Assembly Language Code

loop:

MOV AX, [loopcount]

cmp ax, 0x06

jge signs

call readbx

cmp bx,0x0a

jeq dontwrite

call writebx

dontwrite:

MOV AX, [loopcount]

inc ax

mov [loopcount], ax

jmp loop

finished:

jmp loopreset

halt

org 0x100

loopcount:dc.w 0x00

firstnum:dc.w 0x00

sign:dc.w 0x00

secondnum:dc.w0x00

anwser:dc.w 0x00

digit1:dc.w 0x00

digit2:dc.w 0x00

thirdnum:dc.w 0x00

fourthnum:dc.w 0x00

number1:dc.w 0x00

number2:dc.w 0x00

digit3:dc.w 0x00

//checks 3 digits

checking 3:

mov bx,[anwser]

cmp bx,0x64

JGE printingfull

ret

// converts 3 digits

printingfull:

mov ax,[anwser]

div ax,0x64

mov [digit1], ax

mov ax,[anwser]

mod ax, 0x64

div ax,0x0a

mov[digit2],ax

mov ax,[anwser]

mod ax, 0x0a

mov [digit3],ax

jmp printing3

//prints 3 digits

printing3:

mov bx,[digit1]

call bxtoascii

call writebx

mov bx,[digit2]

call bxtoascii

call writebx

mov bx,[digit3]

call bxtoascii

call writebx

jmp finished

//checks for double digits

checking2:

call checking 3

mov bx,[anwser]

cmp bx,0x0A

JGE printfull

ret

//converts to double digits

printfull:

mov ax,[anwser]

div ax,0x0A

mov [digit1],ax

mov ax,[anwser]

mod ax,0x0A

mov[digit2],ax

jmp Printing2

//prints double digits

printing2:

mov bx,[digit1]

call bxtoascii

call writebx

mov bx,[digit2]

call bxtoascii

call writebx

jmp finished

//print equal sign

equalsign:

mov bx, 0x3d

call writebx

jmp print

//prints single digit

print:

call checking2

call bxtoascii

call writebx

jmp finished

//checks the sign

Signs:

mov ax,[sign]

cmp ax,0x2b

Jeq addding

cmp ax, 0x2d

jeq subing

cmp ax,0x2a

jeq multi

cmp ax,0x2f

jeq diving

jmp finished

//does the adding

addding:

mov ax, [firstnum]

mov bx, ax

sub ax, 0x30

mov bx, [thirdnum]

sub bx, 0x30

add ax, bx

mov [number1], ax

mov ax, [secondnum]

sub ax, 0x30

mov bx, [fourthnum]

sub bx, 0x30

add ax, bx

mov [number2], ax

mov ax, [number1]

mul ax, 0x0a

mov bx, [number2]

add ax, bx

mov [anwser],ax

jmp equalsign

//does the subbing

subing:

mov bx,[firstnum]

sub bx,0x30

mov [firstnum],bx

mov bx,[secondnum]

sub bx,0x30

mov [secondnum],bx

mov ax,[firstnum]

mul ax, 0x0a

mov bx,[secondnum]

add ax, bx

mov [number1],ax

mov bx,[thirdnum]

sub bx,0x30

mov [thirdnum],bx

mov bx,[fourthnum]

sub bx,0x30

mov [fourthnum],bx

mov ax,[thirdnum]

mul ax, 0x0a

mov bx,[fourthnum]

add ax, bx

mov bx,[number1]

sub bx,ax

mov [anwser],bx

jmp equalsign

//does multiply

multi:

mov bx,[firstnum]

sub bx,0x30

mov [firstnum],bx

mov bx,[secondnum]

sub bx,0x30

mov [secondnum],bx

mov ax,[firstnum]

mul ax, 0x0a

mov bx,[secondnum]

add ax, bx

mov [number1],ax

mov bx,[thirdnum]

sub bx,0x30

mov [thirdnum],bx

mov bx,[fourthnum]

sub bx,0x30

mov [fourthnum],bx

mov ax,[thirdnum]

mul ax, 0x0a

mov bx,[fourthnum]

add ax, bx

mov bx,[number1]

mul bx,ax

mov [anwser],bx

jmp equalsign

//does divide

diving:

mov bx,[firstnum]

sub bx,0x30

mov [firstnum],bx

mov bx,[secondnum]

sub bx,0x30

mov [secondnum],bx

mov ax,[firstnum]

mul ax, 0x0a

mov bx,[secondnum]

add ax, bx

mov [number1],ax

mov bx,[thirdnum]

sub bx,0x30

mov [thirdnum],bx

mov bx,[fourthnum]

sub bx,0x30

mov [fourthnum],bx

mov ax,[thirdnum]

mul ax, 0x0a

mov bx,[fourthnum]

add ax, bx

mov bx,[number1]

div bx,ax

mov [anwser],bx

jmp equalsign

bxfromascii:

sub bx,0x30

ret

bxtoascii:

add bx,0x30

ret

readbx:

mov ax,[instat]

cmp ax,0x00

jeq readbx

mov bx,[indata]

mov ax,0x00

mov [instat],ax

call check

ret

writebx:

mov ax,[outstat]

cmp ax,0x0

jne writebx

mov[outdata],bx

mov bx,0x1

mov [outstat],bx

ret

//checks the loopcount

check:

mov ax, [loopcount]

cmp ax, 0x00

jeq assignfirst

cmp ax, 0x01

jeq assignsecond

cmp ax,0x02

jeq assignsign

cmp ax, 0x03

jeq assignthird

cmp ax, 0x04

jeq assignfourth

ret

assignfourth:

mov[fourthnum],bx

call digitcheck4

ret

assignthird:

mov [thirdnum],bx

call digitcheck3

ret

assignsecond:

mov [secondnum], bx

call digitcheck2

RET

assignfirst:

mov [firstnum], bx

call digitcheck1

RET

assignsign:

mov [sign], bx

ret

digitcheck1:

mov BX,[firstnum]

cmp bx, 0x40

jge error

ret

digitcheck2:

mov BX,[secondnum]

cmp bx, 0x40

jge error

ret

digitcheck3:

mov BX,[Thirdnum]

cmp bx, 0x40

jge error

ret

digitcheck4:

mov BX,[fourthnum]

cmp bx, 0x40

jge error

ret

error:

mov bx,'E'

call writebx

mov bx,'r'

call writebx

mov bx,'r'

call writebx

mov bx,'o'

call writebx

mov bx,'r'

call writebx

jmp finished

loopreset:

mov ax,0x00

mov [loopcount],ax

mov bx, 0x0a

call writebx

jmp loop

outstat: equ 0xf2

outdata:equ 0xf0

instat: equ 0xf5

indata: equ 0xf3