.ASM

; Programmer Name: Christian A. Nemeno

; Date: September 29, 2024

; Description: This assembly language program will get 3 string inputs and

; display back the 3 strings on separate lines.

.MODEL SMALL

.STACK 100h

.DATA

inputString1 DB 50 DUP('$') ; Reserve 50 bytes for input string

inputString2 DB 50 DUP('$')

inputString3 DB 50 DUP('$')

promptString1 DB 'Enter first string: $'

promptString2 DB 'Enter second string: $'

promptString3 DB 'Enter third string: $'

outputString1 DB 'You entered first string: $'

outputString2 DB 'You entered second string: $'

outputString3 DB 'You entered third string: $'

.CODE

MAIN PROC

; Initialize data segment

MOV AX, @DATA

MOV DS, AX

; Prompt the user for input first string

LEA DX, promptString1

MOV AH, 09h ; DOS function to display a string

INT 21h

; Read input first string

LEA DX, inputString1

MOV AH, 0Ah ; DOS function to read a string

INT 21h

; this is for displaying new line

mov ah,02h

mov cl,0Ah ; 0Ah is new line

mov dl,cl

int 21h

; Prompt the user for input second string

LEA DX, promptString2

MOV AH, 09h ; DOS function to display a string

INT 21h

; Read input second string

LEA DX, inputString2

MOV AH, 0Ah ; DOS function to read a string

INT 21h

; this is for displaying new line

mov ah,02h

mov cl,0Ah ; 0Ah is new line

mov dl,cl

int 21h

; Prompt the user for input third string

LEA DX, promptString3

MOV AH, 09h ; DOS function to display a string

INT 21h

; Read input third string

LEA DX, inputString3

MOV AH, 0Ah ; DOS function to read a string

INT 21h

; this is for displaying new line

mov ah,02h

mov cl,0Ah ; 0Ah is new line

mov dl,cl

int 21h

; Display the output message for first string

LEA DX, outputString1

MOV AH, 09h ; DOS function to display a string

INT 21h

; Display the entered first string

LEA DX, inputString1 + 2 ; Skip the first two bytes (length and max length)

MOV AH, 09h ; DOS function to display a string

INT 21h

; this is for displaying new line

mov ah,02h

mov cl,0Ah ; 0Ah is new line

mov dl,cl

int 21h

; Display the output message for second string

LEA DX, outputString2

MOV AH, 09h ; DOS function to display a string

INT 21h

; Display the entered second string

LEA DX, inputString2 + 2 ; Skip the first two bytes (length and max length)

MOV AH, 09h ; DOS function to display a string

INT 21h

; this is for displaying new line

mov ah,02h

mov cl,0Ah ; 0Ah is new line

mov dl,cl

int 21h

; Display the output message for third string

LEA DX, outputString3

MOV AH, 09h ; DOS function to display a string

INT 21h

; Display the entered third string

LEA DX, inputString3 + 2 ; Skip the first two bytes (length and max length)

MOV AH, 09h ; DOS function to display a string

INT 21h

; Exit program

MOV AX, 4C00h ; DOS function to terminate program

INT 21h

MAIN ENDP

END MAIN

1. **EXER26.ASM**

;

; Filename: EXER26.ASM

; Programmer Name: Christian A. Nemeno

; Date: September 28, 2024

; Description: Make a program to input two integers, add the two integers,

;and display the sum of the two integers.

; note please hit enter upon entering the integer it reads more than one integer

; does not handle negative cases

.MODEL SMALL

.STACK 100H

.DATA

prompt1 DB 'Enter first integer: $'

prompt2 DB 0DH, 0AH, 'Enter second integer: $'

result\_msg DB 0DH, 0AH, 'The sum is: $'

num1 DW ?

num2 DW ?

sum DW ?

.CODE

MAIN PROC

; Initialize data segment

MOV AX, @DATA

MOV DS, AX

; Display first prompt

LEA DX, prompt1

MOV AH, 09H

INT 21H

; Input first number

CALL INPUT\_NUM

MOV num1, AX

; Display second prompt

LEA DX, prompt2

MOV AH, 09H

INT 21H

; Input second number

CALL INPUT\_NUM

MOV num2, AX

; Calculate sum

MOV AX, num1

ADD AX, num2

MOV sum, AX

; Display result message

LEA DX, result\_msg

MOV AH, 09H

INT 21H

; Display sum

MOV AX, sum

CALL DISPLAY\_NUM

; Exit program

MOV AH, 4CH

INT 21H

MAIN ENDP

INPUT\_NUM PROC

XOR BX, BX ; Initialize result to 0

INPUT\_LOOP:

; Read a character

MOV AH, 01H

INT 21H

; Check if it's Enter (carriage return)

CMP AL, 0DH

JE END\_INPUT

; Convert ASCII to number

SUB AL, '0'

; Update result: result = result \* 10 + new\_digit

MOV CL, AL

MOV AX, 10

MUL BX

MOV BX, AX

XOR CH, CH

ADD BX, CX

JMP INPUT\_LOOP

END\_INPUT:

MOV AX, BX

RET

INPUT\_NUM ENDP

DISPLAY\_NUM PROC

MOV BX, 10

XOR CX, CX ; Initialize digit counter

DIVIDE\_LOOP:

XOR DX, DX

DIV BX

PUSH DX ; Push remainder (digit) onto stack

INC CX

TEST AX, AX

JNZ DIVIDE\_LOOP

DISPLAY\_LOOP:

POP DX

ADD DL, '0' ; Convert to ASCII

MOV AH, 02H

INT 21H

LOOP DISPLAY\_LOOP

RET

DISPLAY\_NUM ENDP

END MAIN

1. **EXER27.ASM**

;

; Filename: EXER27.ASM

; Programmer Name: Christian A. Nemeno

; Date: September 28, 2024

; Description: Make a program to input two integers, subtract the second

;integer from the first integer, and display the difference

;of the two integers.

; display back the difference

; note please hit enter upon entering the integer , it handles more than 1 digit

.MODEL SMALL

.STACK 100H

.DATA

prompt1 DB 'Enter first integer: $'

prompt2 DB 0DH, 0AH, 'Enter second integer: $'

result\_msg DB 0DH, 0AH, 'The difference is: $'

num1 DW ?

num2 DW ?

difference DW ?

.CODE

MAIN PROC

; Initialize data segment

MOV AX, @DATA

MOV DS, AX

; Display first prompt

LEA DX, prompt1

MOV AH, 09H

INT 21H

; Input first number

CALL INPUT\_NUM

MOV num1, AX

; Display second prompt

LEA DX, prompt2

MOV AH, 09H

INT 21H

; Input second number

CALL INPUT\_NUM

MOV num2, AX

; Calculate difference

MOV AX, num1

SUB AX, num2

MOV difference, AX

; Display result message

LEA DX, result\_msg

MOV AH, 09H

INT 21H

; Display difference

MOV AX, difference

CALL DISPLAY\_NUM

; Exit program

MOV AH, 4CH

INT 21H

MAIN ENDP

INPUT\_NUM PROC

XOR BX, BX ; Initialize result to 0

INPUT\_LOOP:

; Read a character

MOV AH, 01H

INT 21H

; Check if it's Enter (carriage return)

CMP AL, 0DH

JE END\_INPUT

; Convert ASCII to number

SUB AL, '0'

; Update result: result = result \* 10 + new\_digit

MOV CL, AL

MOV AX, 10

MUL BX

MOV BX, AX

XOR CH, CH

ADD BX, CX

JMP INPUT\_LOOP

END\_INPUT:

MOV AX, BX

RET

INPUT\_NUM ENDP

DISPLAY\_NUM PROC

TEST AX, AX

JNS POSITIVE

PUSH AX

MOV DL, '-'

MOV AH, 02H

INT 21H

POP AX

NEG AX

POSITIVE:

MOV BX, 10

XOR CX, CX ; Initialize digit counter

DIVIDE\_LOOP:

XOR DX, DX

DIV BX

PUSH DX ; Push remainder (digit) onto stack

INC CX

TEST AX, AX

JNZ DIVIDE\_LOOP

DISPLAY\_LOOP:

POP DX

ADD DL, '0' ; Convert to ASCII

MOV AH, 02H

INT 21H

LOOP DISPLAY\_LOOP

RET

DISPLAY\_NUM ENDP

END MAIN

1. **EXER28.ASM**

;

; Filename: EXER28.ASM

; Programmer Name: Christian A. Nemeno

; Date: September 28, 2024

; Description: Make a program to input two integers, multiply the two

;integers, and display the product of the two integers.

; please hit enter after each input the code handles more than one digit

.MODEL SMALL

.STACK 100H

.DATA

prompt1 DB 'Enter first positive integer: $'

prompt2 DB 0DH, 0AH, 'Enter second positive integer: $'

result\_msg DB 0DH, 0AH, 'The product is: $'

num1 DW ?

num2 DW ?

product DW ?

.CODE

MAIN PROC

; Initialize data segment

MOV AX, @DATA

MOV DS, AX

; Display first prompt

LEA DX, prompt1

MOV AH, 09H

INT 21H

; Input first number

CALL INPUT\_NUM

MOV num1, AX

; Display second prompt

LEA DX, prompt2

MOV AH, 09H

INT 21H

; Input second number

CALL INPUT\_NUM

MOV num2, AX

; Calculate product

MOV AX, num1

MUL num2 ; Unsigned multiplication

MOV product, AX

; Display result message

LEA DX, result\_msg

MOV AH, 09H

INT 21H

; Display product

MOV AX, product

CALL DISPLAY\_NUM

; Exit program

MOV AH, 4CH

INT 21H

MAIN ENDP

INPUT\_NUM PROC

XOR BX, BX ; Initialize result to 0

INPUT\_LOOP:

; Read a character

MOV AH, 01H

INT 21H

; Check if it's Enter (carriage return)

CMP AL, 0DH

JE END\_INPUT

; Convert ASCII to number

SUB AL, '0'

; Update result: result = result \* 10 + new\_digit

MOV CL, AL

MOV AX, 10

MUL BX

MOV BX, AX

XOR CH, CH

ADD BX, CX

JMP INPUT\_LOOP

END\_INPUT:

MOV AX, BX

RET

INPUT\_NUM ENDP

DISPLAY\_NUM PROC

MOV BX, 10

XOR CX, CX ; Initialize digit counter

DIVIDE\_LOOP:

XOR DX, DX

DIV BX

PUSH DX ; Push remainder (digit) onto stack

INC CX

TEST AX, AX

JNZ DIVIDE\_LOOP

DISPLAY\_LOOP:

POP DX

ADD DL, '0' ; Convert to ASCII

MOV AH, 02H

INT 21H

LOOP DISPLAY\_LOOP

RET

DISPLAY\_NUM ENDP

END MAIN

1. **EXER29.ASM**

;

; Filename: EXER29.ASM

; Programmer Name: Christian A. Nemeno

; Date: September 28, 2024

; Description: This assembly language program will get 2 integers

; display back the quotient

; please hit enter after each input the code handles more than one digit

.MODEL SMALL

.STACK 100H

.DATA

prompt1 DB 'Enter first positive integer (dividend): $'

prompt2 DB 0DH, 0AH, 'Enter second positive integer (divisor): $'

result\_msg DB 0DH, 0AH, 'The quotient is: $'

error\_msg DB 0DH, 0AH, 'Error: Division by zero!$'

dividend DW ?

divisor DW ?

quotient DW ?

.CODE

MAIN PROC

; Initialize data segment

MOV AX, @DATA

MOV DS, AX

; Display first prompt

LEA DX, prompt1

MOV AH, 09H

INT 21H

; Input dividend

CALL INPUT\_NUM

MOV dividend, AX

; Display second prompt

LEA DX, prompt2

MOV AH, 09H

INT 21H

; Input divisor

CALL INPUT\_NUM

MOV divisor, AX

; Check for division by zero

CMP AX, 0

JE DIVISION\_ERROR

; Perform division

MOV AX, dividend

XOR DX, DX ; Clear DX for division

DIV divisor

MOV quotient, AX

; Display result message

LEA DX, result\_msg

MOV AH, 09H

INT 21H

; Display quotient

MOV AX, quotient

CALL DISPLAY\_NUM

JMP EXIT\_PROGRAM

DIVISION\_ERROR:

; Display error message

LEA DX, error\_msg

MOV AH, 09H

INT 21H

EXIT\_PROGRAM:

; Exit program

MOV AH, 4CH

INT 21H

MAIN ENDP

INPUT\_NUM PROC

XOR BX, BX ; Initialize result to 0

INPUT\_LOOP:

; Read a character

MOV AH, 01H

INT 21H

; Check if it's Enter (carriage return)

CMP AL, 0DH

JE END\_INPUT

; Convert ASCII to number

SUB AL, '0'

; Update result: result = result \* 10 + new\_digit

MOV CL, AL

MOV AX, 10

MUL BX

MOV BX, AX

XOR CH, CH

ADD BX, CX

JMP INPUT\_LOOP

END\_INPUT:

MOV AX, BX

RET

INPUT\_NUM ENDP

DISPLAY\_NUM PROC

MOV BX, 10

XOR CX, CX ; Initialize digit counter

DIVIDE\_LOOP:

XOR DX, DX

DIV BX

PUSH DX ; Push remainder (digit) onto stack

INC CX

TEST AX, AX

JNZ DIVIDE\_LOOP

DISPLAY\_LOOP:

POP DX

ADD DL, '0' ; Convert to ASCII

MOV AH, 02H

INT 21H

LOOP DISPLAY\_LOOP

RET

DISPLAY\_NUM ENDP

END MAIN

1. **EXER30.ASM**

;

; Filename: EXER30.ASM

; Programmer Name: Christian A Nemeno

; Date: September 28, 2024

; Description : Make a program to input two integers. Compare the two

; integers. Display which one is smaller and bigger of the

; two integers.

.model small

.stack 100h

.data

input1 db 'Enter first integer: $'

input2 db 13,10,'Enter second integer: $'

in1 dw ?

in2 dw ?

big db 20 DUP ('$')

smol db 20 DUP ('$')

res db 13,10,'Bigger: $'

res1 db 13,10,'Smaller: $'

.code

itoa:

push ax

push bx

push cx

push dx

mov cx, 0 ; Counter for digits

mov bx, 10 ; Base for decimal conversion

itoa\_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 itoa\_loop1 ; Repeat if not

itoa\_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 itoa\_loop2 ; Continue if not

pop dx

pop cx

pop bx

pop ax

ret

start:

mov ax, @data

mov ds, ax

lea dx, input1

mov ah, 9

int 21h

mov ah, 01h

int 21h

sub al, '0'

mov ah, 0

mov in1, ax

lea dx, input2

mov ah, 9

int 21h

mov ah, 01h

int 21h

sub al, '0'

mov ah, 0

mov in2, ax

mov ax, in1

mov bx, in2

cmp ax, bx

jg isbigger ; Jump to isbigger if in1 > in2

jl islower ; Jump to islower if in1 < in2

isbigger:

lea dx, res

mov ah, 9

int 21h

mov ax, in1

call itoa

lea dx, res1

mov ah, 9

int 21h

mov ax, in2

call itoa

jmp done

islower:

lea dx, res

mov ah, 9

int 21h

mov ax, in2

call itoa

lea dx, res1

mov ah, 9

int 21h

mov ax, in1

call itoa

done:

mov ah, 4Ch

int 21h

end start

1. **EXER31.ASM**

;

; Filename: EXER31.ASM

; Programmer Name: Christian A. Nemeno

; Date: September 28, 2024

; Description: Create a program to input an integer. Using a loop, display

;from 1 to the integer value entered by the user.

; display until nth number

.model small

.stack 100h

.data

prompt db 'Enter an integer: $'

newline db 13, 10, '$'

.code

main proc

mov ax, @data

mov ds, ax

; Display prompt

lea dx, prompt

mov ah, 9

int 21h

; Read integer (assume single digit for simplicity)

mov ah, 1

int 21h

sub al, '0' ; Convert ASCII to number

mov cl, al ; Store the count in CL

; Print newline

lea dx, newline

mov ah, 9

int 21h

; Initialize counter

mov ch, 1 ; CH will be our loop counter, starting from 1

print\_loop:

; Print current number

mov dl, ch

add dl, '0' ; Convert number to ASCII

mov ah, 2

int 21h

; Print newline

lea dx, newline

mov ah, 9

int 21h

; Increment counter

inc ch

; Compare counter with input

cmp ch, cl

jle print\_loop ; If CH <= CL, continue loop

; Exit program

mov ah, 4Ch

int 21h

main endp

end main

1. **EXER32.ASM**

;

;FILENAME: EXER32.ASM

;STUDENT NAME: Christian A. Nemeno

;DATE FINISHED: SEPTEMBER 28,2024

;Description: Copy your code from the 2nd Lab Hands-on Exam. Edit your

;code to perform the actual math operations and display the

;results.

.model small

.stack 200h

.data

inputnum1 db ?

inputnum2 db ?

head DB 13,10,'THE CALCULATOR',13,10,'Created by: CHRISTIAN A.NEMENO',13,10,'DATE: 9/28/2024',13,10,'$'

head1 DB 13,10,'Main Menu',13,10,'$'

choice1 DB 'a - Addition',13,10,'$'

choice2 DB 's - Subtraction',13,10,'$'

choice3 DB 'm - Multiplication',13,10,'$'

choice4 DB 'd - Division',13,10,'$'

choice5 DB 'e - Exit',13,10,'$'

choice DB 'Enter your choice: $'

add1 DB 'Addition',13,10,'$'

add2 DB 'Enter first addend: $'

add3 DB 13,10,'Enter second addend: $'

add4 DB 13,10,'First addend is: $'

add5 DB 13,10,'Second addend is: $'

sub1 DB 'Subtraction',13,10,'$'

sub2 DB 'Enter minuend: $'

sub3 DB 13,10,'Enter subtrahend: $'

sub4 DB 13,10,'Minuend is: $'

sub5 DB 13,10,'Subtrahend is: $'

mul1 DB 'Multiplication',13,10,'$'

mul2 DB 'Enter multiplicand: $'

mul3 DB 13,10,'Enter multiplier: $'

mul4 DB 13,10,'Multiplicand is: $'

mul5 DB 13,10,'Multiplier is: $'

div1 DB 'Division',13,10,'$'

div2 DB 'Enter dividend: $'

div3 DB 13,10,'Enter divisor: $'

div4 DB 13,10,'Dividend is: $'

div5 DB 13,10,'Divisor is: $'

term DB 'Exit Program$'

inv DB 'INVALID CHOICE!','$'

continuePrompt db 13,10,13,10,'Enter to continue...', '$'

bye DB 13,10,'Thank you.','$'

nxt DB 0dh, 0ah, '$'

in1 dw ?

in2 dw ?

in3 db 20 DUP ('$')

in4 db 20 DUP ('$')

su dw ?

dff dw ?

mu dw ?

dv dw ?

suma db 20 DUP ('$')

difa db 20 DUP ('$')

mula db 20 DUP ('$')

diva db 20 DUP ('$')

sum db 13,10,'Sum: $'

diff db 13,10,'Difference: $'

mult db 13,10,'Product: $'

divi db 13,10,'Quotient: $'

.CODE

itoa:

push ax

push bx

push cx

push dx

mov cx, 0

mov bx, 10

loop1:

xor dx, dx

div bx

push dx

inc cx

cmp ax, 0

jne loop1

loop2:

pop dx

add dl, '0'

mov ah, 02h

int 21h

dec cx

cmp cx, 0

jne loop2

pop dx

pop cx

pop bx

pop ax

ret

clearScreen:

mov ah, 00h

mov al, 03h

int 10h

ret

printString:

mov ah, 09h

mov bl, 07h

int 21h

ret

nxln:

lea dx, nxt

MOV ah, 09h

INT 21h

RET

getChar:

mov ah,01h

int 21h

ret

getNum:

mov ah,01h

int 21h

sub al, '0'

mov ah,0

RET

displayNum:

int 21h

RET

InAdd:

mov ah,09h

mov bl,20h

mov cx,8

int 10h

LEA dx, add1

CALL printString

lea dx, add2

CALL printString

call getNum

mov in1, ax

lea dx, add3

CALL printString

CALL getNum

mov in2,ax

mov dx, in1

add dx, in2

mov su, dx

LEA dx, add4

CALL printString

mov ax, in1

call itoa

mov dx, offset in3

mov ah, 9

int 21h

LEA dx, add5

CALL printString

mov ax, in2

call itoa

mov dx, offset in4

mov ah, 9

int 21h

lea dx, sum

mov ah, 9

int 21h

mov ax, su

call itoa

mov dx, offset suma

mov ah, 9

int 21h

; Prompt to continue

lea dx, continuePrompt

mov ah, 09h

int 21h

CALL getChar

CALL clearScreen

jmp loop\_start

InSub:

mov ah,09h

mov bl,30h

mov cx,11

int 10h

lea dx, sub1

CALL printString

lea dx, sub2

CALL printString

CALL getNum

mov in1, ax

lea dx, sub3

CALL printString

CALL getNum

mov in2,ax

mov dx, in1

sub dx, in2

mov dff, dx

LEA dx, sub4

CALL printString

mov ax, in1

call itoa

mov dx, offset in3

mov ah, 9

int 21h

LEA dx, sub5

CALL printString

mov ax, in2

CALL itoa

mov dx, offset in4

mov ah, 9

int 21h

lea dx, diff

mov ah,9

int 21h

mov ax, dff

call itoa

mov dx, offset difa

mov ah, 9

int 21h

lea dx, continuePrompt

mov ah, 09h

int 21h

CALL getChar

CALL clearScreen

jmp loop\_start

multiply:

mov ah,09h

mov bl,40h

mov cx,14

int 10h

mov dx, OFFSET mul1

call printString

mov dx, OFFSET mul2

call printString

call getNum

mov in1, ax

mov dx, OFFSET mul3

call printString

call getNum

mov in2, ax

mov ax, in1

mov dx, in2

mul dx

mov mu, ax

LEA dx, mul4

CALL printString

mov ax, in1

call itoa

mov dx, offset in3

mov ah, 9

int 21h

LEA dx, mul5

CALL printString

mov ax, in2

call itoa

mov dx, offset in4

mov ah, 9

int 21h

lea dx, mult

mov ah, 9

int 21h

mov ax, mu

call itoa

mov dx, offset mula

mov ah, 9

int 21h

lea dx, continuePrompt

mov ah, 09h

int 21h

CALL getChar

CALL clearScreen

jmp loop\_start

ProgramStart:

mov ax, @data

mov ds, ax

loop\_start:

CALL clearScreen

LEA dx, head

CALL printString

LEA dx, head1

CALL printString

mov ah,09h

mov bl,20h

mov cx,12

int 10h

LEA dx, choice1

CALL printString

mov ah,09h

mov bl,30h

mov cx,15

int 10h

LEA dx, choice2

CALL printString

mov ah,09h

mov bl,40h

mov cx,18

int 10h

LEA dx, choice3

CALL printString

mov ah,09h

mov bl,50h

mov cx,12

int 10h

LEA dx, choice4

CALL printString

mov ah,09h

mov bl,70h

mov cx,8

int 10h

LEA dx, choice5

CALL PrintString

LEA dx, choice

CALL printString

CALL getChar

; DECISION

cmp al,'a'

je Addition

cmp al,'s'

je Subtraction

cmp al,'m'

je Multiplication

cmp al,'d'

je Division

cmp al,'e'

je Exit

jne InvalidInput

InvalidInput:

CALL nxln

CALL nxln

mov ah,09h

mov bl,11001110B

mov cx,15

int 10h

LEA dx, inv

CALL printString

CALL stop

Exit:

CALL nxln

CALL nxln

mov ah,09h

mov bl,70h

mov cx,12

int 10h

LEA dx, term

CALL printString

CALL stop

Addition:

CALL nxln

CALL nxln

jmp InAdd

Subtraction:

CALL nxln

CALL nxln

jmp InSub

Multiplication:

call nxln

call nxln

jmp multiply

Division:

CALL nxln

CALL nxln

mov ah,09h

mov bl,50h

mov cx,8

int 10h

LEA dx, div1

CALL printString

LEA dx, div2

CALL printString

CALL getNum

mov in1, ax

LEA dx, div3

CALL printString

CALL getNum

mov in2, ax

mov ax, in1

xor dx, dx

mov bx, in2

div bx

mov dv, ax

LEA dx, div4

CALL printString

mov ax, in1

call itoa

mov dx, offset in3

mov ah, 9

int 21h

LEA dx, div5

CALL printString

mov ax, in2

call itoa

mov dx, offset in4

mov ah, 9

int 21h

lea dx, divi

mov ah, 9

int 21h

mov ax, dv

call itoa

mov dx, offset diva

mov ah, 9

int 21h

lea dx, continuePrompt

mov ah, 09h

int 21h

CALL getChar

CALL clearScreen

jmp loop\_start

stop:

CALL nxln

LEA dx, bye

CALL printString

mov ax,4C00h

int 21h

END ProgramStart