**Разработка многомодульной программы.**

**Цель:** закрепление навыков разработки и отладки многомодульной программы.

Задание: разработать многомодульную программу на языке ассемблера. Эта программа состоит из модуля верхнего уровня MAIN, обеспечивающего своевременный вызов программных модулей, каждый из которых содержит только одну процедуру, предназначенную для решения выделенной подзадачи.

Создаваемая многомодульная программа должна обеспечивать возможность последовательного ВВОДА с клавиатуры двух двоичных N-разрядных чисел. Для ввода числового объекта и маски необходимо создать программный модуль INPUT, обеспечивающий ввод с клавиатуры двоичного числа нужной разрядности, а своевременный вызов этого модуля осуществляется из программного модуля MAIN.

Создаваемая программа должна после ВВОДА числового объекта и маски осуществить преобразование числового объекта и количественно оценить результат этого преобразования. Для указанных действий необходимо создать соответствующий программный модуль MASK преобразования числового объекта.

После решения подзадачи преобразования числового объекта, осуществляемого модулем MASK, необходимо в программном модуле MAIN, во-первых, выполнить подсчет количества единиц (или нулей) в преобразованном числовом объекте, во-вторых, проверить наличие заданного свойства преобразованного числового объекта и по результату проверки осуществить вызов либо программного модуля OUTSTAR, либо модуля OUTNUMBER.

Если количество единиц (или нулей) числового объекта характеризуется заданным свойством, то активизируется работа программного модуля OUTSTAR, выполняющего отображение на экране символа звездочки (\*) в количестве, равном количеству единиц (или нулей) в числовом объекте.

Если количество единиц (или нулей) в числовом объекте не характеризуется заданным свойством, то вызывается программный модуль OUTNUMBER, осуществляющий вывод на экран десятичного числа, соответствующего количеству единиц (или нулей) в преобразованном числовом объекте. После вывода на экран десятичного числа программный модуль MAIN активизирует работу программного модуля FILECHANGE обработки предварительно созданного текстового файла.

Обработка файла заключается, во-первых, в перемещении содержимого файла в буфер, созданный в программе. Во-вторых, в добавлении к элементам файла новых элементов в соответствии с вариантами задания. В-третьих, в осуществлении вывода отредактированного текста на экран и в размещении его в новом файле, который необходимо создать. Вывод текста на экран осуществляется по согласованию с преподавателем либо с помощью специальных функций операционной системы, либо путем пересылки текста в память видеоадаптера.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вариант | N | А | В | Свойство X | Что нужно заменять (добавлять) |
| 6 | 8 | Инверсное | 1 | Кратное 5 | В начало файла свое имя |

**Код программы:**

**LAB14.asm:**

TITLE MAIN

EXTRN INPUT:far,mask1:far,input\_gone\_wrong\_flag:byte,outnum:far,outstar:far,putfile:far

PUBLIC resmask,msk,object

.model small

.stack 10h

.data

string\_mask db 10,13,'Mask, 8 bits (1 - invert the digit of the number 0 - non-inverted): $'

string\_number db 10,13,'Number: $'

string\_press\_any\_key db 10,13,'Press any key to continue... $'

msk dw ?

object dw ?

resmask db ?

.code

MAIN:

mov ax,@data

mov ds,ax

INPUT\_label:

mov ah,9

lea dx,string\_mask

int 21h

mov msk,0

lea di,msk

call INPUT

cmp input\_gone\_wrong\_flag,1

je INPUT\_label

mov ah,9

lea dx,string\_number

int 21h

mov object,0

lea di,object

call INPUT

cmp input\_gone\_wrong\_flag,1

je INPUT\_label

call mask1

cmp resmask,0

je OUTNUM\_label

mov ax,0

mov bl,5

mov al,resmask

div bl

cmp ah,0

jne OUTNUM\_label

call outstar

jmp PUTFILE\_label

OUTNUM\_label:

call outnum

PUTFILE\_label:

call putfile

mov ah,9

lea dx,string\_press\_any\_key

int 21h

mov ah,1

int 21h

Mov ax,4c00h

Int 21h

End MAIN

**INPUT.asm:**

; Дополнительный исходный файл

TITLE input

Public INPUT, input\_gone\_wrong\_flag

EXTRN msk:byte,object:byte

.Model Small

.STACK 100h

.data

input\_gone\_wrong\_flag db ?

string\_wrong\_number db 10,13,'Wrong number $'

.code

INPUT proc

mov ax,@data

mov ds,ax

mov cx,8

mov input\_gone\_wrong\_flag,0

main\_input\_loop:

mov ah,1

int 21h

cmp al,31h

jz case\_1

cmp al,30h

jz case\_0

mov ah,9

lea dx,string\_wrong\_number

int 21h

mov input\_gone\_wrong\_flag,1

jmp exit

case\_1:

stc

rcl word ptr[di],1

jmp iteration\_end

case\_0:

clc

rcl word ptr[di],1

iteration\_end:

loop main\_input\_loop

exit:

ret

INPUT endp

End

**OUTSTAR.asm:**

; Дополнительный исходный файл

TITLE OUTSTAR

Public OUTSTAR

EXTRN resmask:byte

.Model Small

.STACK 100h

.data

string\_endl db 13,10,'$'

.code

OUTSTAR proc

mov ax,@data

mov ds,ax

mov ah,9

lea dx,string\_endl

int 21h

mov cl,resmask

mov dl,2Ah

mov ah,2

star\_print\_loop:

int 21h

loop star\_print\_loop

ret

OUTSTAR endp

end

**MASK1.asm**

; Дополнительный исходный файл

TITLE input

Public INPUT, input\_gone\_wrong\_flag

EXTRN msk:byte,object:byte

.Model Small

.STACK 100h

.data

input\_gone\_wrong\_flag db ?

string\_wrong\_number db 10,13,'Wrong number $'

.code

INPUT proc

mov ax,@data

mov ds,ax

mov cx,8

mov input\_gone\_wrong\_flag,0

main\_input\_loop:

mov ah,1

int 21h

cmp al,31h

jz case\_1

cmp al,30h

jz case\_0

mov ah,9

lea dx,string\_wrong\_number

int 21h

mov input\_gone\_wrong\_flag,1

jmp exit

case\_1:

stc

rcl word ptr[di],1

jmp iteration\_end

case\_0:

clc

rcl word ptr[di],1

iteration\_end:

loop main\_input\_loop

exit:

ret

INPUT endp

end

**PUTFILE.asm:**

; Дополнительный исходный файл

TITLE putfile

Public putfile

.model small

.stack 100h

.data

string\_err db 13,10,'ERR $'

string\_list\_start db 13,10, 'listing of newtext.txt: $'

string\_list\_end db 13,10, 'end of listing $'

string\_name db 10,13,'Name (will be putted in the start of newtext.txt): $'

string db 13,10

string\_endl db 10,13,'$'

buf db 100 dup(?)

buf\_new db 15 dup(?)

OldFileName db 'C:\text.txt', 0

FM dw ?

NewFileName db 'C:\newtext.txt', 0

.code

putfile proc

mov ax, @data

mov ds, ax

clc

mov ax, 3D00h

lea dx, OldFileName

int 21h

mov dl,'O'

jc Error

mov FM, ax

mov ah, 3Fh

mov bx, FM

mov cx, 100

lea dx, buf

int 21h

mov dl, 'R'

jc Error

mov bx, FM

mov ah, 3Eh

int 21h

mov dl, 'C'

jc Error

mov ax,3c00h

lea dx,NewFileName

int 21h

jc Error

mov FM,ax

mov ah,9

lea dx,string\_name

int 21h

lea di,buf\_new

mov cx,0

r1:

mov ah,1

int 21h

cmp al,0dh

je r2

mov [di],al

inc di

inc cx

jmp r1

r2:

mov bx,FM

mov ah,40h

lea dx,buf\_new

int 21h

jc Error

mov byte ptr[di],' '

inc di

mov byte ptr[di],'$'

mov ah,40h

mov cx,2

lea dx,string

int 21h

mov ah,40h

lea dx,buf

mov cx,100

int 21h

jc Error

jmp buf\_start

Error:

mov ah,9

lea dx,string\_err

int 21h

mov ah, 02h

int 21h

jmp return

buf\_start:

lea bx,buf

add bx,ax

mov byte ptr[bx-1],'$'

mov ah, 3Eh

mov bx, FM

int 21h

mov dl, 'C'

jc Error

mov ah,9

lea dx,string\_list\_start

int 21h

mov ah,9

lea dx,string\_endl

int 21h

mov ah,9

lea dx,buf\_new

int 21h

mov ah,9

lea dx,string\_endl

int 21h

mov ah,9

lea dx,buf

int 21h

mov ah,9

lea dx,string\_list\_end

int 21h

return:

ret

putfile endp

end

**OUTNUM.asm:**

; Дополнительный исходный файл

TITLE OUTNUM

Public OUTNUM

EXTRN resmask:byte

.Model Small

.STACK 100h

.data

string\_found db 10,13,'Found $'

string\_bits db ' bits equal to one$'

.code

OUTNUM proc

mov ax,@data

mov ds,ax

mov ah,9

lea dx,string\_found

int 21h

mov dl,resmask

add dl,30h

mov ah,2

int 21h

mov ah,9

lea dx,string\_bits

int 21h

ret

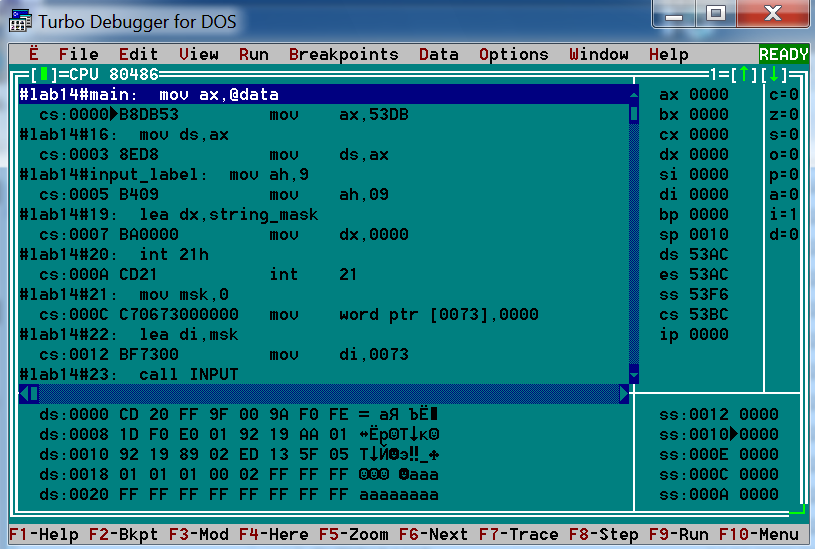
OUTNUM endp

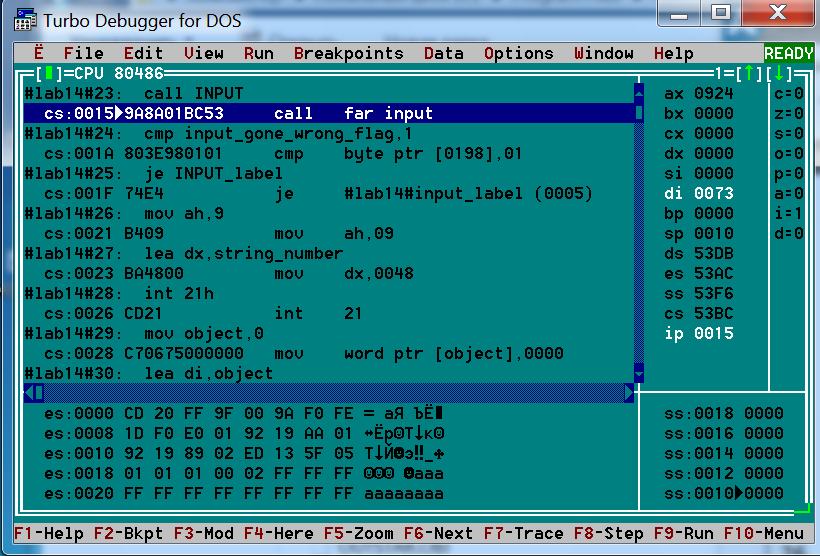
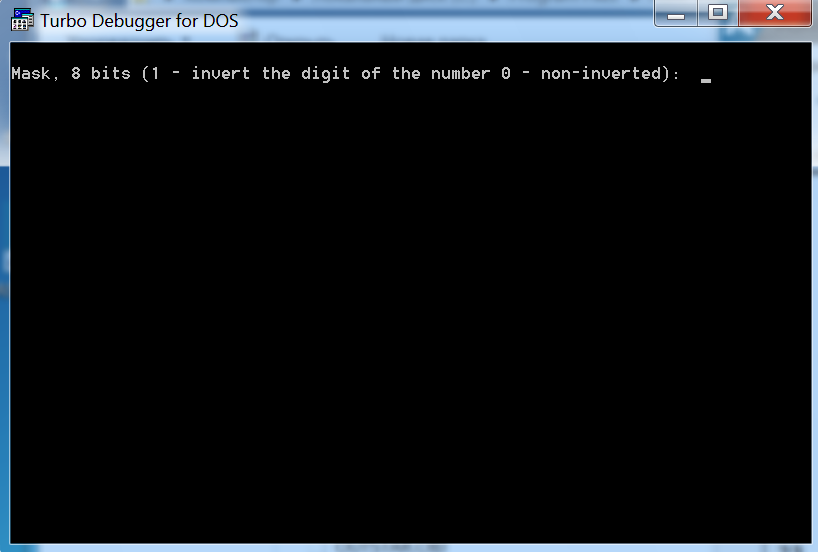
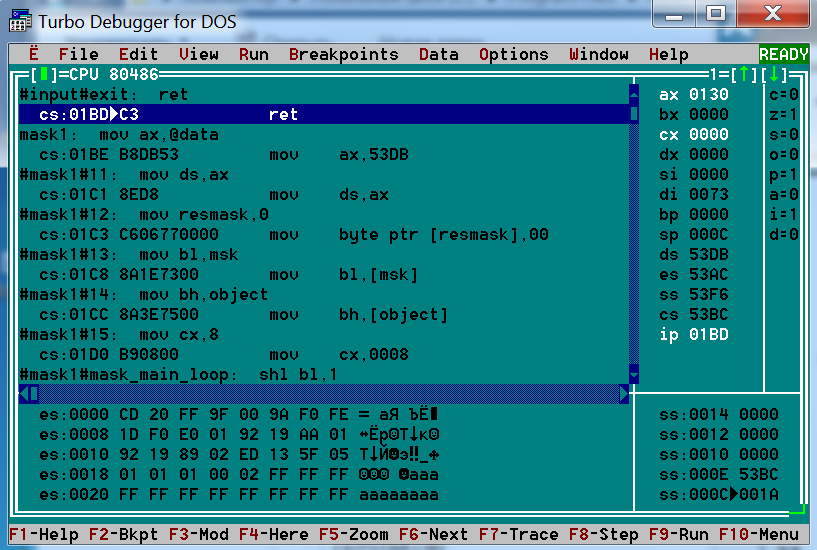
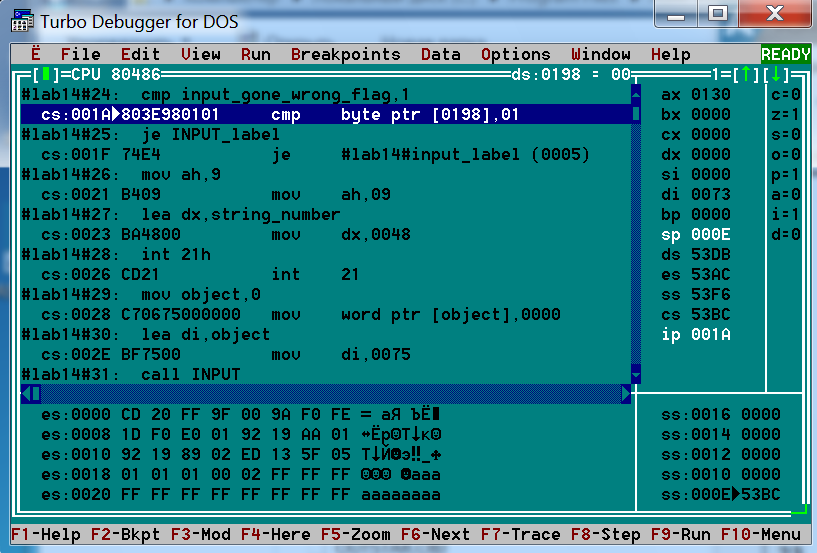
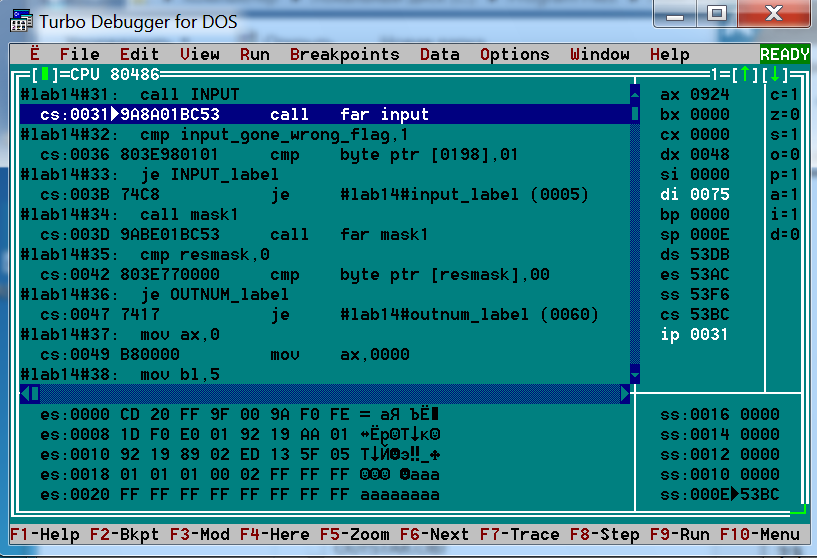
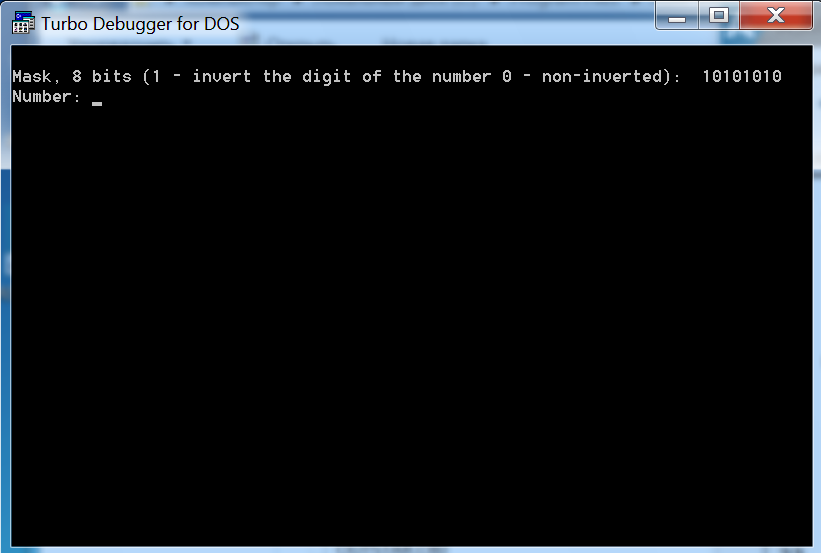
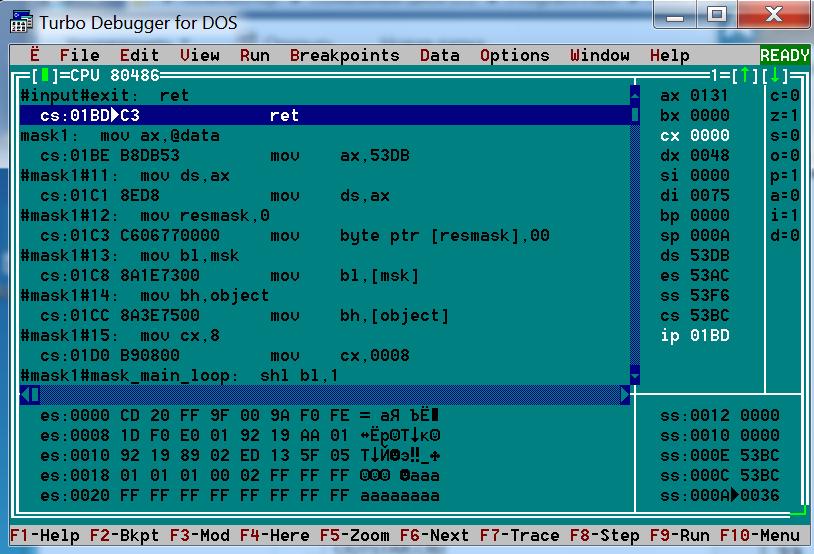
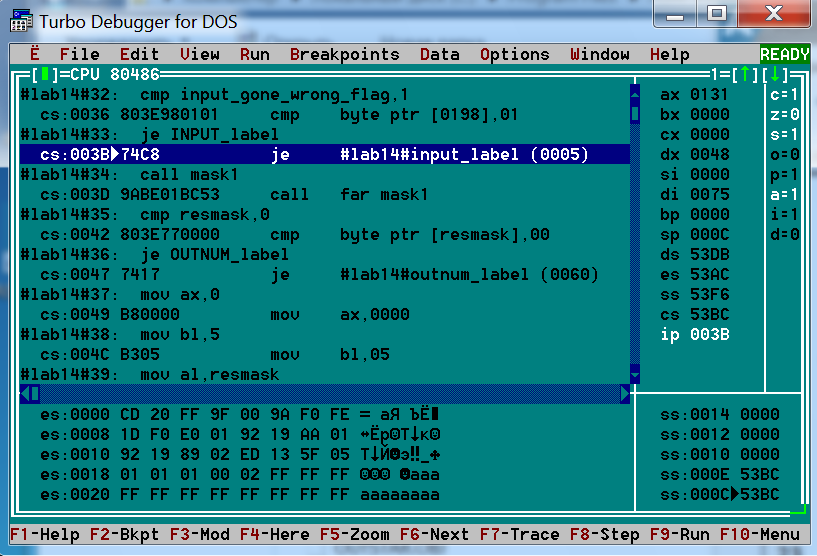
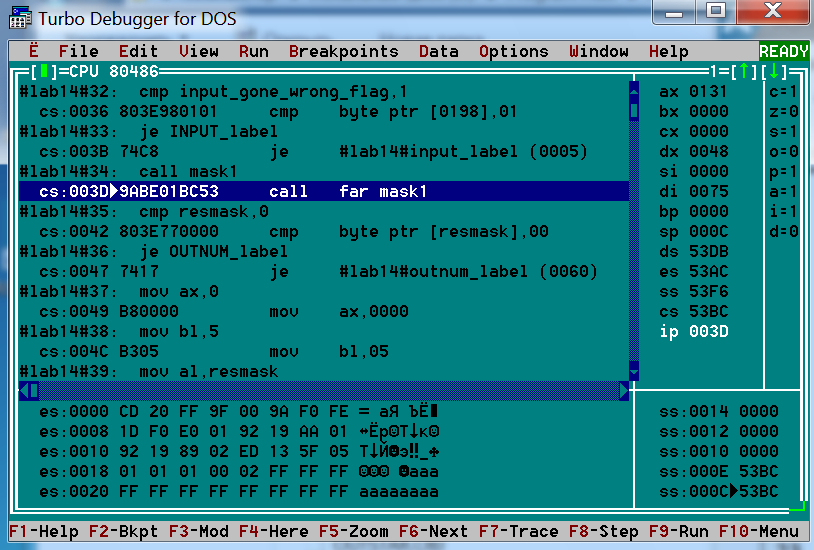
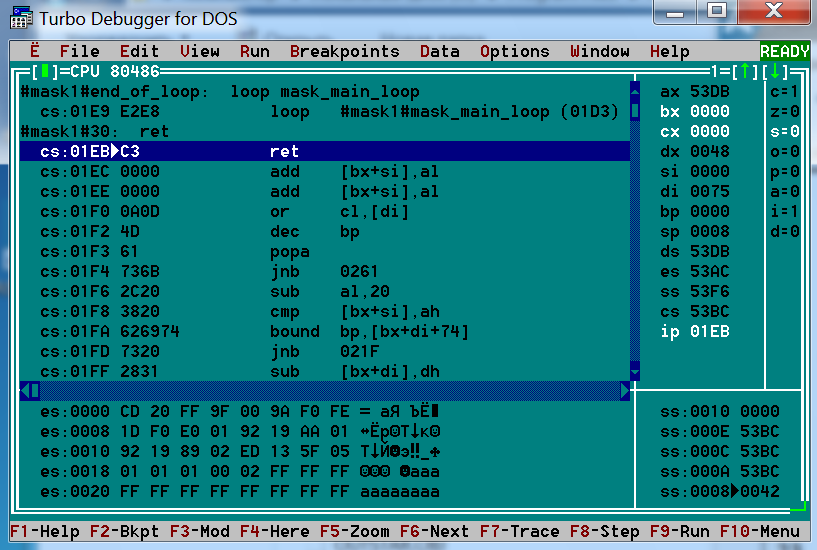
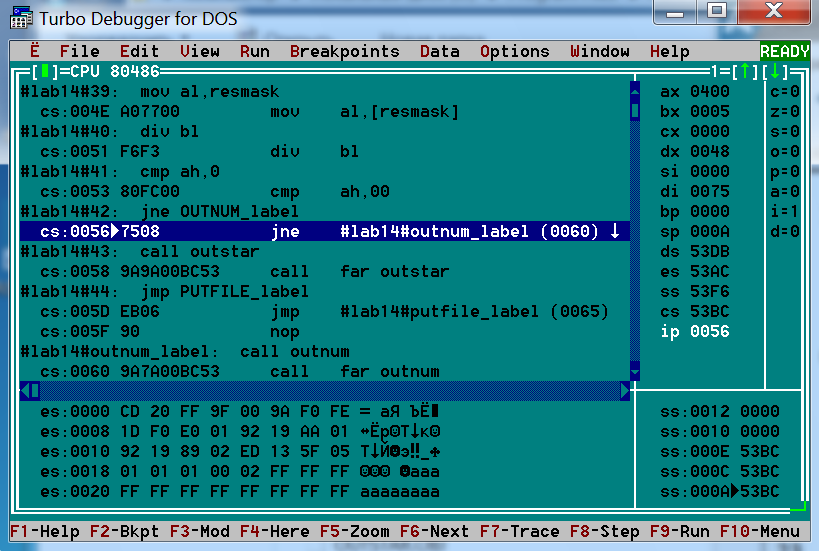
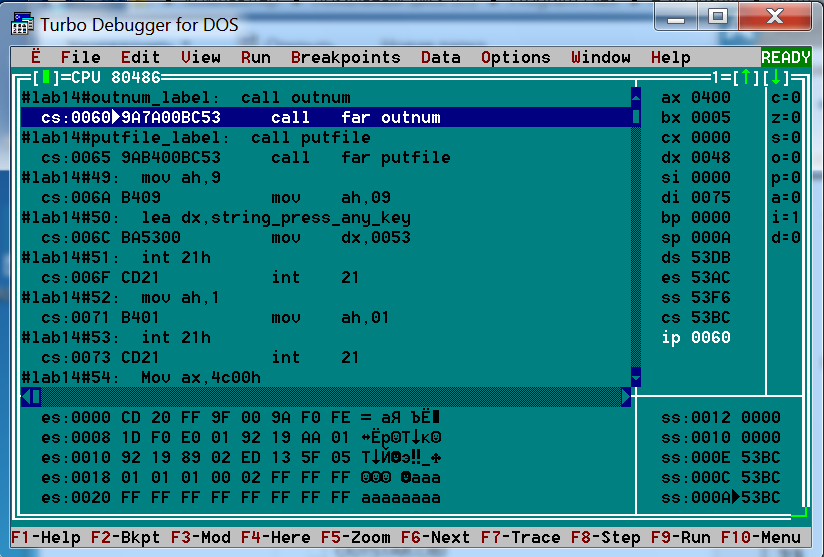
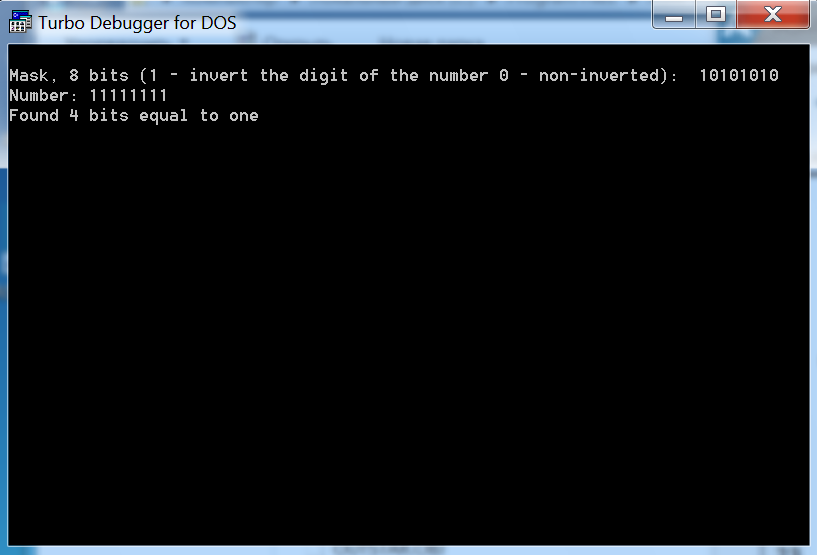
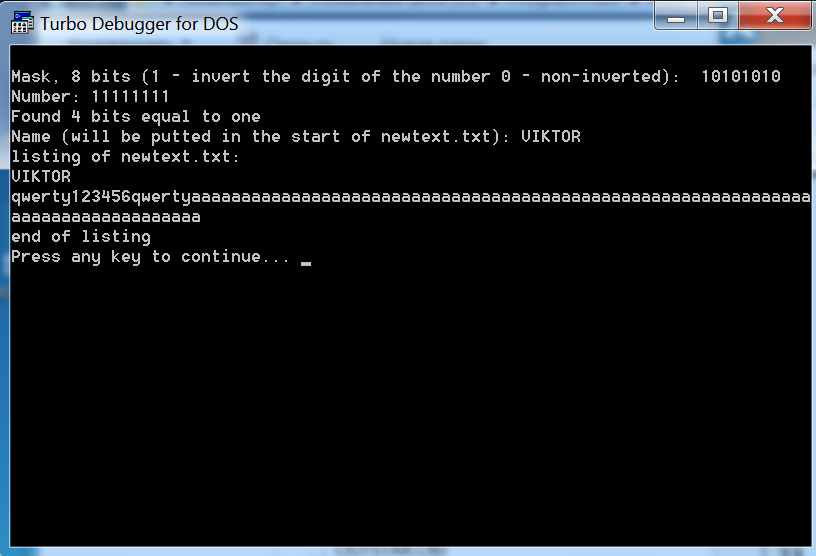
End

**Тестирование программы:**

**Тест 1. Маска 1010 1010 число 1111 1111 => 01010101 => вывод сообщений о 4 единицах в результате**

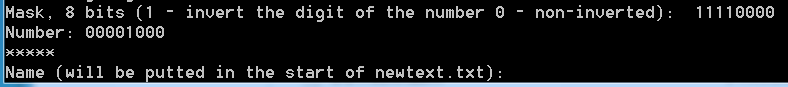
1. **Программа до отладки**



1. Вызов INPUT  
   
2. Запрос ввода маски  
   
3. Выход из INPUT после ввода маски  
   
4. Проверка успешности ввода маски  
   
5. Вызов INPUT для числа  
   
6. Вывод сообщения для ввода числа  
   
7. Возврат из INPUT в MAIN  
   
8. Проверка флага корректности ввода  
   
9. Вызов обработчика маски и числа  
   
10. Возврат из обработчика  
    
11. Проверка результата на соответствие условию задачи для вывода количества бит равных 1 в результате  
    
12. Вызов вывода количества бит равных 1 в результате применения маски к числу  
    
13. Вывод значения количества бит равных 1 в консоль  
    
14. Программа после дебагинга, на экран также отображены значения NULL  
    

|  |  |
| --- | --- |
| Листинг файла text.txt | Листинг файла newtext.txt |
| qwerty123456qwerty | VIKTOR  qwerty123456qwerty |

**Тест 2 Маска 1111 0000 число 0000 1000 => 1111 1000 => результат соответствует вызову OUTSTAR**

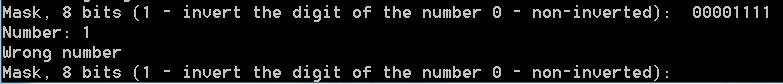
****

**Тест 3 Маска 0000 0000 число 0000 0000 => 0000 0000 => вывод сообщения о 0 единиц в результате**

****

**Тест 4 Маска 0000 1111 число 1111 0000 => 1111 1111 => вывод сообщения о 8 единицах в результате**

****

**Тест 5 Неверный ввод числа  
**

**Вывод:**

1. В данной лабораторной работе применялись знания, полученные в ходе выполнения 6,9, 10 и 12 лабораторных работ, в частности из лабораторной работы 10 использованы модифицированные участки кода для работы с числами в двоичной системе счисления, а из 12 взяты наработки для работы с файлами.
2. В ходе работы было освоено совместное применение тем:
   1. многомодульные программы – программа состоит из шести модулей, Управляющий модуль MAIN имеет связь с каждым из следующих исполнительных модулей: INPUT, MASK, OUTSTAR, OUTNUMBER, FILECHANGE.
   2. Работа с файлами
   3. Обработка чисел в двоичной системе счисления
   4. Вывод на экран результатов вычислений
   5. Ввод чисел, задаваемых в различных системах счисления
   6. Процедуры
   7. Ветвление
3. Освоены и применены основные принципы тестирования:
   1. Функциональное тестирование
   2. Тестирование граничных значений
   3. Негативное тестирование
4. В результате выполнения программа применяет инверсию бит введённого числа в соответствии с задаваемой маской, проверяет количество бит равных 1 в результирующем числе и выводит в зависимости от условия количество \* или количество бит равных единице в результирующем числе. Помимо обработки чисел программа считывает текстовый файл из корня диска C, модифицирует его и сохраняет новый файл также в корень диска С.