

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №2

Вычисления с вводом данных и выводом результата

Выполнил: студент гр. 053502
Герчик Артём Вадимович

Руководитель: ст. преподаватель
Шиманский В.В.

Минск 2021

СОДЕРЖАНИЕ

1. Введение
2. Постановка задачи
3. Программная реализация
4. Выводы

Литература

Приложение

1. Введение

Целью данной работы является изучить следующий материал:

- 1) Что такое сегмент, сегмент данных, кода и стека. Сегментные регистры. Что такое смещение. Команды LEA и OFFSET.
- 2) Что такое стек и как он используется. Команды PUSH и POP.
- 3) Что такое подпрограмма, использование подпрограмм в ассемблере. Команды CALL и RET.
- 4) Понятие дальнего перехода и дальнего вызова подпрограммы.
- 5) Команда INT.
- 6) Функции 21h прерывания для ввода и вывода символа и строки.
- 7) Организация циклов в Ассемблере. Команда LOOP.
- 8) Алгоритмы ввода двоичного, десятичного и шестнадцатеричного чисел.
- 9) Алгоритмы вывода двоичного, десятичного и шестнадцатеричного чисел.

2. Постановка задачи

2.1. Текст задания

Для разработанной в лабораторной работе 1 программы, написать подпрограммы для ввода и вывода десятичных чисел.

Если $a * (c + b) * (d \wedge 2) = (a - d) * (b + c)$ то

Если $a > b \wedge 2$ то

Результат = $c \wedge 2 / (d - c) - d \wedge 2$

Иначе

Если $a < c + d$ то

Результат = $d \wedge 2 + (b \text{ OR } c)$

Иначе

Результат = $a + (b \text{ AND } c)$

2.2. Условие задания

В главной программе необходимо ввести числа при помощи этих подпрограмм, выполнить расчеты согласно варианту задания по лабораторной работе 1 и вывести результат на экран.

3. Программная реализация

3.1. В главной программе вводятся числа при помощи подпрограмм. Программа разбита при помощи меток на несколько логических частей, каждая из которых выполняет определенную ветку условия.

3.2. Результат можно видеть в окне консоли

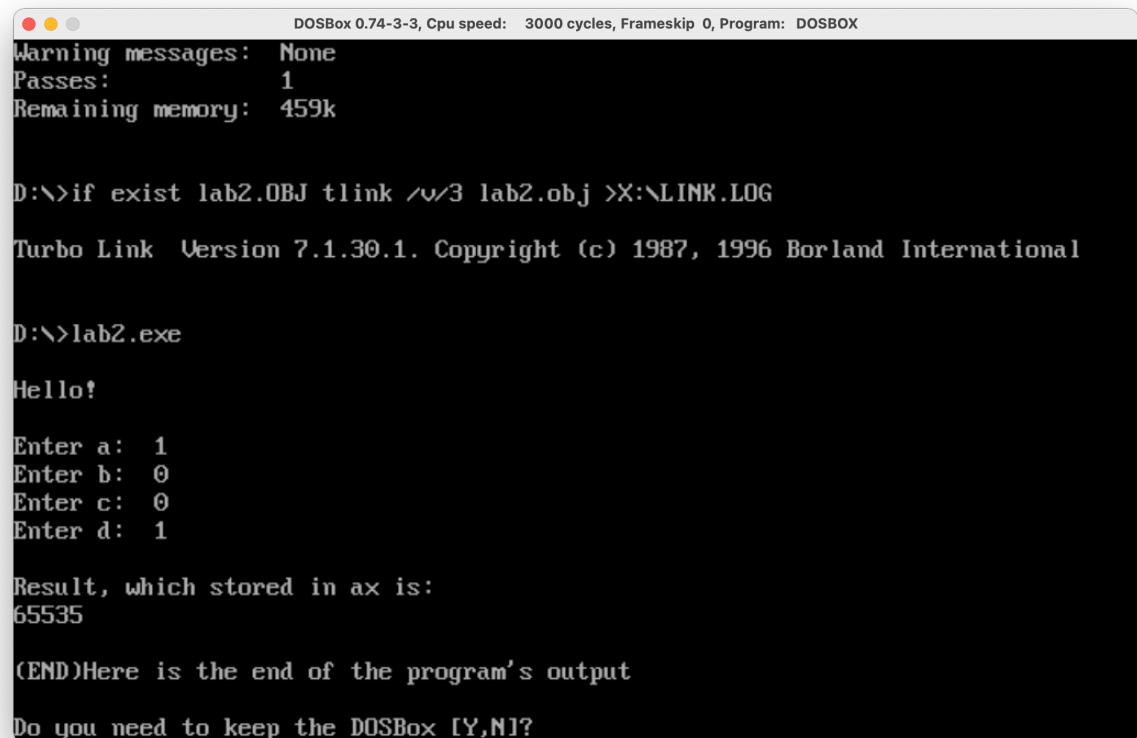
3.3. Примеры:

3.3.1 Ветка №1

Тест для ветки

« Результат = $c^2 / (d - c) - d^2$ »

a = 1 b = 0 c = 0 d = 1



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /o/3 lab2.obj >X:\LINK.LOG
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe
Hello!
Enter a: 1
Enter b: 0
Enter c: 0
Enter d: 1

Result, which stored in ax is:
65535

(END)Here is the end of the program's output
Do you need to keep the DOSBox [Y,N]?
```

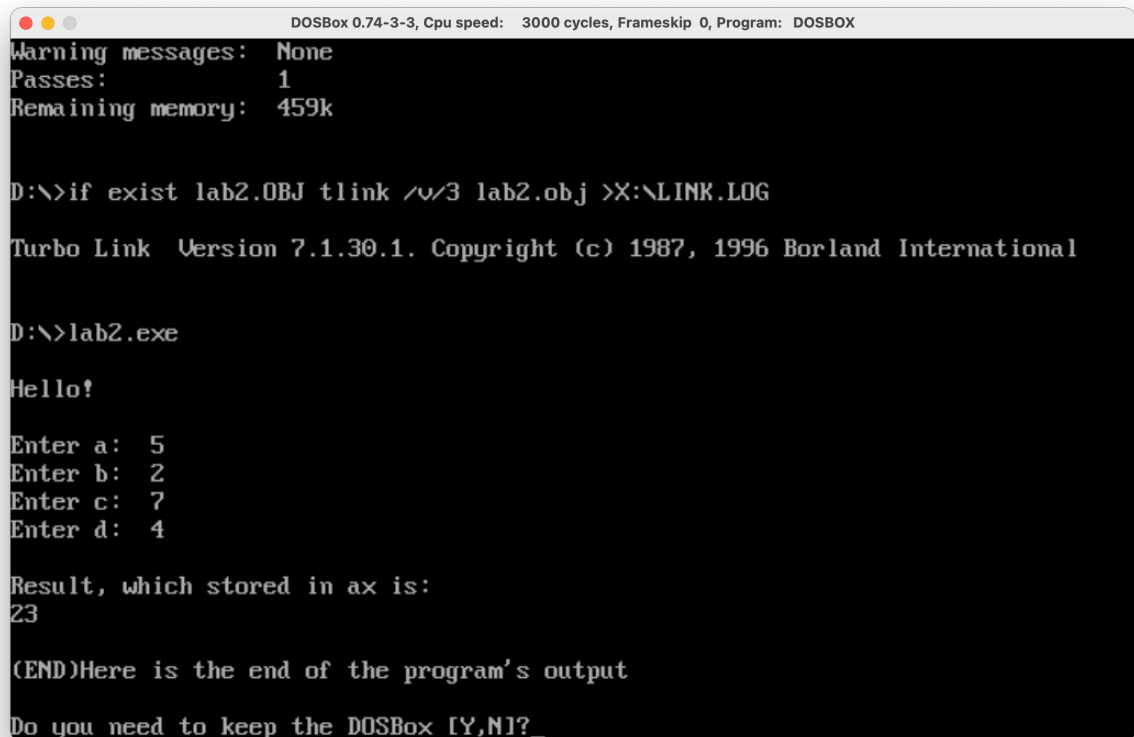
Ответ: -1_{10}

3.3.2 Ветка №2

Тест для ветки

« Результат = $d^2 + (b \text{ OR } c)$ »

a = 5 b = 2 c = 7 d = 4



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /v/3 lab2.obj >X:\LINK.LOG

Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe

Hello!

Enter a: 5
Enter b: 2
Enter c: 7
Enter d: 4

Result, which stored in ax is:
23

(END)Here is the end of the program's output

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

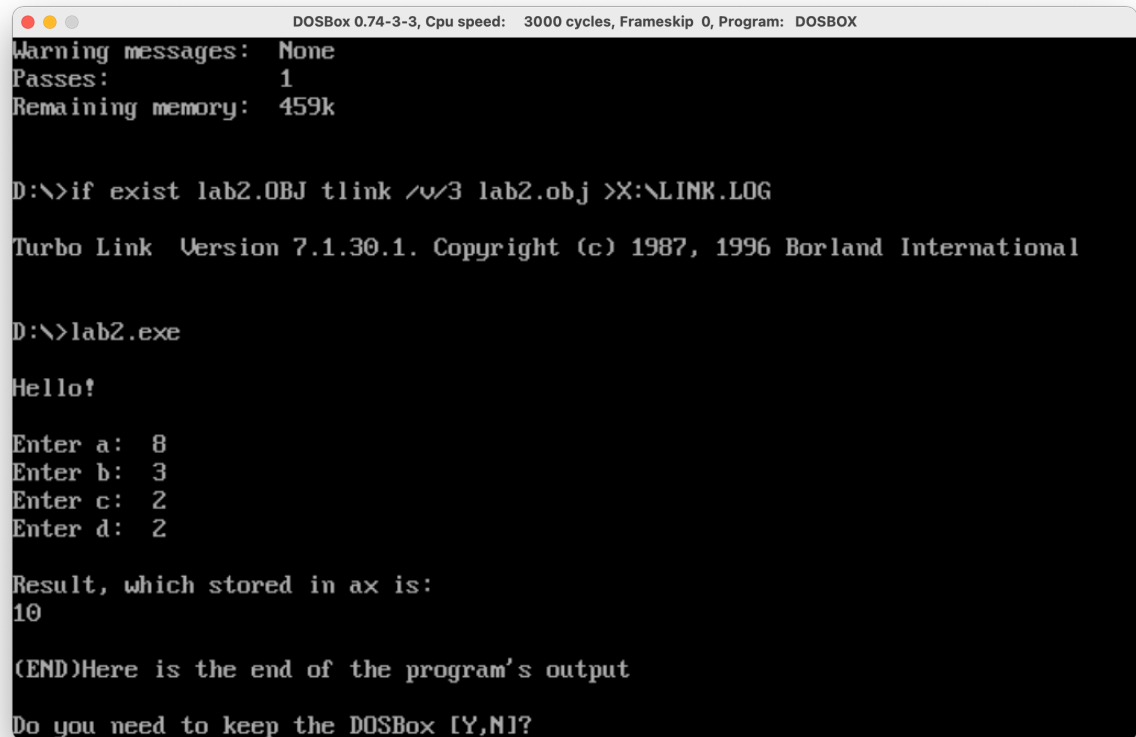
Ответ: 23₁₀

3.3.3 Ветка №3

Тест для ветки

« Результат = $a + (b \text{ AND } c)$ »

$a = 8$ $b = 3$ $c = 2$ $d = 2$



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /v/3 lab2.obj >X:\LINK.LOG

Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe

Hello?

Enter a: 8
Enter b: 3
Enter c: 2
Enter d: 2

Result, which stored in ax is:
10

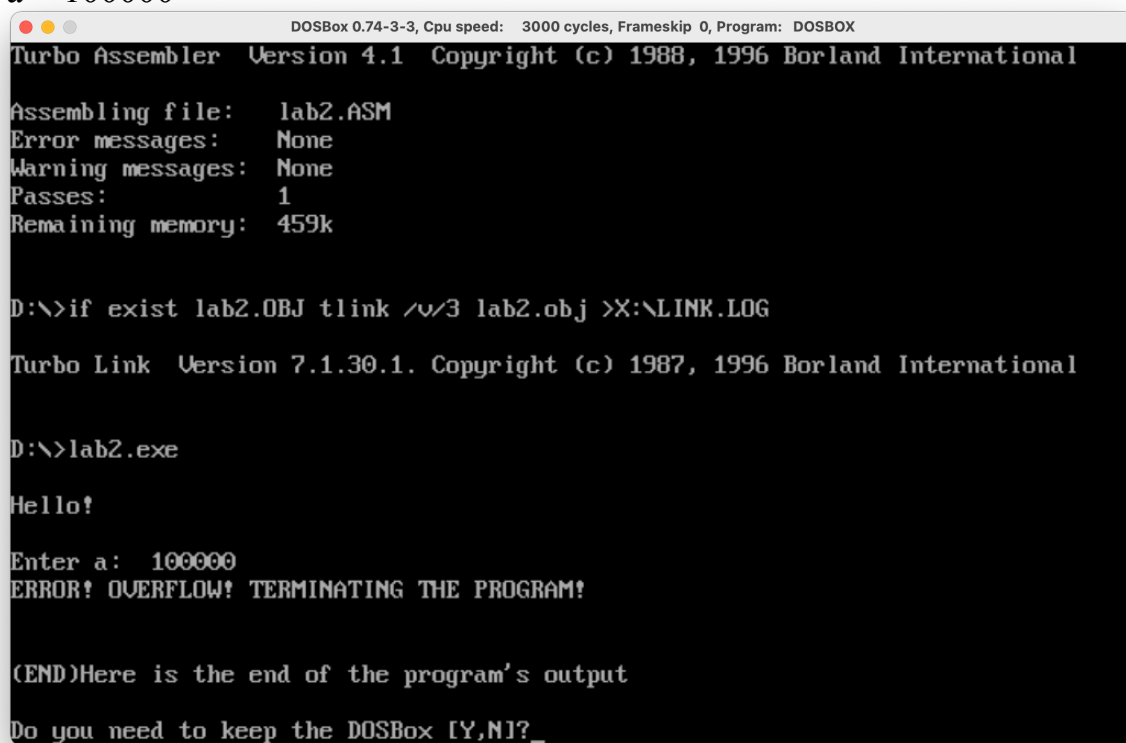
(END)Here is the end of the program's output

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

Ответ: 10_{10}

3.3.4 Пример обработки переполнения на вводе №1 (140 строка)

a = 100000



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file: lab2.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /v/3 lab2.obj >X:\LINK.LOG

Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe

Hello!

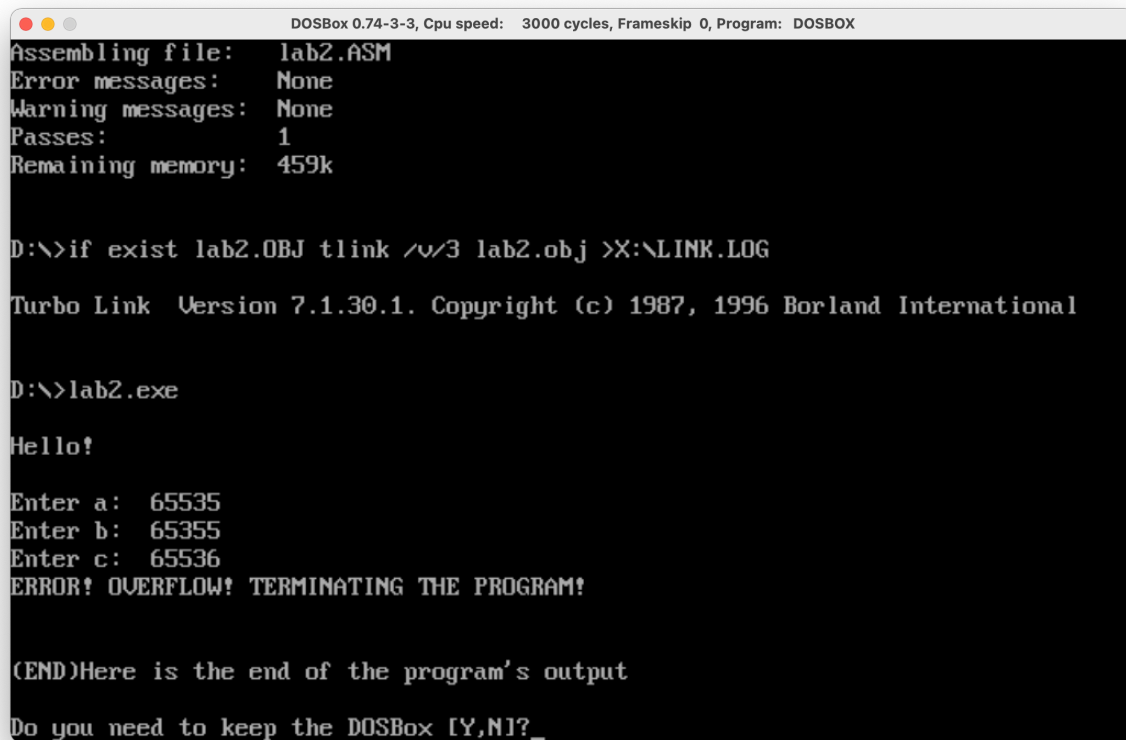
Enter a: 100000
ERROR! OVERFLOW! TERMINATING THE PROGRAM!

(END)Here is the end of the program's output

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


3.3.5 Пример обработки переполнения на вводе №2 (146 строка)

a = 65535 b = 65355 c=65536



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Assembling file: lab2.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /u/3 lab2.obj >X:\LINK.LOG

Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe

Hello!

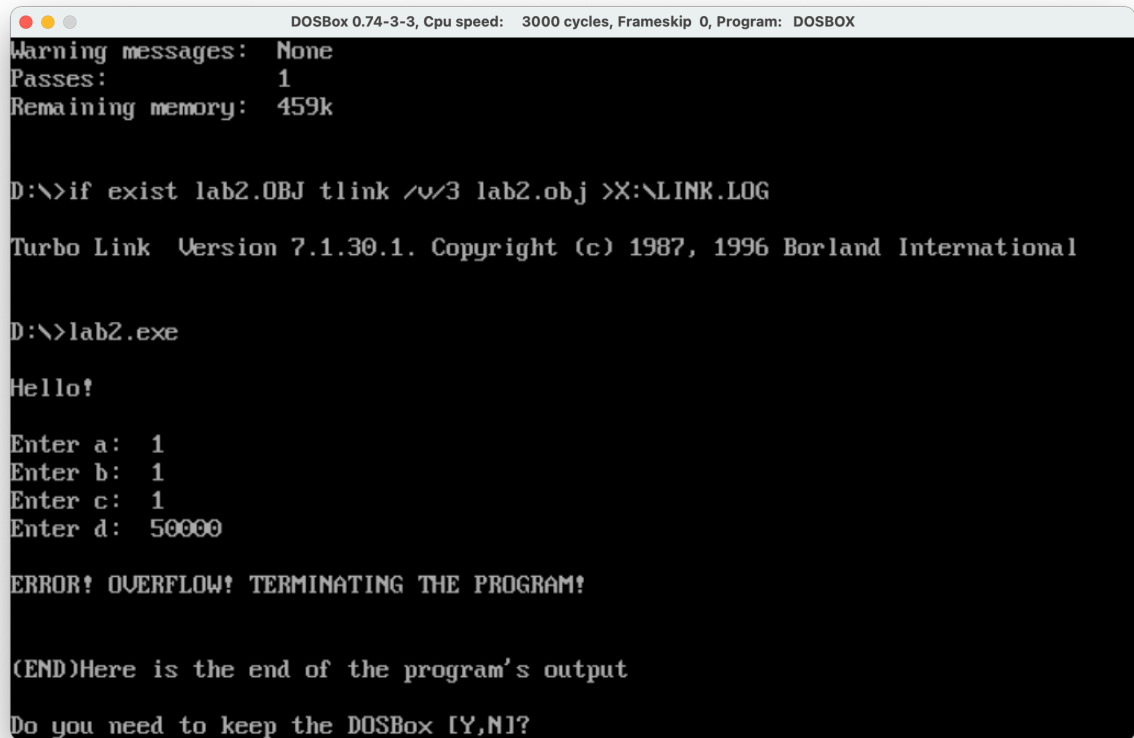
Enter a: 65535
Enter b: 65355
Enter c: 65536
ERROR! OVERFLOW! TERMINATING THE PROGRAM!

(END)Here is the end of the program's output

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

3.3.6 Пример обработки переполнения во время выполнения №1 (231 строка)

a = 1 b = 1 c = 1 d = 50000



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /v/3 lab2.obj >X:\LINK.LOG

Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe

Hello!

Enter a: 1
Enter b: 1
Enter c: 1
Enter d: 50000

ERROR! OVERFLOW! TERMINATING THE PROGRAM!

(END)Here is the end of the program's output

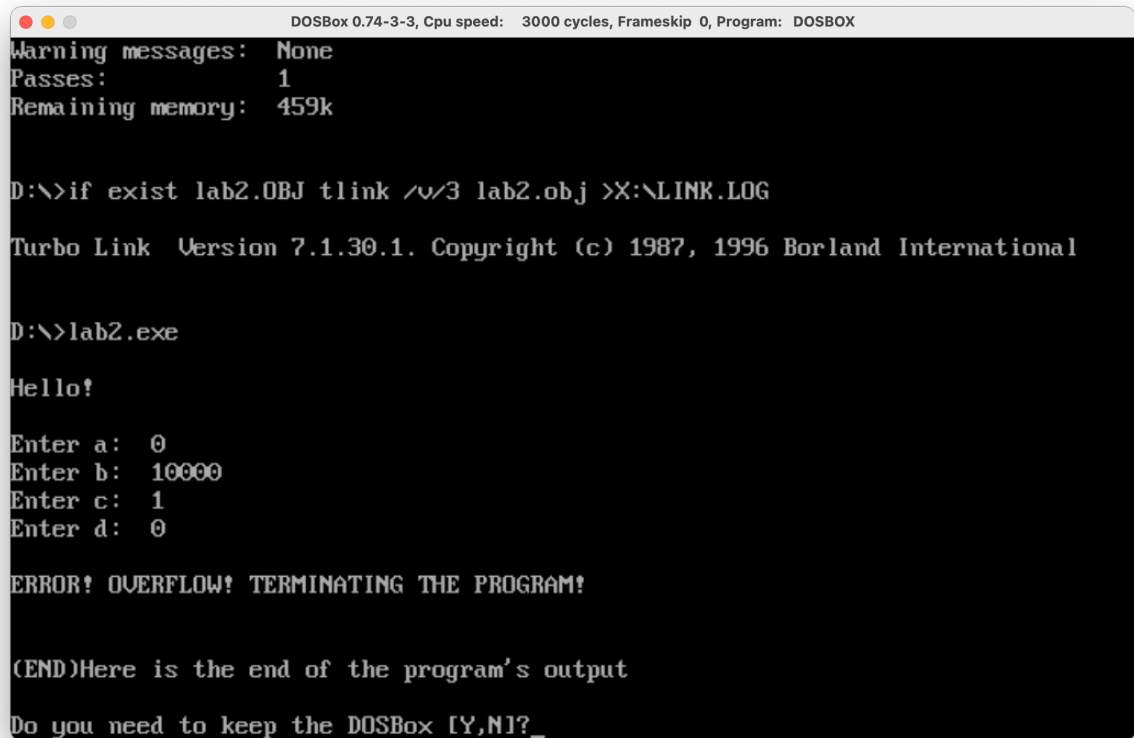
Do you need to keep the DOSBox [Y,N]?
```

Переполнение происходит в условии:

“ Если $a * (c + b) * (d ^ 2) = (a - d) * (b + c)$ ”

3.3.7 Пример обработки переполнения во время выполнения №2 (267 строка)

a = 0 b = 10000 c = 1 d = 0



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /v/3 lab2.obj >X:\LINK.LOG
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe
Hello!
Enter a: 0
Enter b: 10000
Enter c: 1
Enter d: 0

ERROR! OVERFLOW! TERMINATING THE PROGRAM!

(END)Here is the end of the program's output
Do you need to keep the DOSBox [Y,N]?_
```

Переполнение происходит в условии:

“ Если $a > b^2$ ”

Для остальных условий, а именно:

“Результат = $c^2 / (d - c) - d^2$ ”

“Если $a < c + d$ ”

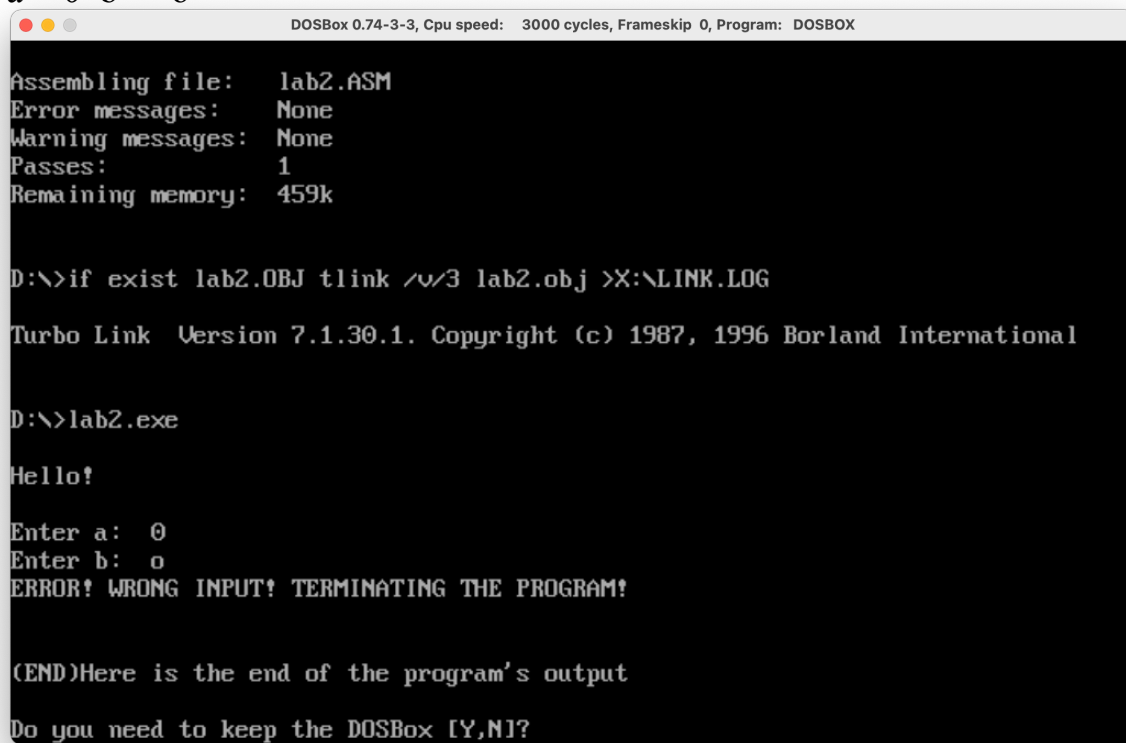
“Результат = $d^2 + (b \text{ OR } c)$ ”

“Результат = $a + (b \text{ AND } c)$ ”

не имеет смысла проверки переполнения, так как оно было бы вызвано раньше.

3.3.8 Пример обработки неправильного ввода №1 (132 строка)

a = 0 b = 'o'



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

Assembling file: lab2.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /v/3 lab2.obj >X:\LINK.LOG

Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe

Hello!

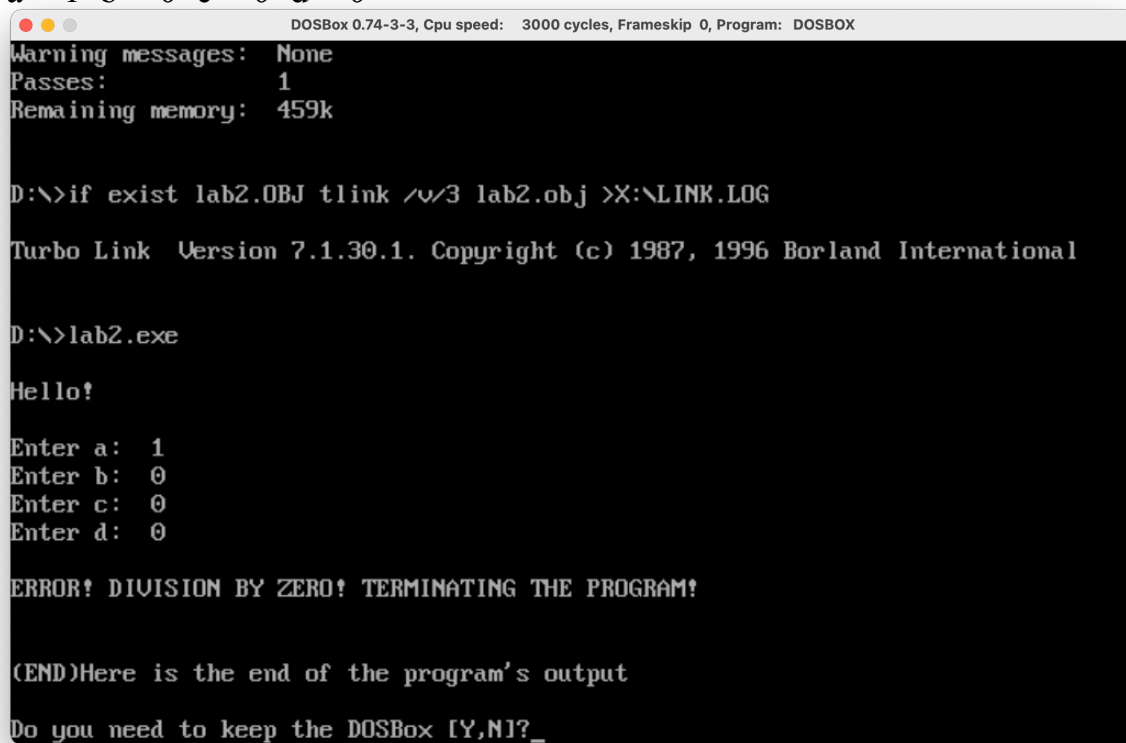
Enter a: 0
Enter b: o
ERROR! WRONG INPUT! TERMINATING THE PROGRAM!

(END)Here is the end of the program's output

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

3.3.9 Пример обработки деления на ноль №1 (284 строка)

a = 1 b = 0 c = 0 d = 0



The screenshot shows a DOSBox window with the following text:

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /u/3 lab2.obj >X:\LINK.LOG
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe
Hello!

Enter a: 1
Enter b: 0
Enter c: 0
Enter d: 0

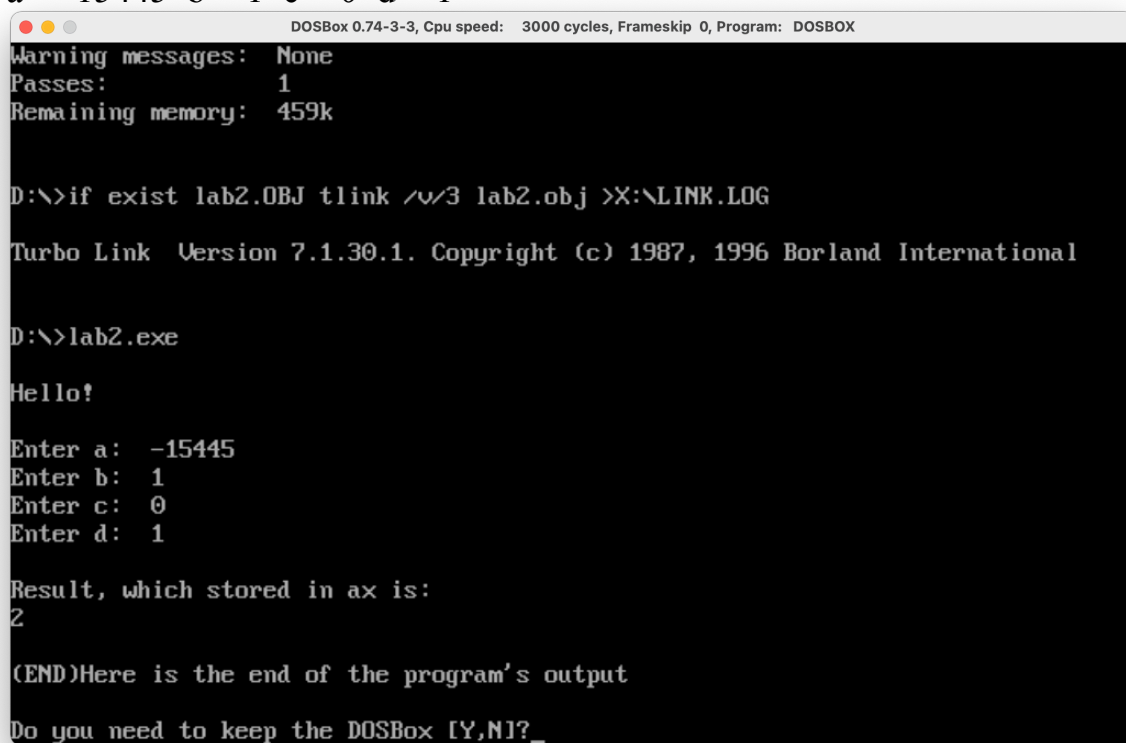
ERROR! DIVISION BY ZERO! TERMINATING THE PROGRAM!

(END)Here is the end of the program's output

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

3.3.10 Пример обработки отрицательных чисел №1

a = -15445 b = 1 c = 0 d = 1



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Warning messages: None
Passes: 1
Remaining memory: 459k

D:\>if exist lab2.OBJ tlink /v/3 lab2.obj >X:\LINK.LOG
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>lab2.exe
Hello!
Enter a: -15445
Enter b: 1
Enter c: 0
Enter d: 1

Result, which stored in ax is:
2

(END)Here is the end of the program's output
Do you need to keep the DOSBox [Y,N]?_
```

Ответ: 2₁₀

4. Выводы

На практике было изучено и опробовано, в соответствии с поставленной задачей:

Что такое сегмент, сегменты данных, кода и стека.

Сегментные регистры.

Что такое смещение.

Команды LEA и OFFSET.

Что такое стек и как он используется.

Команды PUSH и POP.

Что такое подпрограмма, использование подпрограмм в ассемблере.

Команды CALL и RET.

Понятие дальнего перехода и дальнего вызова подпрограммы.

Команда INT.

Функции 21h прерывания для ввода и вывода символа и строки.

Организация циклов в Ассемблере.

Команда LOOP.

Алгоритмы ввода двоичного, десятичного и шестнадцатеричного чисел.

Алгоритмы вывода двоичного, десятичного и шестнадцатеричного чисел.

Литература

1. Юров В.И. – «Assembler. Учебник для вузов. 2-ое издание, 2003 год».
2. Юров В.И. – «Assembler. Практикум. 2-ое издание, 2006 год».
3. Калашников О.А. – «Ассемблер - это просто. 2-ое издание, 2011 год».

Приложение

.model small

.stack 256

.data

greetings db 10,'Hello!',13,10,10,'\$'

space db 10,'\$'

wrongInput db 10,'ERROR! WRONG INPUT! TERMINATING THE
PROGRAM!',13,10,'\$'

overflow db 10,'ERROR! OVERFLOW! TERMINATING THE
PROGRAM!',13,10,'\$'

divisionByZero db 10,'ERROR! DIVISION BY ZERO! TERMINATING THE
PROGRAM!',13,10,'\$'

enterA db 'Enter a: \$'

enterB db 'Enter b: \$'

enterC db 'Enter c: \$'

enterDq db 'Enter d: \$'

output db 10,'Result, which stored in ax is: ',13,10,'\$'

a dw 0

b dw 0

c dw 0

d dw 0

aEntered dw 0

bEntered dw 0

cEntered dw 0

dEntered dw 0

minus dw 0

zero dw 0

counter dw 0

.code

.386

main:

mov ax, @data

mov ds, ax

greetingsFunc:

lea dx, greetings

mov ah, 09h ; display string, which is stored in dx

int 21h

enterAFunc:

lea dx, enterA

mov ah, 09h ; display string, which is stored in dx

int 21h

jmp handleFirstNum

enterBFunc:

lea dx, enterB

mov ah, 09h ; display string, which is stored in dx
int 21h

jmp handleFirstNum

enterCFunc:

lea dx, enterC

mov ah, 09h ; display string, which is stored in dx
int 21h

jmp handleFirstNum

enterDFunc:

lea dx, enterDq

mov ah, 09h ; display string, which is stored in dx
int 21h

jmp handleFirstNum

handleFirstNum:

mov ah, 01h

int 21h ; al - first symbol

cmp al, 2dh ; if '-'

je minusLabel

cmp al, 30h ; if 0 - handle next var

je firstIsZero

cmp al, 30h ; 30h - 0 in decimal

jnl wrongInputLabel

cmp al, 39h ; 39h - 9 in decimal

jg wrongInputLabel

sub al, 30h ; now al - first number (30h - 'ascii 0')

mov ah, 0 ; extend to word

mov bx, 10 ; if more than one digit (input 12 == $(1 * 10) + 2$)

mov cx, ax ; cx - first num

jmp loopLabel

minusLabel:

mov minus, 1

jmp handleFirstNum

loopLabel:

mov ah, 01h

int 21h ; al - next symbol

cmp al, 0dh ; cmp with 'enter button'

je endLoop ; if 'enter button' pressed - return

cmp al, 30h ; 30h - 1 in decimal

jnl wrongInputLabel

cmp al, 39h ; 39h - 9 in decimal

jg wrongInputLabel

sub al, 30h ; al - next number

cbw ; extend to word

xchg ax, cx ; now cx - next number, ax - previous

mul bx ; ax * 10

jo overflowLabel

add cx, ax ; $cx = (ax * 10) + cx$

jo overflowLabel

cmp cx, 0

je overflowLabel

jmp loopLabel; continue input

firstIsZero:

mov zero, 1

mov cx, 0

lea dx, space

mov ah, 09h ; display string, which is stored in dx

int 21h

jmp endLoopA

endLoop:

cmp minus, 1 ; 1 means that number with minus

jne endLoopA

neg cx

endLoopA:

cmp aEntered, 0

jne endLoopB

mov a, cx

mov aEntered, 1

mov minus, 0

```
mov zero, 0  
jmp enterBFunc
```

endLoopB:

```
cmp bEntered, 0  
jne endLoopC  
mov b, cx  
mov bEntered, 1  
mov minus, 0  
mov zero, 0  
jmp enterCFunc
```

endLoopC:

```
cmp cEntered, 0  
jne endLoopD  
mov c, cx  
mov cEntered, 1  
mov minus, 0  
mov zero, 0  
jmp enterDFunc
```

endLoopD:

```
cmp dEntered, 0  
jne leftPart
```

```
mov d, cx
mov dEntered, 1
mov minus, 0
mov zero, 0
```

```
;mov ax, 17
;jmp loophere
```

testValues:

```
mov ax, a
mov bx, b
mov cx, c
mov dx, d
```

leftPart:

```
mov ax, a

mov bx, b
add bx, c
jo overflowLabel

mul bx    ; ax = ax * bx
jo overflowLabel

mov bx, ax
```



```
mov ax, d
mul ax    ; ax = ax ^ 2
jo overflowLabel
```

```
mul bx    ; ax = ax * bx
jo overflowLabel
```

```
mov bx, ax ; bx = a * (c + b) * (d ^ 2) leftPart
mov ax, 0
```

rightPart:

```
mov ax, a
sub ax, d
jo overflowLabel
```

```
mov cx, b
add cx, c
jo overflowLabel
```

```
mul cx    ; ax = ax * cx
jo overflowLabel
```

```
mov cx, ax ; cx = (a - d) * (b + c) rightPart
mov ax, 0
```

compareLeftAndRight:

cmp cx, bx

jne secondIf

firstIf: ; $a > b^2$

mov bx, a

mov ax, b

mul ax ; $ax = a^2$

jo overflowLabel

cmp bx, ax ; $bx = a, ax = b^2$

jle secondIf

firstSolution: ; $c^2 / (d - c) - d^2$

mov ax, c

mul ax ; $ax = c^2$

jo overflowLabel

mov bx, d

sub bx, c ; $bx = d - c$

jo overflowLabel

cmp bx, 0

je divisionByZeroLabel ; if denominator = 0 => exit (exception)

div bx ; ax = ax / bx

jo overflowLabel

mov bx, ax ; $bx = c^2 / (d - c)$

mov ax, d

mul ax

jo overflowLabel

sub bx, ax ; $bx = c^2 / (d - c) - d^2$

jo overflowLabel

mov ax, bx

jmp putResultOnStack

secondIf: ; $a < c + d$

mov ax, a

mov bx, c

add bx, d

jo overflowLabel

cmp ax, bx

jg thirdSolution

secondSolution: ; $d^2 + (b \text{ OR } c)$

mov ax, b

mov bx, c

or ax, bx ; puts 'or' result in ax

jo overflowLabel

mov bx, ax ; then in bx

mov ax, d

mul ax

jo overflowLabel

add ax, bx

jo overflowLabel

jmp putResultOnStack

thirdSolution: ; $a + (b \text{ AND } c)$

mov ax, b

mov bx, c

and ax, bx

jo overflowLabel

mov bx, ax

mov ax, a

add ax, bx

jo overflowLabel

jmp putResultOnStack

putResultOnStack:

mov cx, 10

mov dx, 0

div cx

add dl, '0'

push dx

inc counter

cmp ax, 0

jnz putResultOnStack

startOutput:

lea dx, output

mov ah, 09h ; display string, which is stored in dx

int 21h

popResultFromStack:

pop dx

mov ah, 02h ; 02h is the function number of output char

int 21h

dec counter

cmp counter, 0

jne popResultFromStack

jmp exit

overflowLabel:

lea dx, overFlow

mov ah, 09h ; display string, which is stored in dx

int 21h

jmp exit

wrongInputLabel:

lea dx, wrongInput

mov ah, 09h ; display string, which is stored in dx

int 21h

jmp exit

divisionByZeroLabel:

lea dx, divisionByZero

mov ah, 09h ; display string, which is stored in dx

int 21h

jmp exit

exit:

lea dx, space

mov ah, 09h ; display string, which is stored in dx

int 21h

mov ax, 4c00h

mov al, 0

int 21h

end main