Практическая работа №3

Тема: *Программирование циклических алгоритмов. Операции с памятью. Обработка структур данных (массивов).*

Цель работы – научиться разработке простейших алгоритмов обработки массивов и их реализации на языке Ассемблера.

Теоретическая часть

Массив (array) – индексированная последо-вательность однотипных данных. Представ-ляет собой последовательность элементов, каждый из которых имеет порядковый ин-декс. Аналог массива в математике – вектор.

идентификатор

$$A[5]=0_0,15_1,-25_2,40_3,-15_4$$

индекс

Типы данных в ассемблере

Разрядность	Вид	Название	Диапазон значений	Директива для описания	Регистры
Восьми-разрядные (одно-байтные)	**Без знака**	**Byte**	00h..FFh 0..255	**db**	**AL AH BL BH CL CH DL DH**
	Со знаком	**Short integer**	80h..FFh,00h,01h..7Fh -128..-1,0,1..127		
Шестнадцати-разрядные (двух-байтные)	**Без знака**	**Word**	0000h..FFFFh 0..65535	**dw**	**AX SI BX DI CX SP DX BP**
	Со знаком	**Integer**	8000h..FFFFh,0000h,0001h,..7FFFh -32768..-1,0,1..32767		

Применение директив

db (определить байт),
dw (определить слово).

Директивы для описание групп данных:
dd (4 байта), **dq** (8 байт), **dt** (10 байт). Данные описываются при написании программы в текстовом файле

Описание строится следующим образом:
Имя_переменной db значение (список значений)
Имя_переменной dw значение (список значений)

Например,
```
X db 5
Y dw 4
```

При описании массива перечисляется список значений, перечисляемых через запятую.
```
A db 1,2,3,4,7
B dw 9,-4,3,2
```

Если значения переменных в начале не известны (они рассчитываются в процессе работы программы), то они заменяются на символ «**?**».
```
X db ?
A db ?,?,?,?,?    ;5 неизвестных элементов размером 1 байт
```

После символа «;» содержимое строки в ассемблерной программе не анализируется, это комментарий.

Кроме того, можно зарезервировать сразу несколько байт памяти с помощью директивы **dup**.
```
A db 17 dup(?)     ;массив из 17-ти элементов типа байт, неизвестных
B dw 100 dup(0)  ;массив из 100 элементов типа word равных нулю.
```

Представление массивов в памяти

A db 0,15,-25,40,-15

Ячейка памяти	**00**	**0F**	**E7**	**28**	**F1**					
Адрес памяти	0200	0201	0202	0203	0204	0205	0206	0207	0208	0209
Индекс элемента	0	1	2	3	4					

A dw 0,15,-25,40,-15

Ячейка памяти	**00**	**00**	**0F**	**00**	**E7**	**FF**	**28**	**00**	**F1**	**FF**
Адрес памяти	0200	0201	0202	0203	0204	0205	0206	0207	0208	0209
Индекс элемента	0		1		2		3		4	

Адресация

Для того, чтобы *обратится к ячейке памяти*, мы должны *указать её адрес*.

Указанием на обращение к памяти в командах ассемблера служат **квадратные скобки**.

mov al,a[si] (в программном коде)

Способ задания операндов в команде называется **типом адресации.**

Он определяет возможные способы задания адреса памяти, с которым будет работать данная команда.

Виды адресации

№	Название	Описание	Пример
1	**Непосредственный**	Операнд задается в команде	mov ax,**5**
2	**Регистровый**	Операнд задается в регистре	mov ax,**bx**

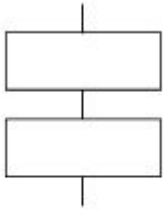
3	**Относительный**	Адрес ячейки указывается непосредст-венно. Загрузка фиксированной ячейки памяти	mov ax,**[0200]**
			mov ax,a[2]
			mov ax,a

4	**Базовый**	Адрес ячейки берется из регистра (могут быть использованы только BX, SI, DI). Ко-манда будет работать с той ячейкой па-мяти, адрес которой записан в регистре	mov ax,**[si]**
5	**Базовый относительный**	Адрес берется из суммы значения регис-тра (BX, SI, DI, BP) с фиксированным адресом	mov ax,**[si+0200]** mov ax,a[SI]

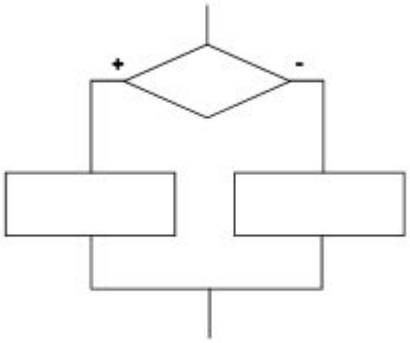
6	**Базовый индексный**	Адрес берется из суммы одного из базо-вых регистров (**BX,BP**) с индексным (**SI,DI**)	mov ax,**[bx+si]**
			mov ax,**[bx+di]**
			~~mov ax,[bx+bp]~~

7	**Базовый индексный**	Адрес берется из суммы одного из базо-вых регистров (**BX,BP**) с индексным (**SI,DI**) и с фиксированным	mov ax,**[bx+si+0200]**
			mov ax,**[bx+di+0200]**
			mov ax,a[bx][si]

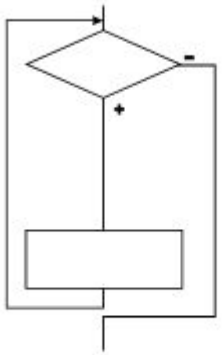
Блок-схемы алгоритмов



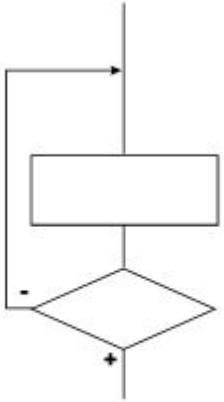
Следование



Ветвление



Цикл с предусловием
(while-do)



Цикл с постусловием
(repeat-until)

Цикл со
счетчиком
(for)

Циклы со счетчиком в ассемблере

Способ 1: *Ветвящийся алгоритм с условием сравнения и изменяющейся переменной.*

```
m0:   cmp si,5 ;сравнение
      jnl m1    ;условие выхода из цикла
      …      ;тело цикла
      inc si      ;увеличение переменной
                  ;цикла на единицу
      jmp m0  ; возвращение к условию
                  ;выхода
m1:   …      ;метка выхода из цикла
```

Циклы со счетчиком в ассемблере

Способ 2: *С помощью команды* ***LOOP****.*

Loop <метка>

Команда Loop *вычитает из регистра CX единицу* и передает управление на указанную метку, до тех пор, *пока регистр CX не равен 0*. Таким образом, регистр CX используется для указания повторений цикла.

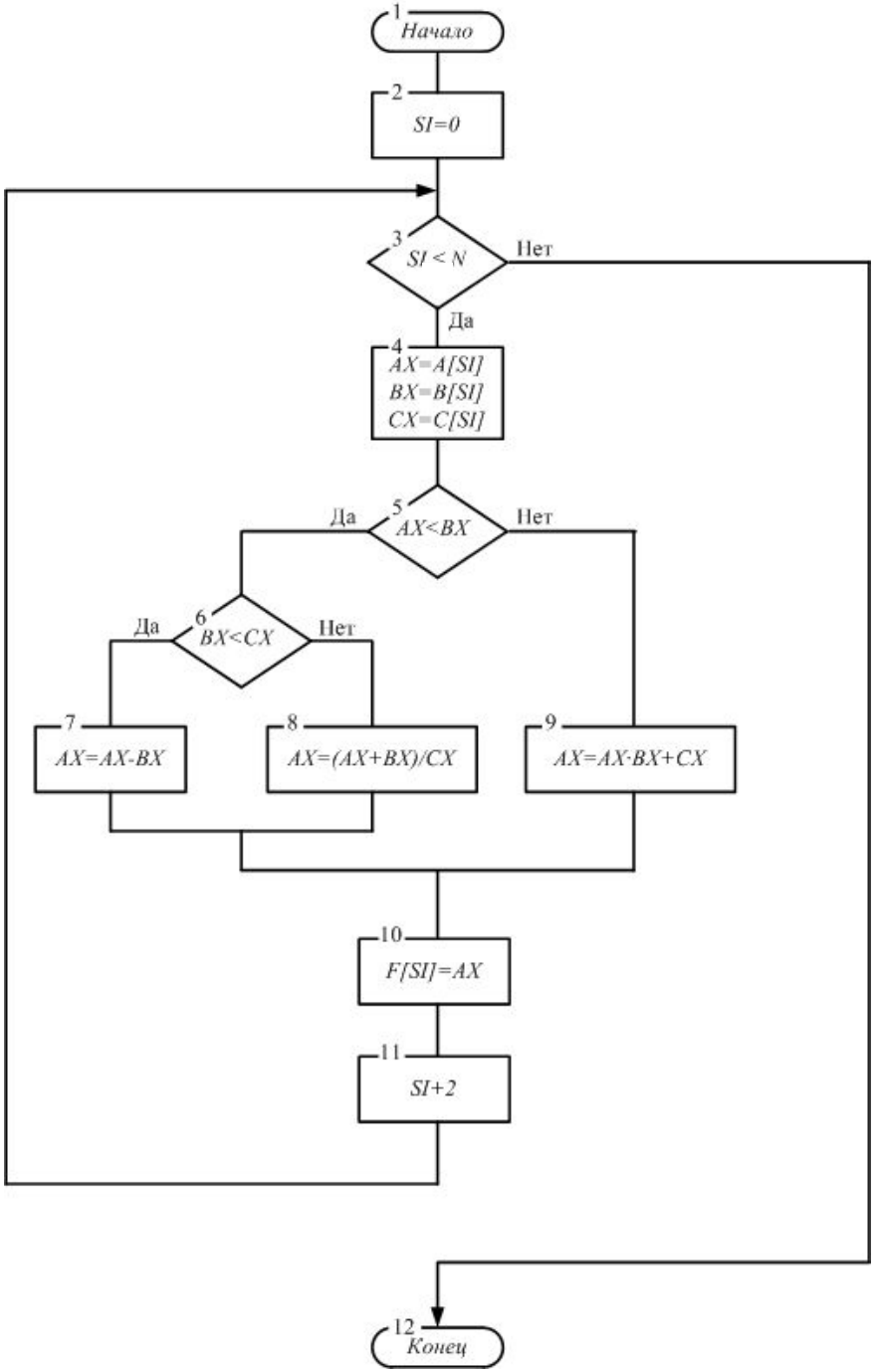
```
Mov CX,N      ;количество повторений цикла
m0:           …       ;тело цикла
              Loop m0
```

Пример задания

Заданы массивы A[N], B[N], C[N] из элементов типа integer (целое, 16-ти разрядное со знаком). Составить программу, формирующую массив F[N] в соответствии с функцией:

$$F = \begin{cases} A - B, \text{если } A < B < C \\ (A + B) / C, \text{если } A < B \geq C \\ A * B - C, \text{иначе} \end{cases}$$

Блок-схема алгоритма



Особенности представления программы в текстовом файле

.model tiny ; спецификатор модели памяти
;(односегментная)
.code ; указатель начала сегмента кода
org 100h ; переопределения счетчика адресов
start: ; точка входа (может называться как угодно,
;но должна быть)
…… ; команды вашей программы
ret ; команда возврата (в конце основной com- ;
программы – выход в операционную ;систему
a db …. ; описание данных в вашей программе

end start ; указание конца файла и точки входа. После этой
;строки не анализируются компилятором

Основные отличия программы при написание в текстовом файле и представления в отладчике

В текстовом файле	В отладчике
Jl m1 Mov ax,bx Jmp m3 m1: mov bx,ax m3:	0110 Jl 0117 0112 Mov ax,bx 0114 Jmp short 0119 * 0117 mov bx,ax 0119
Mov ax,a[SI]	0110 mov ax,[SI+0130] *(0130 – адрес переменной a в памяти)*

Mov al,a[si]
Mov bx,b[di]
Mov ax,X
Mov X,5
A db 2,3,4
B dw 5,6,-2
X dw 4
	Mov al,[si+**0140**]
	Mov bx,[di+**0170**]
	Mov ax,[**0180**]
	Mov byte ptr [**0180**],5 **
	В окне данных в отладчике (внизу) по адресу, соответствующему переменной А (можно узнать из команды обращения к данной переменной)
	0CB0:**0140 02 03 04 00** 00 00 00 00 00 00 00
	0CB0:**0170 05 00 06 00 FE FF** 00 00 00 00
	0CB0:**0180 04 00** 00 00 00

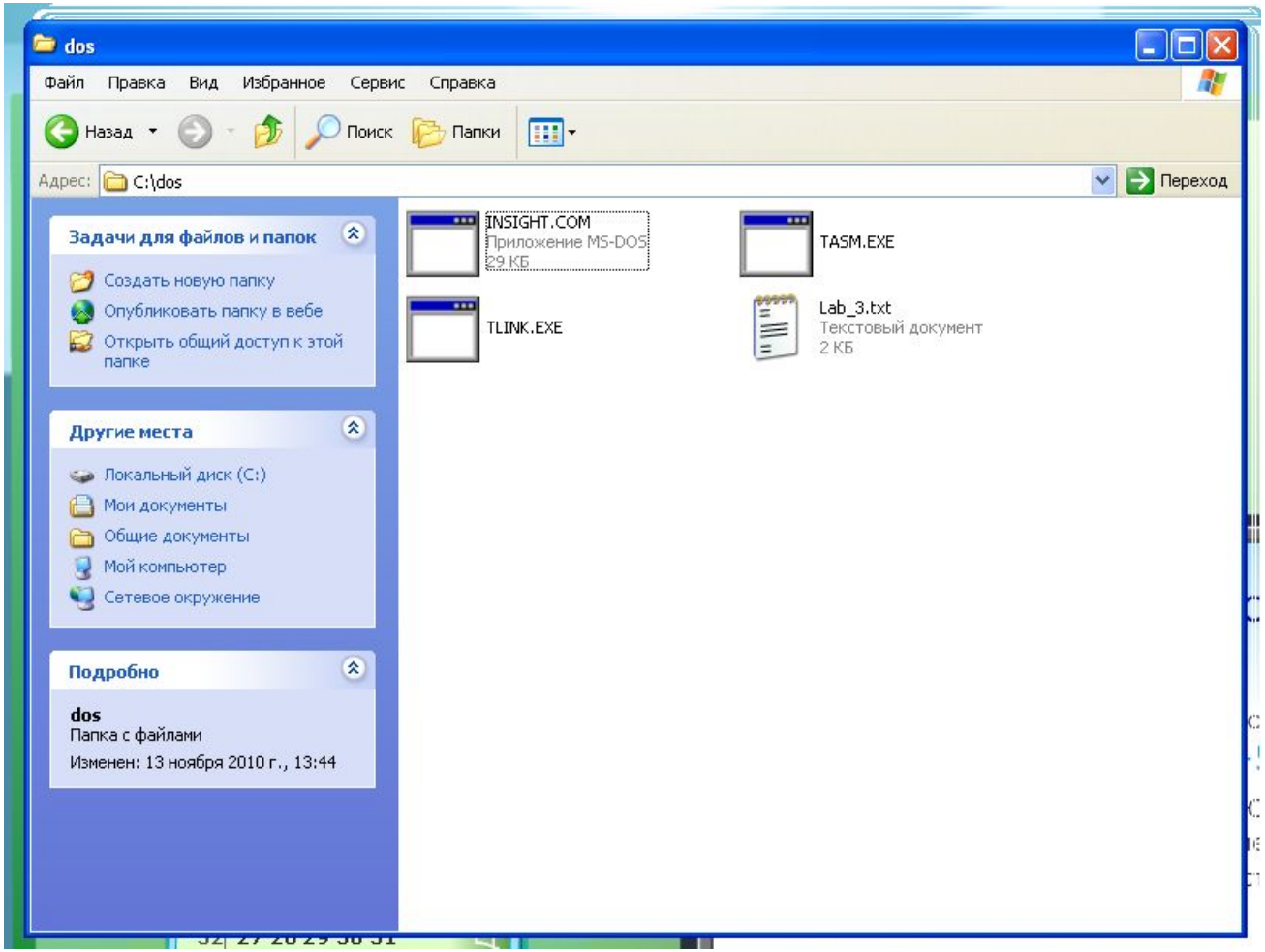
Числа указываются в десятичной системе счисления по умолчанию. Можно также указывать в шестнадцатеричной, двоичной, восьмеричной: A db 32 A db 20h A db 00100000b A db 40q	Все числа отображаются как шестнадцатеричные 0CB0:**0140 20**
	(short – указание на короткий переход, в пределах 127 байт, по умолчанию формируется компилятором, если метка расположена по программному коду ближе 127 байт)* * (указатели размера операнда – byte ptr и word ptr используются тогда, когда нельзя определить, слово или байт (5) необходимо записать в память? Иногда их приходится использовать и в тексте программы)*

Кодирование алгоритма

```
.model tiny      ;спецификатор модели памяти (односегментная)
.code            ;указатель начала сегмента кода
org 100h         ;переопределение счетчика адресов
start:           ;точка входа (может называться как угодно, но должна быть)
mov si,0         ;указатель индекса массивов приравниваем к 0
        ;начало цикла
m0:     cmp si,6         ;проверяем условие выхода из цикла
        jnl m1           ;если указатель индекса >=6, выход из программы
        mov ax,A[si]     ;загружаем в регистры соответствующие элементы массивов
        mov bx,B[si]
        mov cx,C[si]
        cmp ax,bx        ;начало вычисления функции F аналогично работе №2
        jl m2
        imul bx
        sub ax,cx
        jmp m4
m2:     cmp bx,cx
        jl m3
        add ax,bx
        cwd
        idiv cx
        jmp m4
m3:     sub ax,bx        ;конец вычисления функции F, результат получен в ax
m4:     mov F[si],ax     ;запись результата вычислений в ячейку памяти
        add si,2         ;сдвиг индекса массива (на 2, т.к. тип данных двухбайтный)
        jmp m0           ;переход в начало цикла
m1:     ret              ;команда возврата (в конце основной com-программы -
                ;- выход в операционную систему
A dw 1,1,3       ;задание массивов A,B и C
B dw 3,3,1
C dw 5,2,2
F dw 3 dup(?)    ;резервирование трех ячеек памяти для массива F
end start        ;указание конца файла и точки входа
                 ;после этой, строки не анализируются компилятором.
```

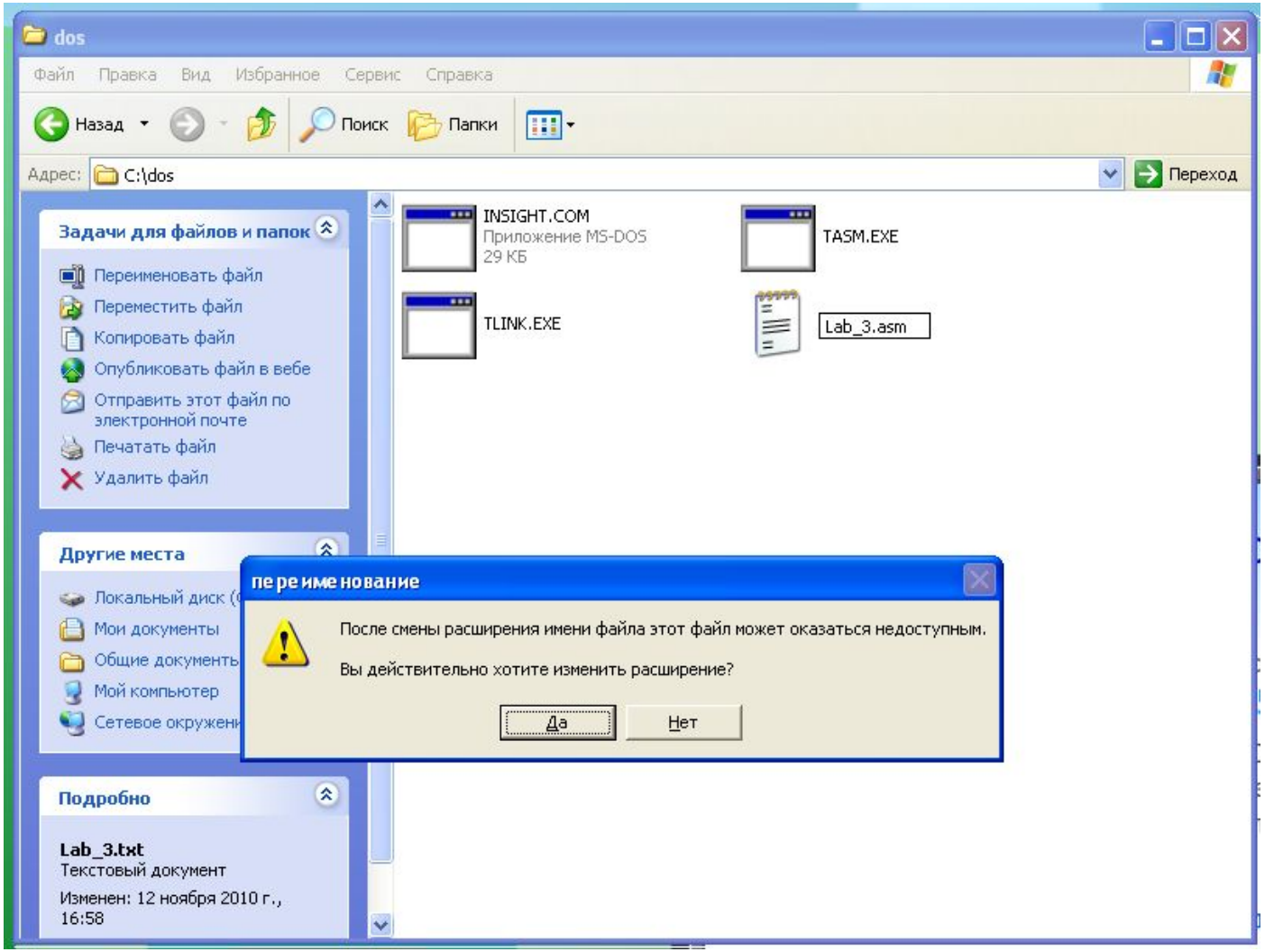
Компиляция

1. Создать в корне диска (для облегчения доступа) папку и поместить в нее текстовый файл с программой, а также файл отладчика (**Insight.com**), компилятора (**Tasm.exe**) и компановщика (**Tlink.exe**)



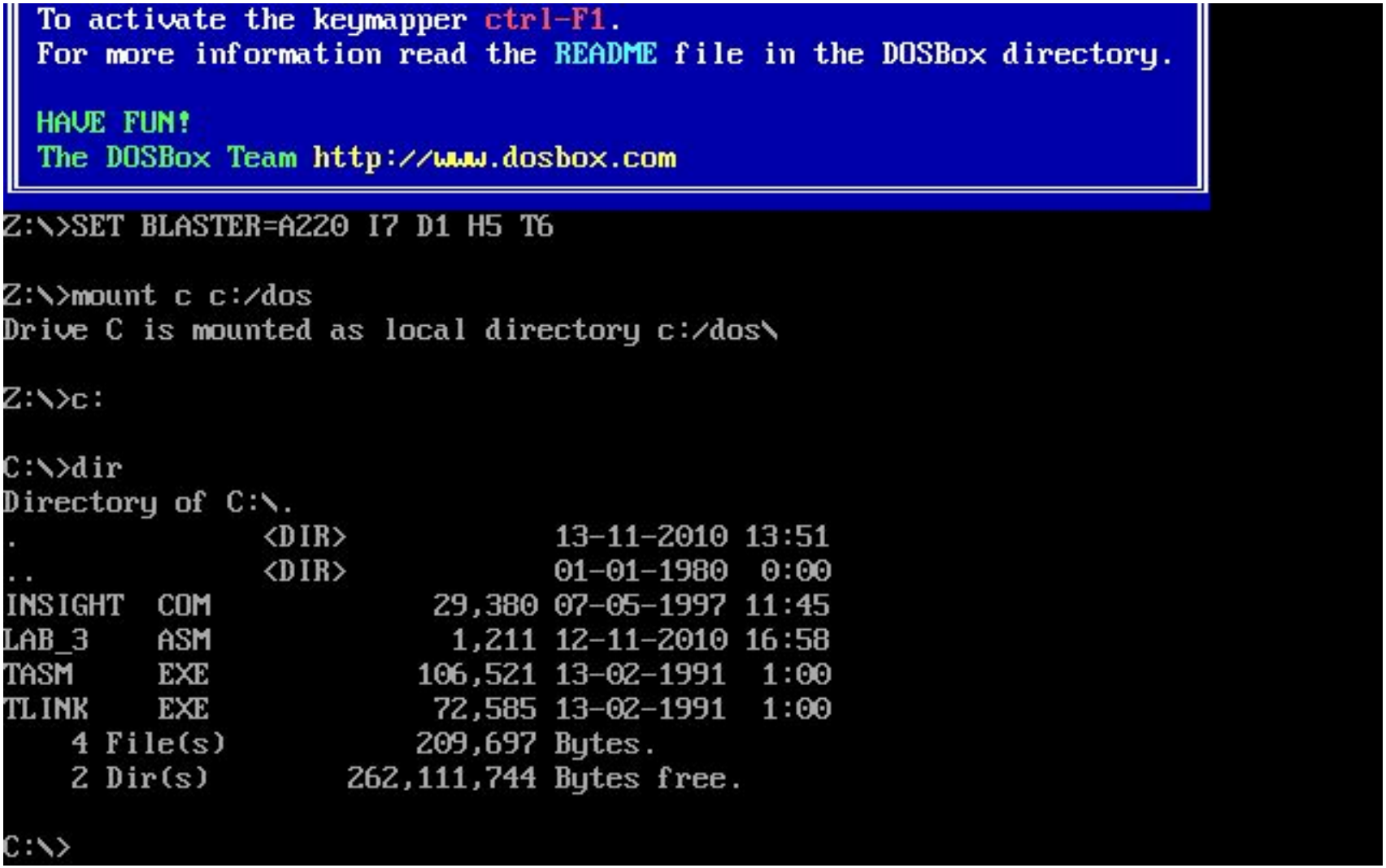
Компиляция

2. Изменить разрешение Вашего файла с **.txt** на **.asm**



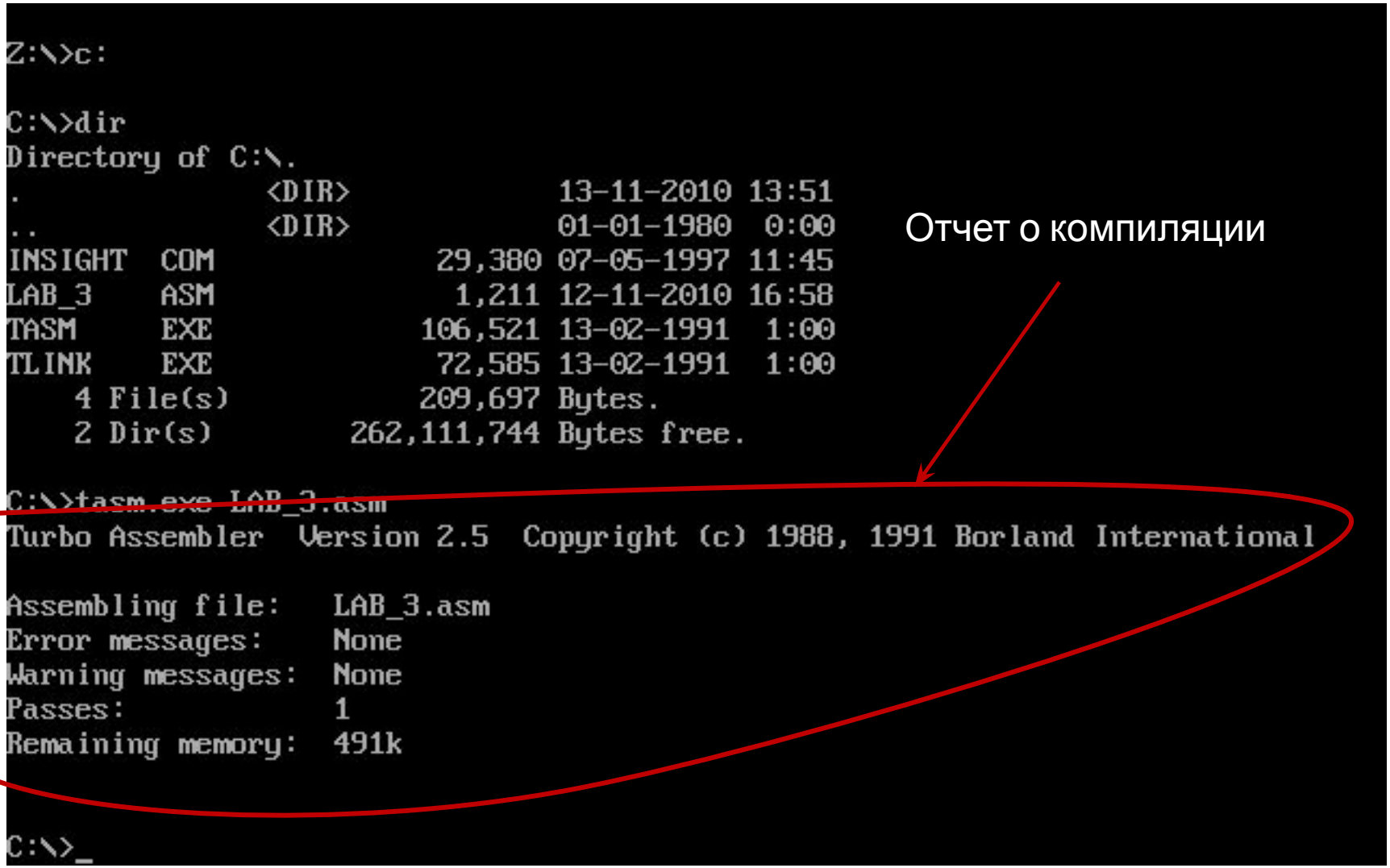
Компиляция

3. Запустить любой эмулятор (**Far manager, DOS Box**), поддерживающий командную строку DOS, и перейти в данную папку



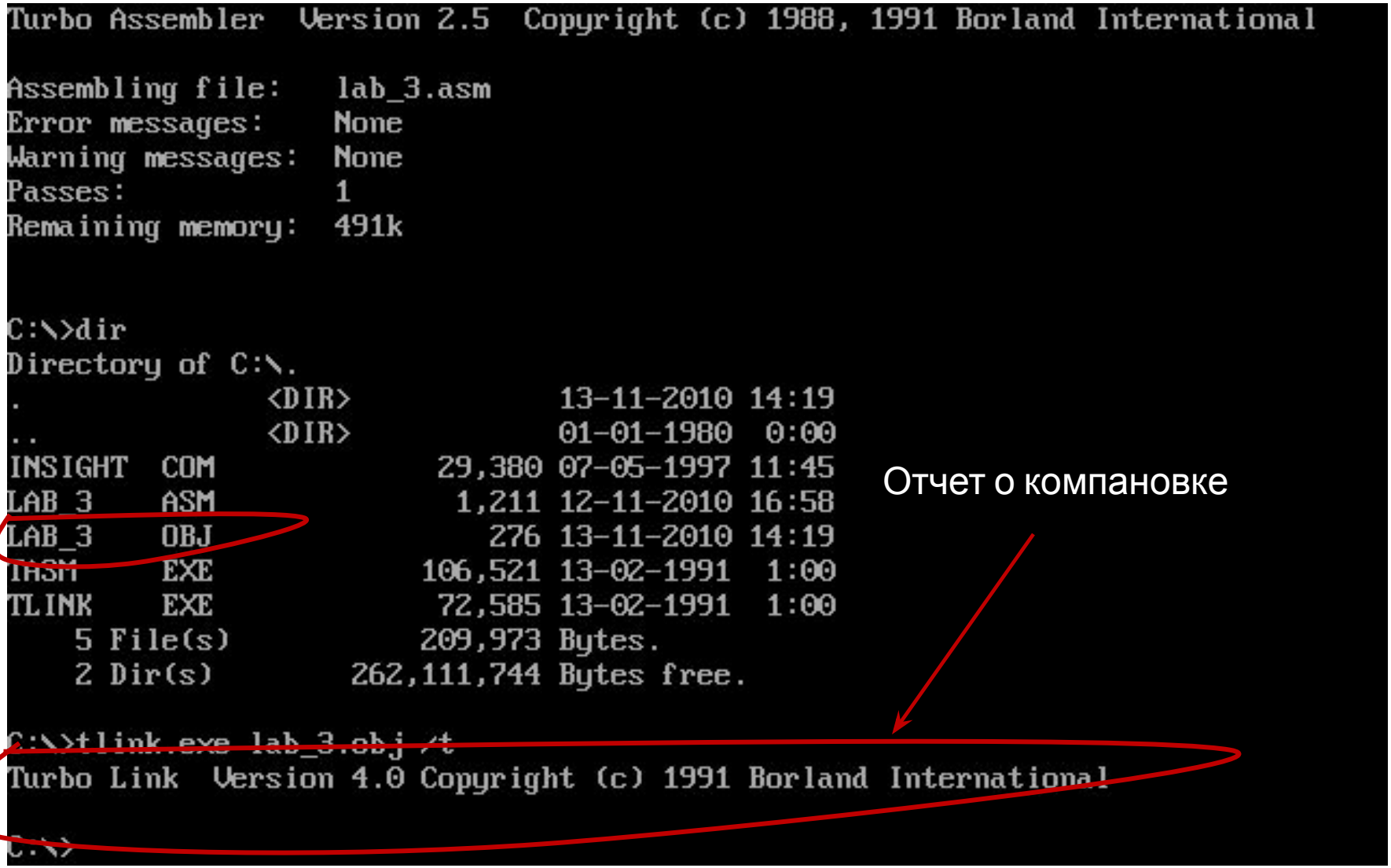
Компиляция

4. Набрать в командной строке: **tasm.exe_LAB_3.asm** и нажмите **ENTER**



Компиляция

5. Убедившись, что появился файл **LAB_3.obj**, набрать в командной строке: **tlink.exe_ LAB_3.obj _/t** и нажать **ENTER**



Загрузка программы в отладчик

В результате работы компилятора и компановщика должен появиться файл с расширением **.com**. Этот файл и загружается в отладчик командой: **Insight.com_ LAB_3.com**

```
LAB_3    ASM              1,211 12-11-2010 16:58
LAB_3    OBJ                276 13-11-2010 14:19
TASM     EXE            106,521 13-02-1991  1:00
TLINK    EXE             72,585 13-02-1991  1:00
    5 File(s)           209,973 Bytes.
    2 Dir(s)        262,111,744 Bytes free.

C:\>tlink.exe lab_3.obj /t
Turbo Link  Version 4.0 Copyright (c) 1991 Borland International

C:\>dir
Directory of C:\.
.              <DIR>            13-11-2010 14:20
..             <DIR>            01-01-1980  0:00
INSIGHT  COM            29,380 07-05-1997 11:45
LAB_3    ASM             1,211 12-11-2010 16:58
LAB_3    COM                73 13-11-2010 14:20
LAB_3    MAP               150 13-11-2010 14:20
LAB_3    OBJ               276 13-11-2010 14:19
TASM     EXE            106,521 13-02-1991  1:00
TLINK    EXE             72,585 13-02-1991  1:00
    7 File(s)           210,196 Bytes.
    2 Dir(s)        262,111,744 Bytes free.

C:\>insight.com lab_3.com_
```

Программа в отладчике с результатами работы