Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №3

Работа со знаковыми числами

Выполнил: Студент: гр. 053502

Юрьев Вадим Андреевич

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) Знаковые и беззнаковые числа.

2) Что такое дополнительный код. Представление чисел в дополнительном коде.

3) Команда NEG.

4)  Команды IMUL и IDIV и их отличие от команд MUL и DIV.

5)  Команды CBW и CWD и их использование.

6)  Алгоритмы ввода и вывода знаковых десятичных чисел.

1. Постановка задачи

1) Написать процедуру для вывода знакового числа размерностью слово из регистра AX на экран  
 -  на входе число в регистре AX  
 -  на выходе это число на экране  
 Все задействованные регистры должны быть сохранены в процедуре  
  
2) Написать процедуру для ввода знакового числа с клавиатуры в регистр AX  
  - на входе пользовательский ввод, минус может быть только как первый символ  
  - на выходе введенное число в регистре AX  
Все задействованные регистры должны быть сохранены в процедуре (кроме AX)  
   также должна быть проверка вводимого числа на правильность:  
   -  нельзя вводить не цифры, минус может быть только первым символом,  
   -  проверка на допустимые значения (-32768...32767)  
 Дополнительно можно реализовать обработку клавиши “бэкспэйс” (удалить один  
  последний введенный символ) и клавиши “ескэйп” (удалить все число).

Модифицировать программы второй лабораторной работы таким образом, чтобы можно было вводить и выводить знаковые числа.

1. Программная реализация

3.1. Расчеты происходят по заданным ниже выражениям:

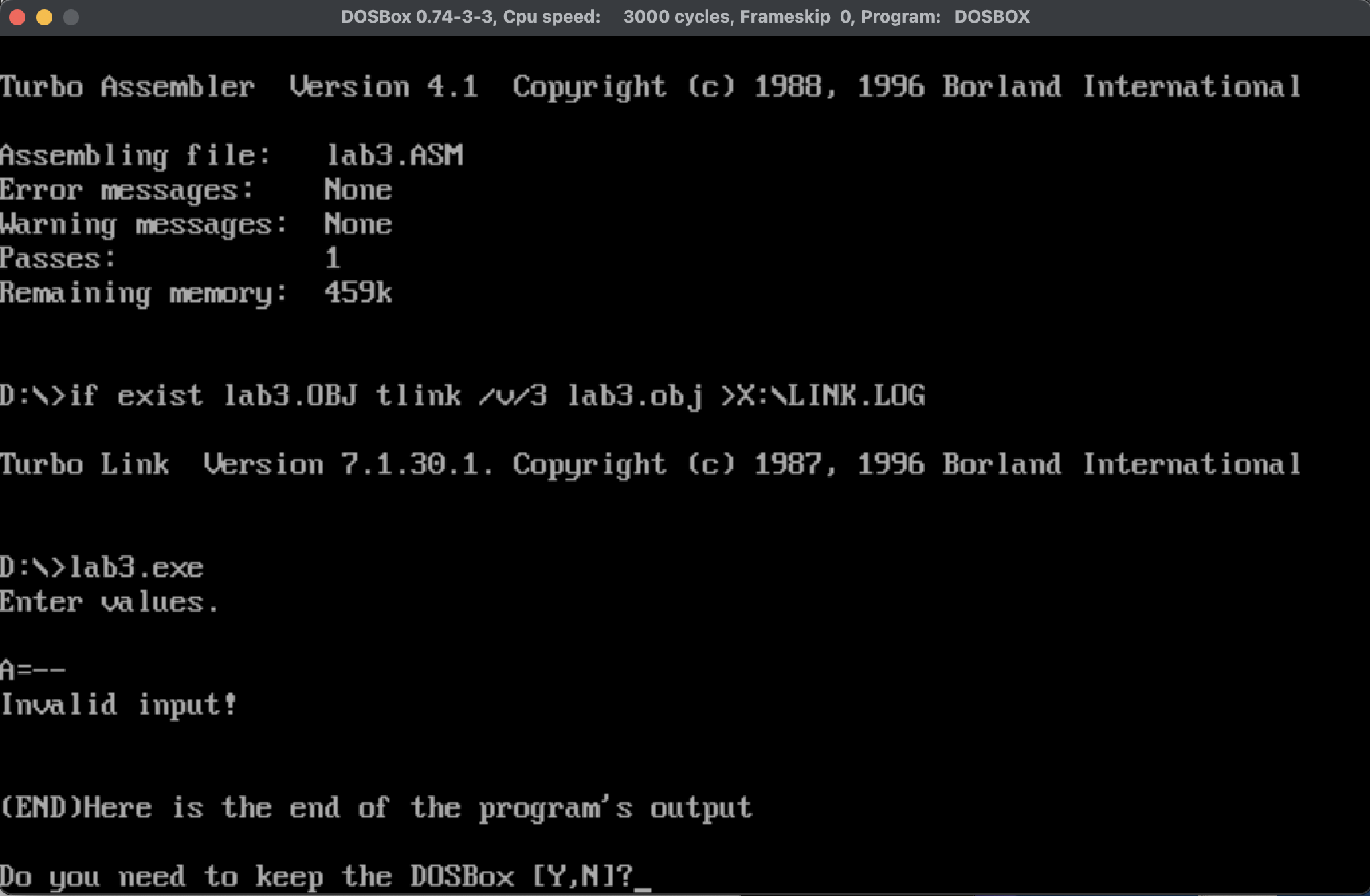
Если a = c ^ 2 то  
      Результат = с / (a \* b) + d  
  Иначе  
      Если  c - b <> a ^ 2 + b ^ 3 то  
       Результат = a AND (b - c)  
    Иначе  
       Результат = a/b/c+(b^2+c^3)/d

* 1. Значения переменных вводятся с устройства ввода. Программа разбита при помощи меток на несколько логических частей, каждая из которых выполняет определенную ветку условия.
  2. Результат можно видеть в выводе в консоли.
  3. Примеры:

3.4.1 Отладка программы №1

Проверка правильности ввода.

Ввод: A=--

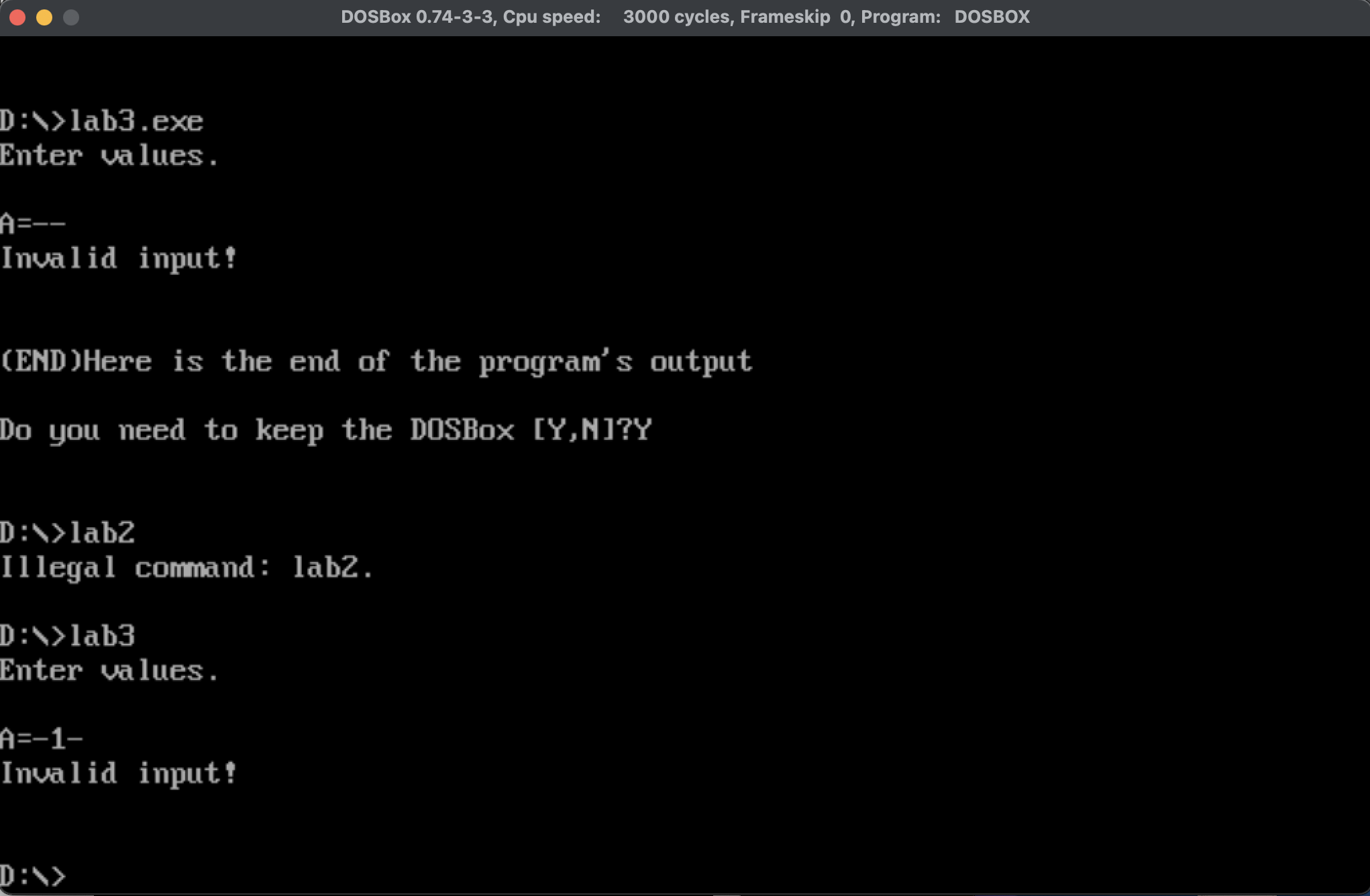


Результат: вывод об ошибке – некорректный ввод.

3.4.2 Отладка программы №2

Проверка правильности ввода.

Ввод: А=-1-

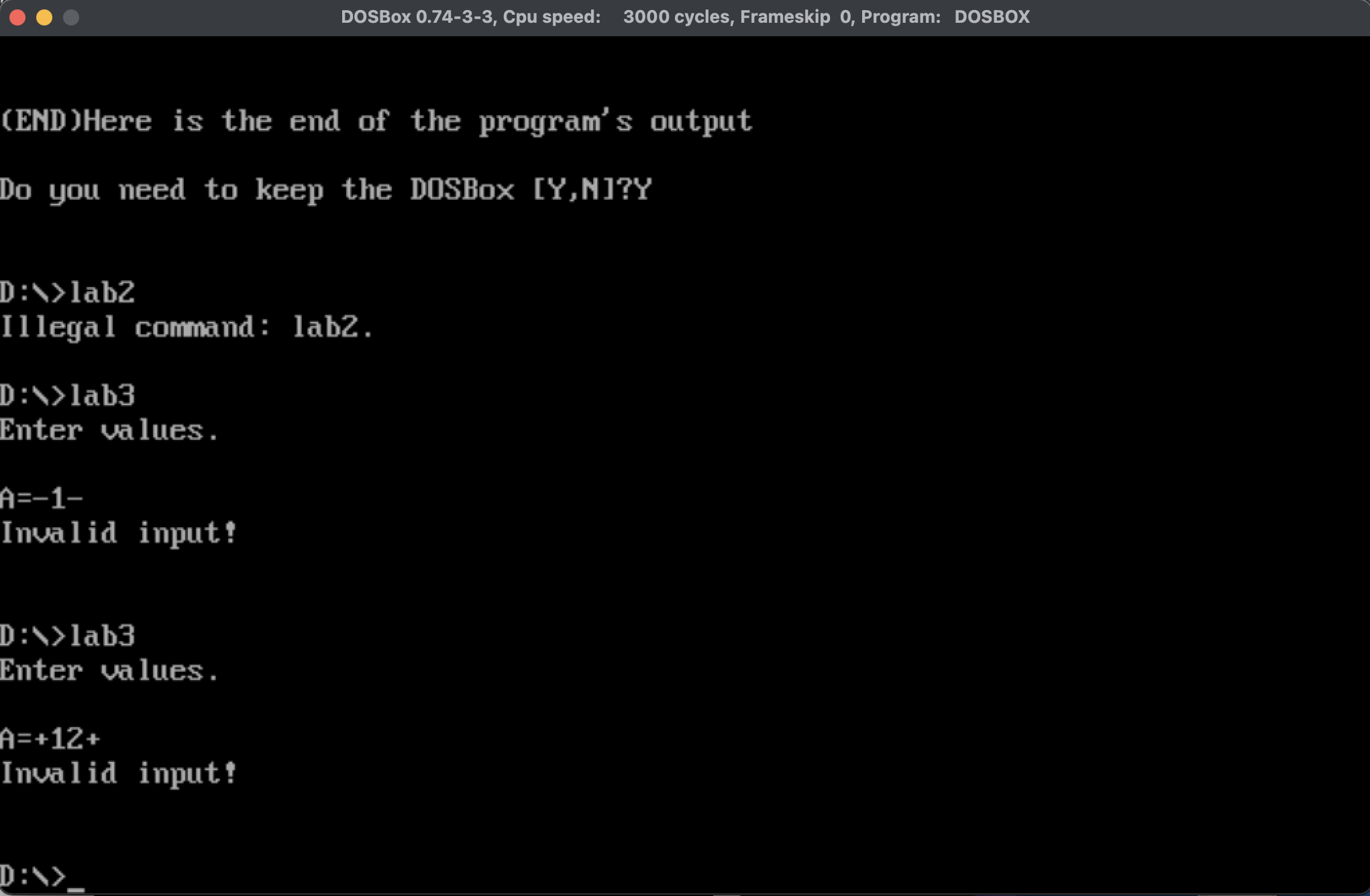


Результат: вывод об ошибке – некорректный ввод.

3.4.3 Отладка программы №3

Проверка правильности ввода.

Ввод: А=+12+



Результат: вывод об ошибке – некорректный ввод.

3.4.4 Отладка программы №4

Проверка правильности ввода.

a = 1 b = 32768

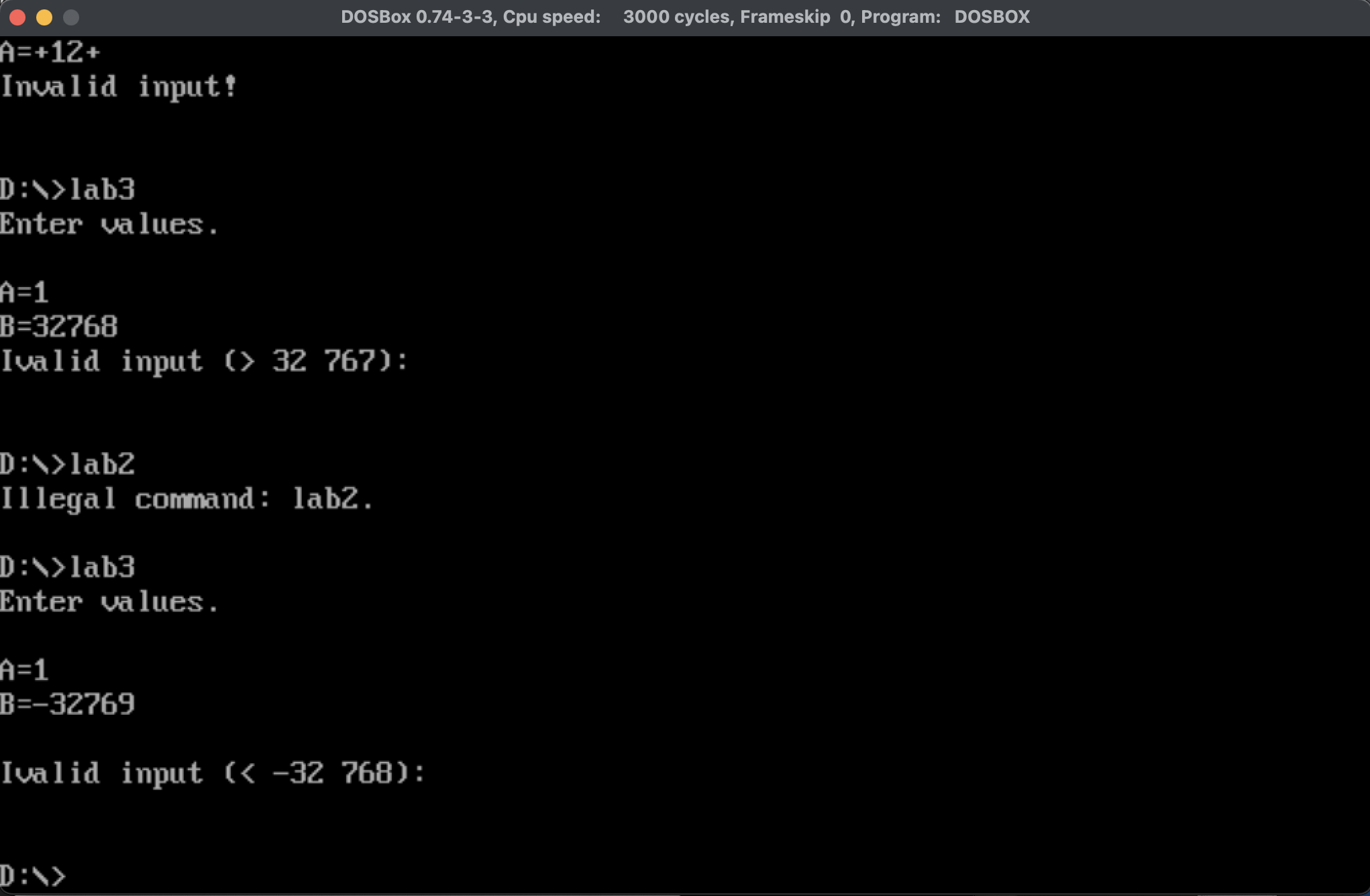


Результат: вывод об ошибке – некорректный ввод, число больше 32767.

3.4.5 Отладка программы №5

Проверка правильности ввода.

a = 1 b = -32769

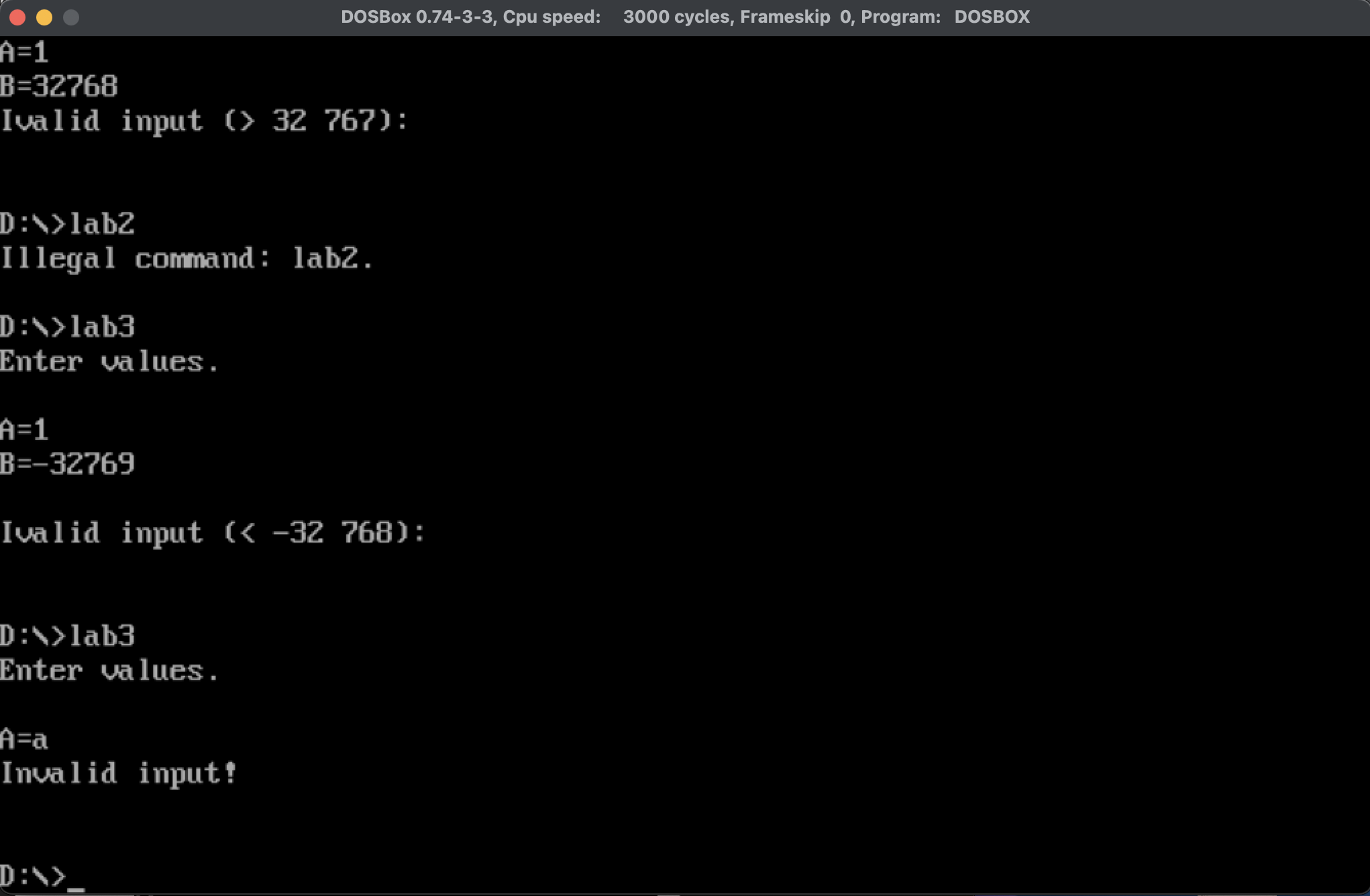


Результат: вывод об ошибке – некорректный ввод, число меньше -32768.

3.4.6 Отладка программы №6

Проверка правильности ввода.

a = a



Результат: вывод об ошибке – некорректный ввод.

3.4.7 Отладка ветки №1

Проверка на переполнение для выражения

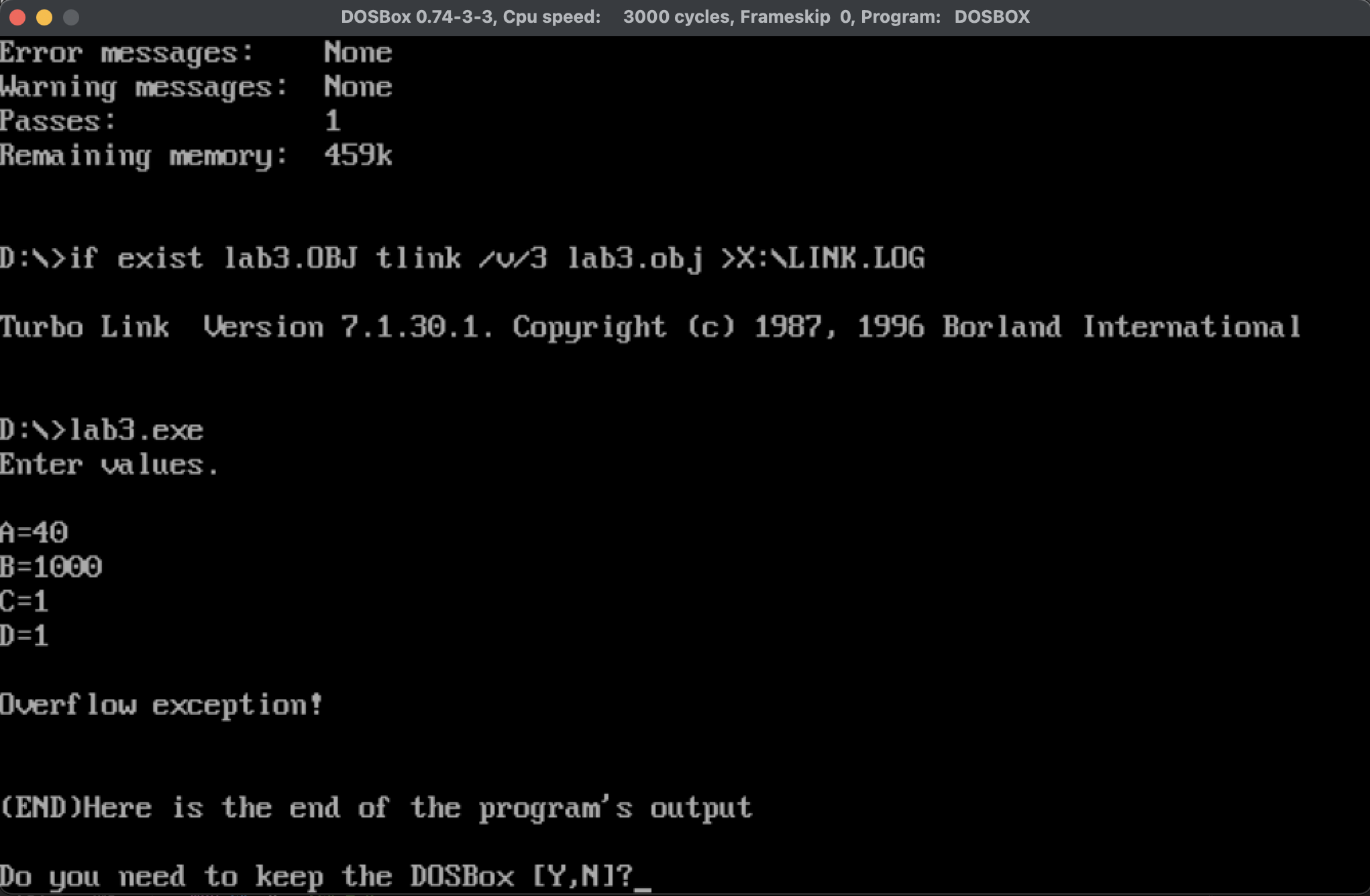
«Если a = c ^ 2 то

Результат = c / (a \* b) + d»

Для выражения “c / (a \* b) + d” переполнения в процессе выполнения быть не может, так как c / (a \* b) будет равно нулю (a \* b > c).

Проверка для выражения “a = c ^ 2”

a = 40 b = 1000 c = 1 d = 1



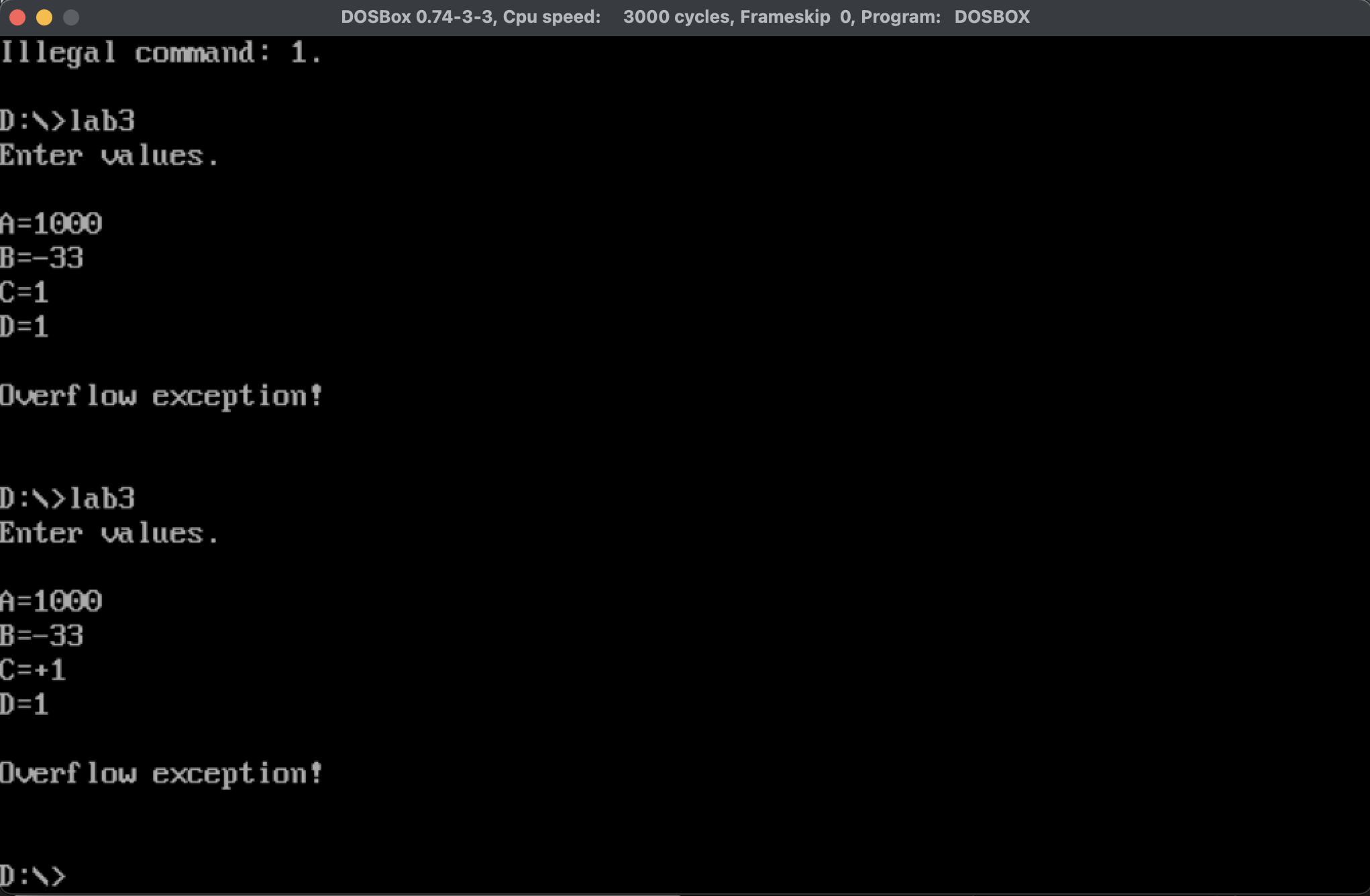
Результат: вывод об ошибке – переполнение.

3.4.8 Отладка ветки №1

Проверка на переполнение для выражения

«a\*b»

a = 1000 b = -33 c = +1 d = 1



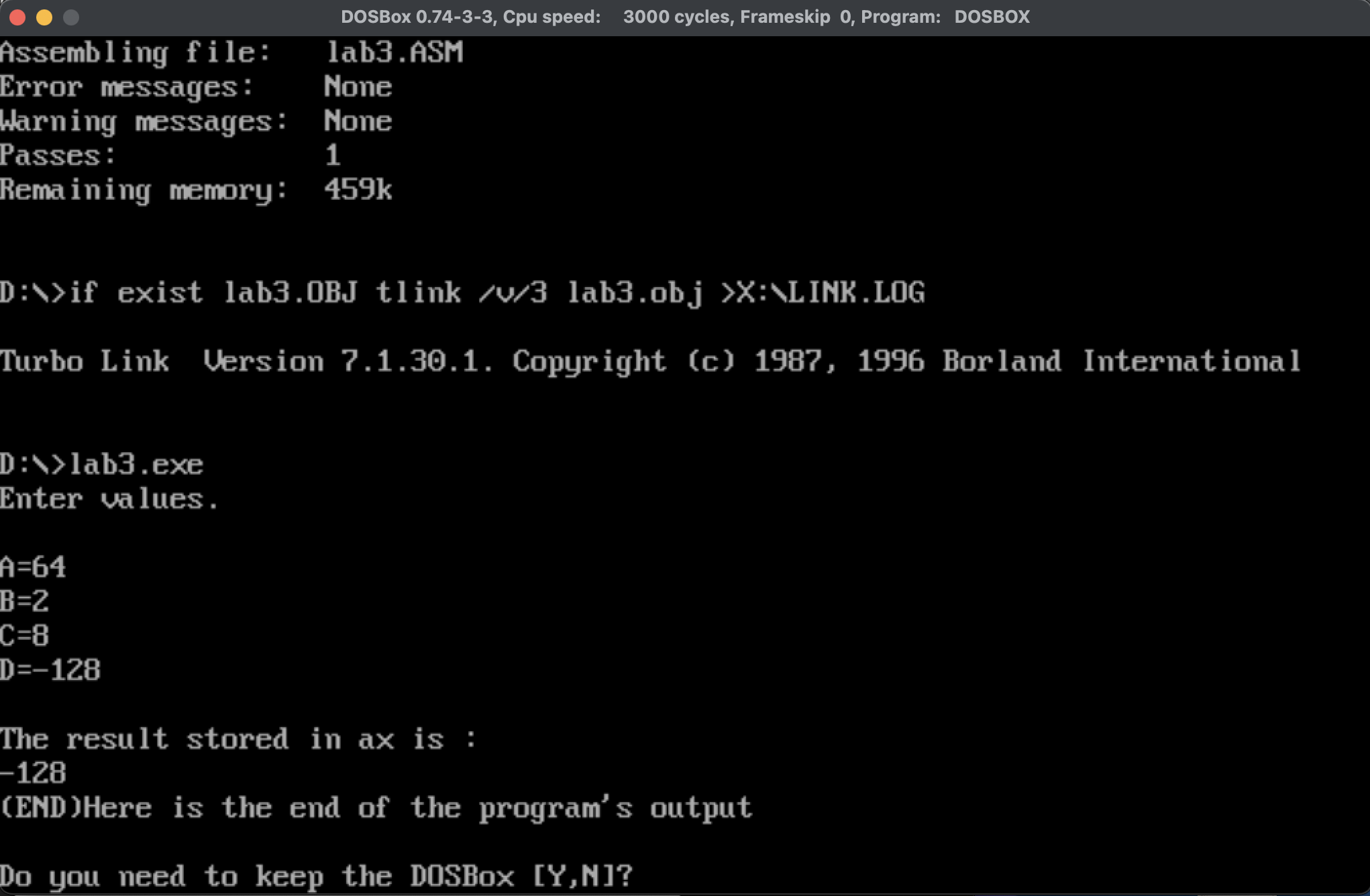
Результат: вывод об ошибке – переполнение.

3.4.9 Отладка ветки №1

Тест для ветки

« Результат =c/(a\*b)+d»

a = 64 b = 2 c = 8 d = -128



Ответ:-128

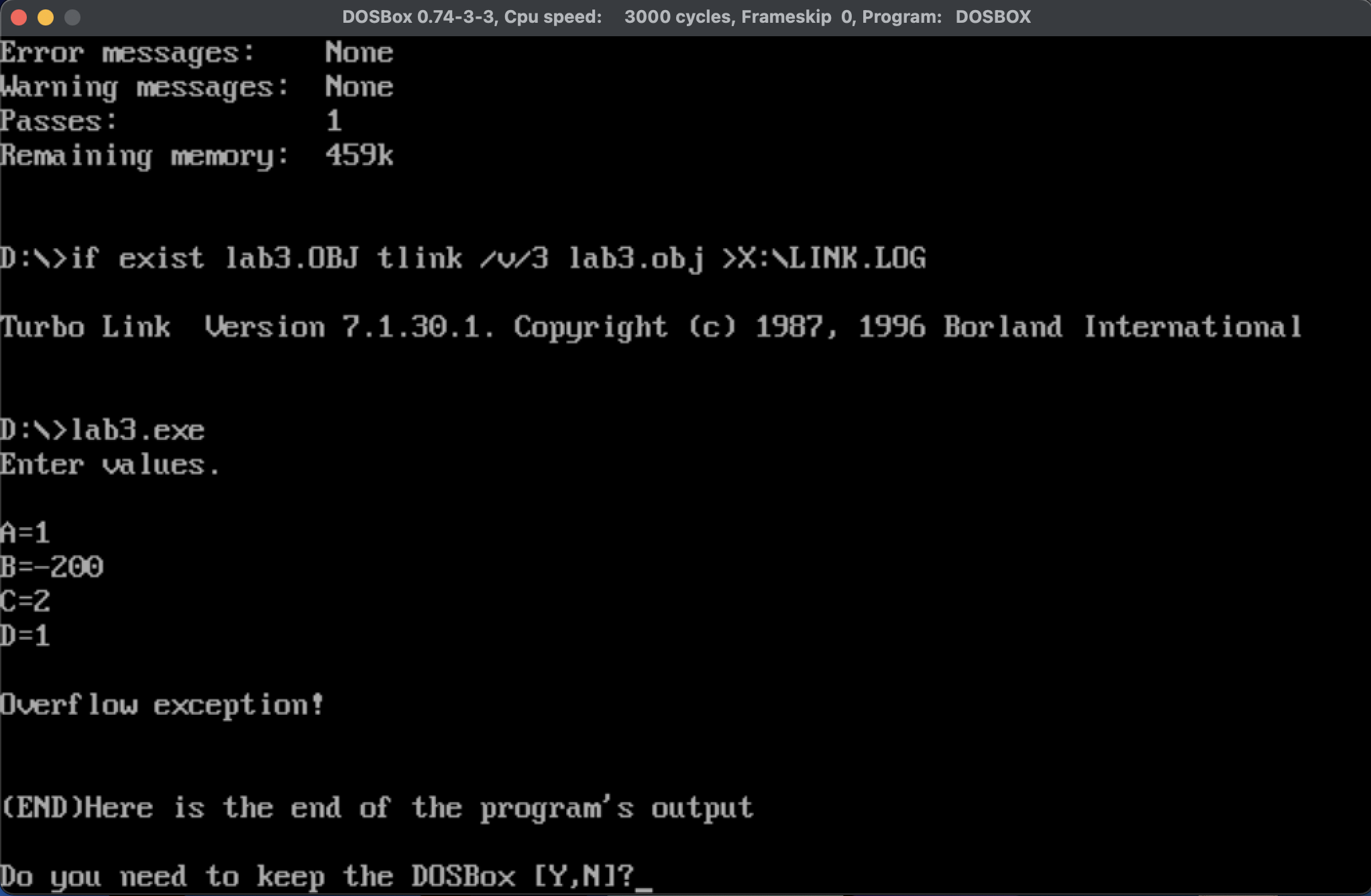
3.4.9 Отладка ветки №2

Если  c - b <> a ^ 2 + b ^ 3 то  
       Результат = a AND (b - c)

Проверка на переполнение для выражения

«b^2»

a = 1 b = -200 c = 2 d = 1



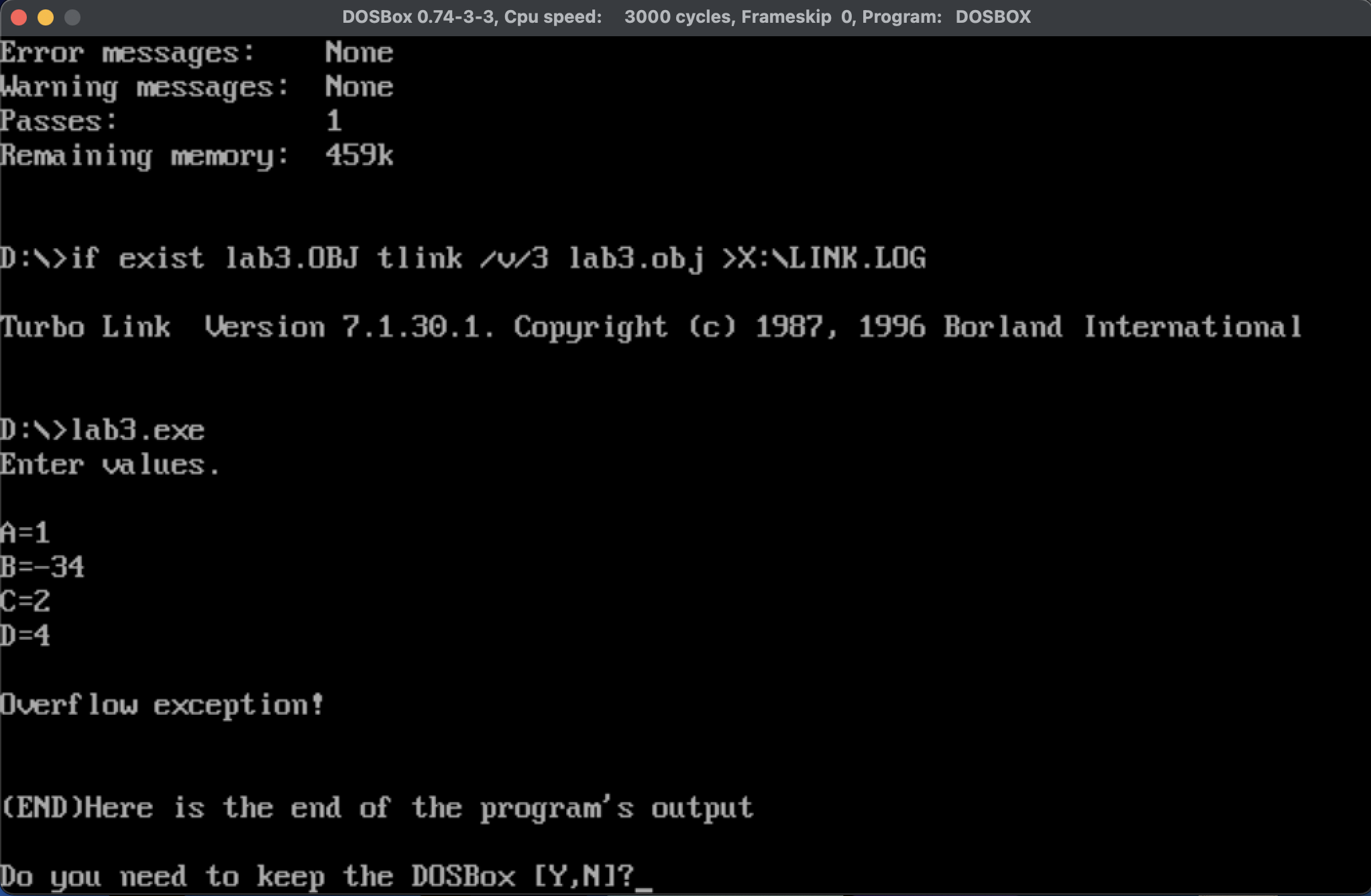
Результат:вывод об ошибке – переполнение.

3.4.10 Отладка ветки №2

Проверка на переполнение для выражения

«b^3»

a = 1 b = -34 c = 2 d = 4

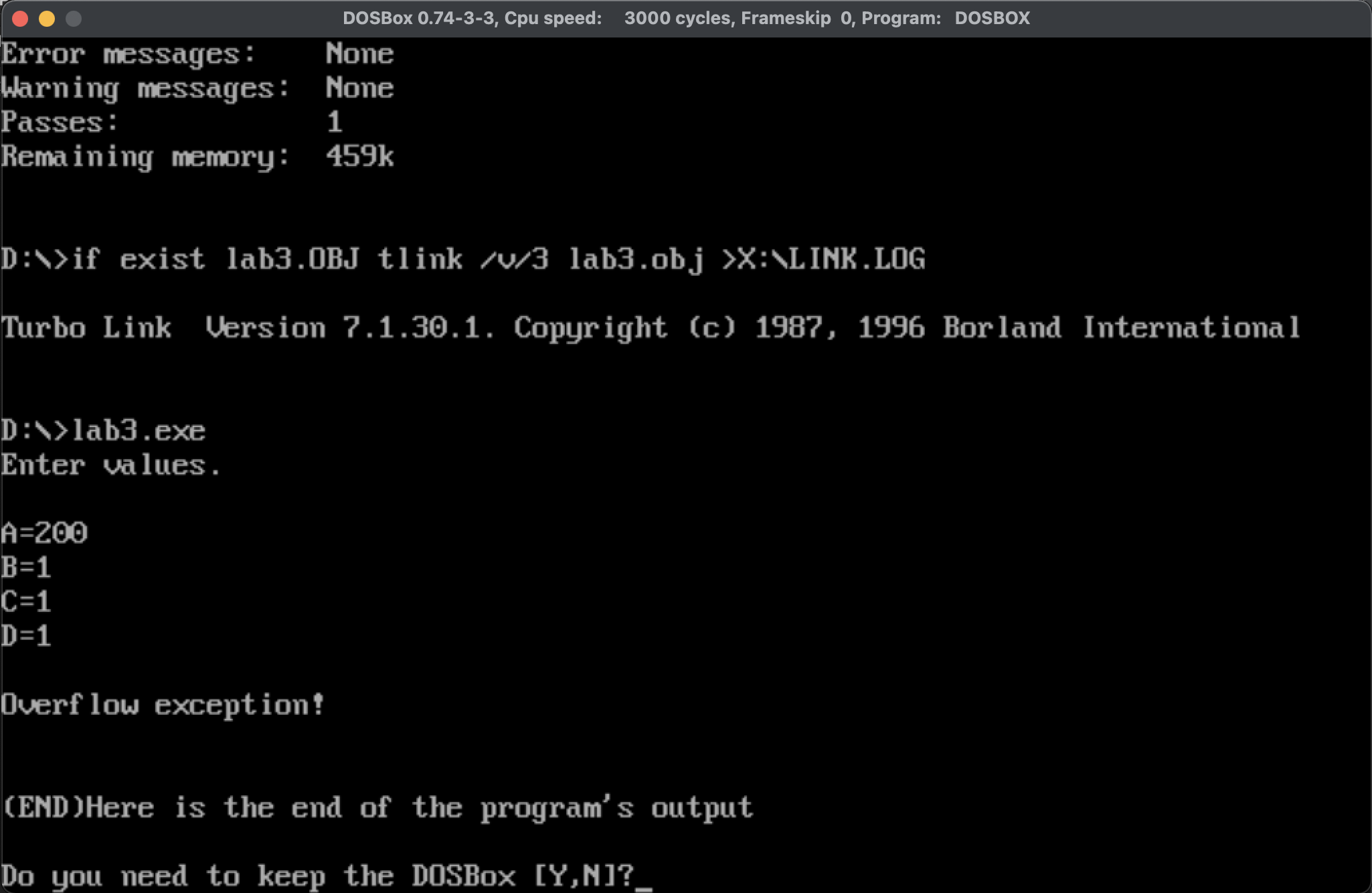


Результат: вывод об ошибке – переполнение.

3.4.11 Отладка программы №2

Проверка на переполнение для выражения «a^2»

a = 200 b = 1 c = 1 d = 1



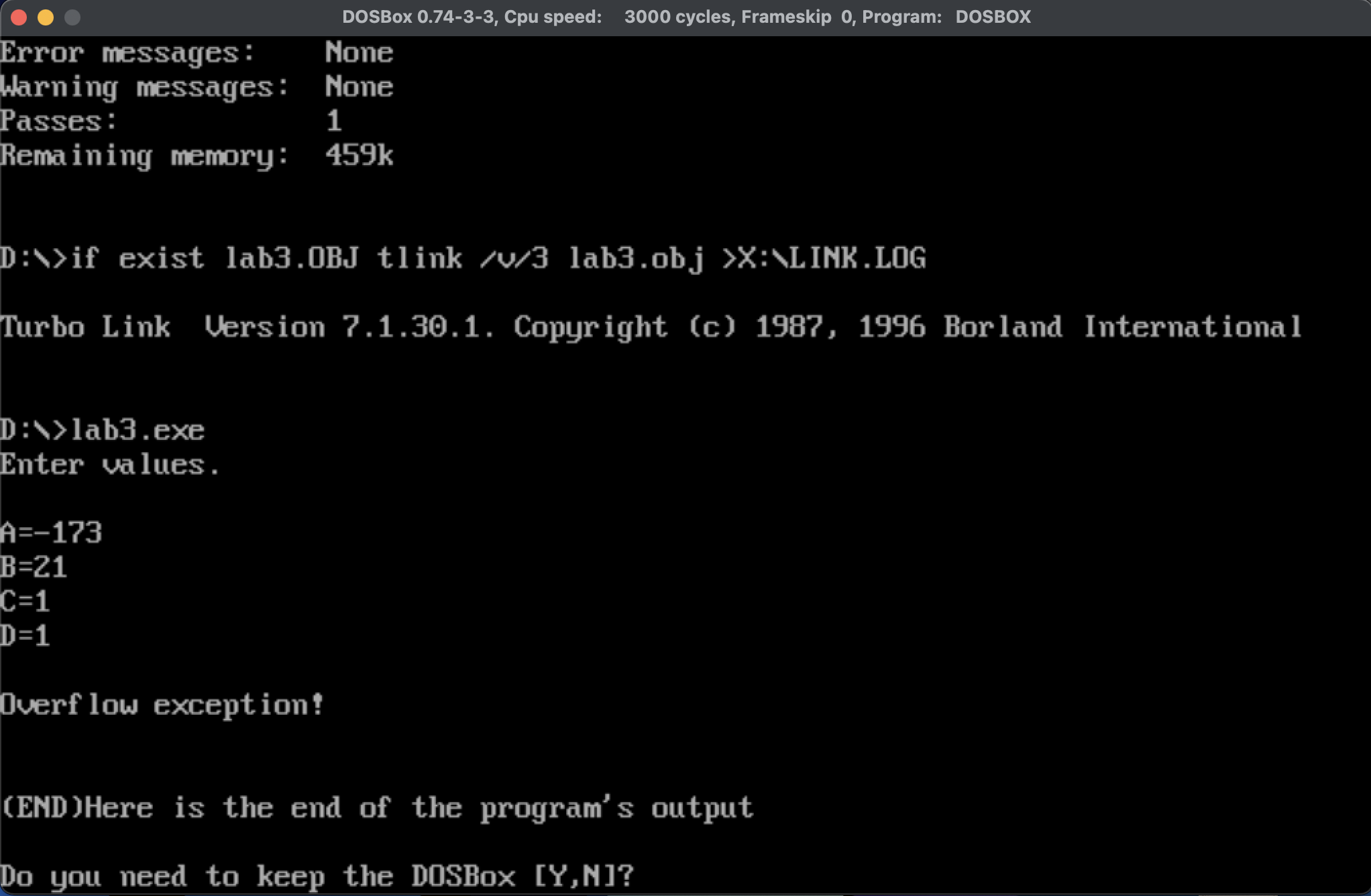
Результат: вывод об ошибке – переполнение.

3.4.12 Отладка ветки №2

Проверка на переполнение для выражения

«a^2 + b^3»

a = -173 b = 21 c = 1 d = 1



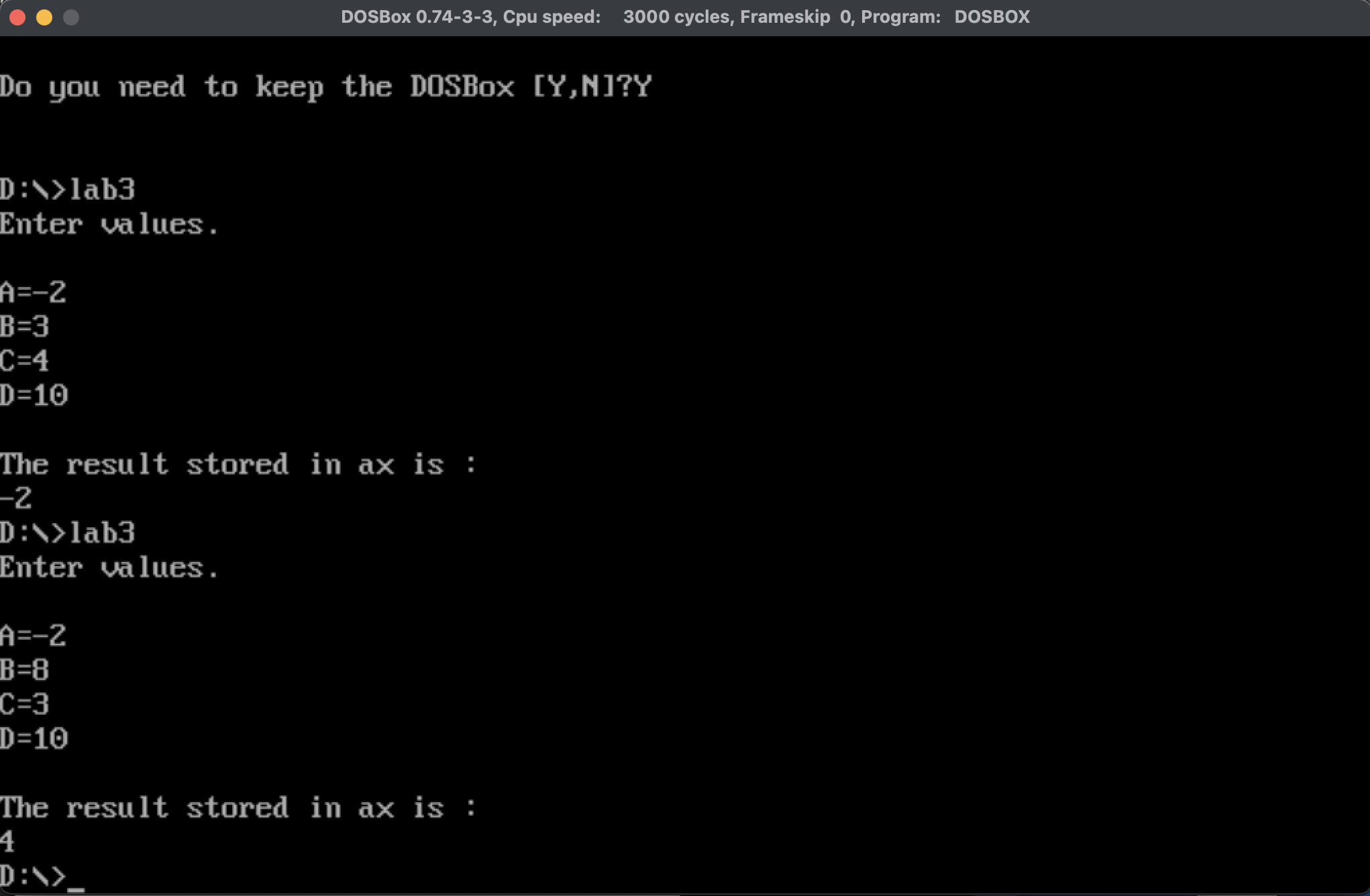
Результат: вывод об ошибке – переполнение.

3.4.13 Отладка ветки №2

Тест для ветки

« Результат = a AND (b-c) »

a = -2 b = 8 c = 3 d = 10



Ответ: 4

3.4.14 Отладка ветки №3

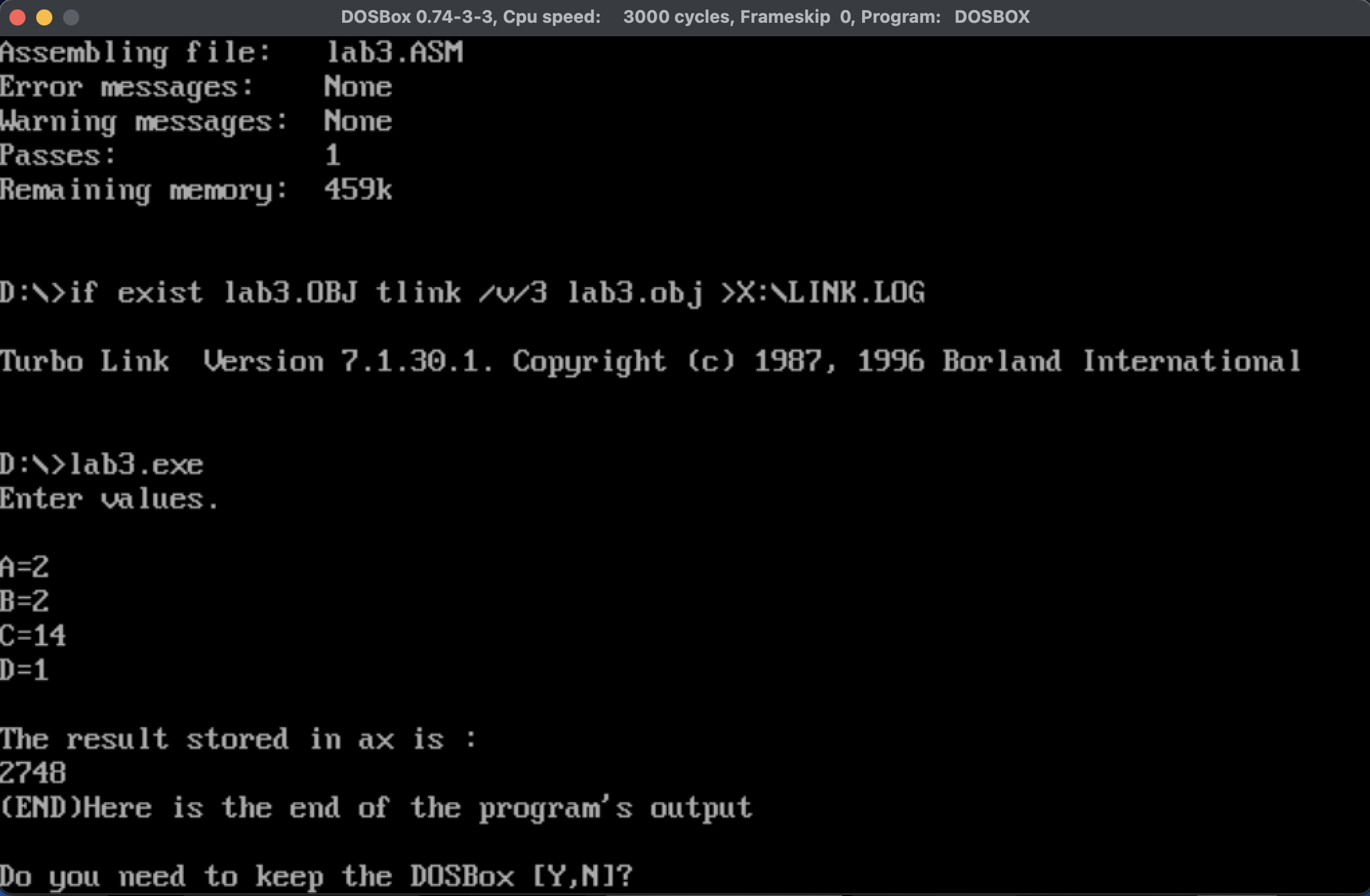
В ветке 3 нельзя подобрать значения для переполнения выражения

“b^2+c^3”

Тест для ветки

« Результат = a/b/c+(b^2+c^3)/d »

a = 2 b = 2 c = 14 d = 1



Ответ: 2748

1. Выводы

На практике было изучено и опробовано, в соответствии с поставленной задачей: знаковые и беззнаковые числа, дополнительный код, представление чисел в дополнительном коде, команда NEG, команды IMUL и IDIV и их отличие от команд MUL и DIV, команды CBW и CWD и их использование, алгоритмы ввода и вывода знаковых десятичных 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

startMessage db 'Enter values.',10,10,'$'

aMessage db 'A=$'

bMessage db 'B=$'

cMessage db 'C=$'

dMessage db 'D=$'

overflowException db 10,'Overflow exception!',10,10,'$'

divisionByZeroException db 10,'idivision by zero exception!',10,10,'$'

invalidInputException db 10,'Invalid input!',10,10,'$'

resultMessage db 10,'The result stored in ax is :',10,'$'

moreThenException db 10,'Ivalid input (> 32 767):',10,10,'$'

lessThenException db 10,'Ivalid input (< -32 768):',10,10,'$'

a dw 0

b dw 0

c dw 0

d dw 0

count dw 0

maxValue dw 32767

enteredA dw 0

enteredB dw 0

enteredC dw 0

enteredD dw 0

negative dw 0

positive dw 0

.code

.386

input proc

mov ah, 01h

int 21h

cmp al, 2dh ;сравнение с минусом

je isNegative

cmp al, 2bh ;сравнение с плюсом

je isPositive

cmp al, 30h

jl invalidInput

cmp al, 39h

jg invalidInput

sub al, 30h

mov ah, 0

mov bx, 10

mov cx, ax

jmp inputLoop

input endp

start:

mov ax, @data

mov ds, ax

startFunc:

lea dx, startMessage

mov ah, 09h

int 21h

aEnter:

lea dx, aMessage

mov ah, 09h

int 21h

jmp input

bEnter:

lea dx, bMessage

mov ah, 09h

int 21h

jmp input

cEnter:

lea dx, cMessage

mov ah, 09h

int 21h

jmp input

dEnter:

lea dx, dMessage

mov ah, 09h

int 21h

jmp input

isNegative:

cmp negative, 1

je invalidInput

mov negative, 1

jmp input

isPositive:

cmp positive, 1

je invalidInput

mov positive, 1

jmp input

inputLoop:

mov ah, 01h

int 21h

cmp al, 0dh

je endInput

cmp al, 30h

jl invalidInput

cmp al, 39h

jg invalidInput

sub al, 30h

cbw

xchg ax, cx

imul bx

jo overflow

add cx, ax

jc overflow

cmp negative, 1

je inputLoop

cmp cx, maxValue

jo moreThen

cmp cx, 0

je overflow

jmp inputLoop

endInput:

cmp negative, 1

jne endAInput

cmp cx, maxValue

jo lessThen

neg cx

endAInput:

cmp enteredA, 0

jne endBInput

mov a, cx

mov enteredA, 1

mov negative, 0

mov positive, 0

jmp bEnter

endBInput:

cmp enteredB, 0

jne endCInput

mov b, cx

mov enteredB, 1

mov negative, 0

mov positive, 0

jmp cEnter

endCInput:

cmp enteredC, 0

jne endDInput

mov c, cx

mov enteredC, 1

mov negative, 0

mov positive, 0

jmp dEnter

endDInput:

cmp enteredD, 0

jne firstIf

mov d, cx

mov enteredD, 1

mov negative, 0

mov positive, 0

firstIf: ;a = c ^ 2

mov ax, c ;ax = c

imul ax ;ax = c ^ 2

jo overflow

mov cx, ax ;cx = c ^ 2

mov ax, a ;ax = a

cmp ax, cx

jne secondIf

firstResult: ;Результат = с / (a \* b) + d

mov ax, a ;ax = a

mov bx, b ;bx = b

mov cx, d ;cx = d

imul bx ;ax = a \* b

jo overflow

mov bx, ax ;bx = a \* b

cmp bx, 0

je divisionByZero

mov ax, c ;ax = c

idiv bx ;ax = c / (a \* b)

add ax, cx ;ax = c / (a \* b) + d

jmp output

secondIf: ;c - b <> a ^ 2 + b ^ 3

mov ax, a ;ax = a

mov bx, a

imul bx ;ax = a ^ 2

jo overflow

mov cx, ax ;cx = a ^ 2

mov ax, cx

mov bx, b ;bx = b

mov ax, b ;ax = b

imul bx ;ax = b ^ 2

jo overflow

imul bx ;ax = b ^ 3

jo overflow

add ax, cx ;ax = a ^ 2 + b ^ 3

jo overflow

mov bx, c ;bx = c

mov cx, b ;cx = b

sub bx, cx ;bx = c - b

cmp ax, bx

je thirdResult

secondResult: ;Результат = a AND (b - c)

mov ax, a ;ax = a

mov bx, b ;bx = b

sub bx, c ;bx = b - c

and ax, bx ;ax = a AND (b - c)

jmp output

thirdResult: ;Результат = a/b/c+(b^2+c^3)/d

mov ax, b ;ax = b

imul ax ;ax = b ^ 2

jo overflow

mov bx, ax ;bx = b ^ 2

mov ax, c ;ax = c

mov cx, c ;cx = c

imul ax ;ax = c ^ 2

jo overflow

imul cx ;ax = c ^ 3

jo overflow

add ax, bx ;ax = c ^ 3 + b ^ 2

jo overflow

mov bx, d ;bx = d

cmp bx, 0

je divisionByZero

idiv bx ;ax = (c ^ 3 + b ^ 2) / d

mov cx, ax ;cx = (c ^ 3 + b ^ 2) / d

mov ax, a ;ax = a

mov bx, b ;bx = b

cmp bx, 0

je divisionByZero

idiv bx ;ax = a / b

mov bx, c ;bx = c

cmp bx, 0 ;0 + 2 = - 8

je divisionByZero

idiv bx ;ax = a / b / c

add ax, cx ;ax = a / b / c + (c ^ 3 + b ^ 2) / d

jo overflow

jmp output

output proc

signTest:

mov cx, ax

lea dx, resultMessage

mov ah, 09h

int 21h

mov ax, cx

cmp ax, 0

jns resultToStack

mov cx, ax

mov ah, 02h

mov dl, '-'

int 21h

mov ax, cx

neg ax

resultToStack:

mov cx, 10

mov dx, 0

idiv cx

add dl, '0'

push dx

inc count

cmp ax, 0

jnz resultToStack

resultOutput:

pop dx

mov ah, 02h

int 21h

dec count

cmp count, 0

jne resultOutput

jmp exit

output endp

overflow:

lea dx, overflowException

mov ah, 09h

int 21h

jmp exit

divisionByZero:

lea dx, divisionByZeroException

mov ah, 09h

int 21h

jmp exit

invalidInput:

lea dx, invalidInputException

mov ah, 09h

int 21h

jmp exit

moreThen:

lea dx, moreThenException

mov ah, 09h

int 21h

jmp exit

lessThen:

lea dx, lessThenException

mov ah, 09h

int 21h

jmp exit

exit:

mov ax, 4c00h

mov al, 0

int 21h

end start