Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №5

на тему:

**«РАБОТА С ФАЙЛАМИ»**

БГУИР 6-05-0612-02 73

|  |
| --- |
| Выполнила студентка группы 353504  ЛЕБЕДЕВА Милана Валерьевна |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 3.** Подсчитать число не пустых строк в файле.

# 2 Выполнение работы

Для корректной обработки результатов используются соответствующие сообщения и переменные. mess\_noData, mess\_noFileName, mess\_start и т.д. – это строки, которые используются для вывода различных сообщений в программе. LENGTH – максимальная длина командной строки, cmdLengthFact – переменная для хранения фактической длины командной строки, CMD – переменная для хранения командной строки, descriptor – дескриптор файла, SIZE – размер буфера, buffer – буфер для хранения прочитанных данных, addLines, cNonEmptyLines, cChars, rf, eof – переменные для хранения различных счетчиков и флагов. Ниже представлен листинг кода программы.

.model small

.stack 100h

.data

;MESSAGES

mess\_noData db "No data.", 0Ah, 0Dh, '$'

mess\_noFileName db "No file with this name.", 0Ah, 0Dh,'$'

mess\_start db "...started working with your file...", 0Ah, 0Dh, '$'

mess\_end db "...finished working with your file...", 0Ah, 0Dh, '$'

mess\_lines db "The number of non-empty lines in the file: ", '$'

mess\_additionalLines db "Additional number: ", 0Ah, 0Dh, '$'

mess\_errorFile db "ERROR: the file was not found.", 0Ah, 0Dh, '$'

mess\_errorPath db "ERROR: the path was not found.", 0Ah, 0Dh,0Ah, 0Dh, '$'

mess\_errorFiles db "ERROR: there are too many open files.",0Ah, 0Dh, '$'

mess\_errorAccess db "ERROR: access to the file is denied.",0Ah, 0Dh, '$'

mess\_errorAccessMode db "ERROR: invalid access mode.",0Ah, 0Dh, '$'

mess\_errorDescriptor db "ERROR: invalid descriptor.",0Ah, 0Dh, '$'

mess\_errorR db "ERROR: the file cannot be read.", 0Ah, 0Dh, '$'

mess\_successfulOpen db "The file has been opened successfully.",0Ah, 0Dh, '$'

mess\_successfulClose db "The file has been closed successfully.",0Ah, 0Dh, '$'

mess\_successfulRead db "The file has been read successfully.",0Ah, 0Dh, '$'

mess db 0Ah, 0Dh, '$'

;CONSTANS

LENGTH equ 126

SIZE equ 50

;FLAGS

rf dw 0

eof dw 0

;COUNTERS

cChars dw 0

cNonEmptyLines dw 0

addLines dw 0

;VARIABLES

cmdLengthFact dw ?

CMD db 126 dup(?)

descriptor dw ?

buffer db SIZE+2 dup('$')

.code

;MACROS

display macro str

pusha

display1 str

popa

endm

display1 macro str

lea dx, str

mov ah, 09h

int 21h

endm

endProg macro

mov ax, 4Ch

int 21h

endm

start:

mov ax, @data

mov ds, ax

;get arguments of CMD

call getComandLineArgs

;check CMD

mov ax, cmdLengthFact

cmp ax, 1

jle noData

;display CMD

display mess\_start

lea dx, CMD

call openFile

cmp ax, 1

je endStart

call numberLines

display1 mess\_lines

mov ax, cNonEmptyLines

call print

display1 mess

call closeFile

cmp ax, 1

je endStart

display mess\_end

jmp endStart

noData:

display mess\_noData

endStart:

endProg

;PROCEDURES

getComandLineArgs proc

push ax

push cx

mov cx, 0

mov cl, ES:[80h]

mov cmdLengthFact, cx

cmp cx, 1

jle endProc

cld ;set the line direction

mov di, 81h

mov al, ' '

rep scasb ;find ' ' in cmd to define the end of path file

dec di ;show to char after ' '

lea si, CMD

copy:

mov al, ES:[DI] ;ES:[DI] - cmd

cmp al, 0Dh ;/r

je endCopy

cmp al, 20h ;' '

je endCopy

cmp al, 9h ;tab

je endCopy

mov DS:[SI], al ;DS:[SI] - CMD(array)

inc di

inc si

jmp copy

endCopy:

;set the end of CMD

inc si

mov DS:[SI], word ptr '$'

endProc:

pop cx

pop ax

ret

endp getComandLineArgs

;OPENFILE

openFile proc

;ax=0 - ok, ax=1 - error

;DS:DX - fileName

push cx

mov ah, 3Dh

mov al, 0h ;access mode: open for reading

int 21h

jc errorFile ;if CF=1 => opening error

mov descriptor, ax ;copy descriptor

jmp successfulFileOpening

errorFile:

cmp al, 02h

jne errorPath

display mess\_errorFile

jmp errorOpenFile

errorPath:

cmp al, 03h

jne errorManyOpenFiles

display mess\_errorPath

jmp errorOpenFile

errorManyOpenFiles:

mov al, 04h

jne errorNoAccess

display mess\_errorFiles

jmp errorOpenFile

errorNoAccess:

mov al, 05h

jne errorInvalidAccessMode

display mess\_errorAccess

jmp errorOpenFile

errorInvalidAccessMode:

mov al, 0Ch

jne errorOpenFile

display mess\_errorAccessMode

jmp errorOpenFile

errorOpenFile:

mov ax, 1

jmp endProcOpen

successfulFileOpening:

mov ax, 0

display mess\_successfulOpen

jmp endProcOpen

endProcOpen:

pop cx

ret

endp openFile

;READFILE

readFile proc

;ax=1 - error, ax=0 - ok

;bx - descriptor

mov ax, 0

mov ah, 3Fh

mov cx, 50

lea dx, buffer

int 21h

jc errorR

mov cx, ax

mov ax, 0

;display mess\_successfulRead

jmp endProcRead

errorR:

display mess\_errorR

mov ax, 1

jmp endProcRead

endProcRead:

ret

endp readFile

;CLOSEFILE

closeFile proc

;bx - descriptor

mov bx, descriptor

mov ah, 3Eh

int 21h

jnc successfulFileClosing

display mess\_errorDescriptor

mov ax, 1

jmp endProcClose

successfulFileClosing:

mov ax, 0

display mess\_successfulClose

endProcClose:

ret

endp closeFile

;PRINTRESULT

print proc

;ax - number of lines

pusha

xor cx, cx

mov bx, 10

loopf:

xor dx, dx

div bx

push dx ;dx - ostatok ;ax - chastnoe

inc cx

cmp ax, 0

ja loopf

loops:

pop dx

add dx, 30h ;convert to ASCII

mov ah, 02h ;output symbol

int 21h

loop loops ;continue for cx!=0

popa

ret

endp print

scanBuffer proc

lea si, buffer

push ax

scan:

lodsb ;char si -> al and si++

cmp al,13 ;/r

je returnChar

cmp al, 10 ;/n

je newlineChar

cmp al, 20

je next

inc cChars

jmp next

returnChar:

mov rf, 1

jmp next

newlineChar:

cmp rf, 1

jne next

mov rf, 0

cmp cChars, 0

je next

mov cChars, 0

inc cNonEmptyLines

jo overflow

jmp next

overflow:

inc addLines

mov ax, addLines

call print

display mess\_additionalLines

dec cNonEmptyLines ;;;;;;;;;;;;;;

mov ax, cNonEmptyLines

call print

display mess

mov cNonEmptyLines, 0

inc cNonEmptyLines

next:

loop scan ; go scan for cx!=0

cmp cChars, 0

je endProcS

cmp eof, 0

je endProcS

mov cChars, 0

inc cNonEmptyLines

jo overflow

endProcS:

pop ax

ret

endp scanBuffer

numberLines proc

pusha

mov cNonEmptyLines, 0

mov cChars, 0

mov addLines, 0

mov rf, 0

mov eof, 0

mov bx, descriptor

main:

call readFile

cmp ax, 1

je endProcN

cmp cx, SIZE

jb last ;count of reading bytes < SIZE(buffer)

call scanBuffer

jmp main

last:

mov eof, 1

call scanBuffer

jmp endProcN

endProcN:

popa

ret

endp numberLines

end start

Передача имени файла через командную строку представлена на рисунке 1.

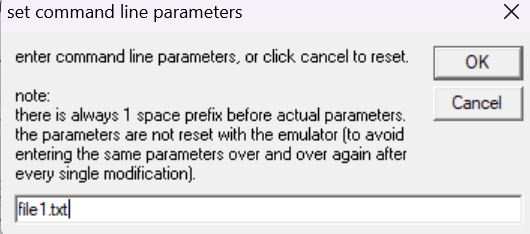


Рисунок 1 – Передача имени файла через командную строку

Содержимое файла file1.txt представлено на рисунке 2.

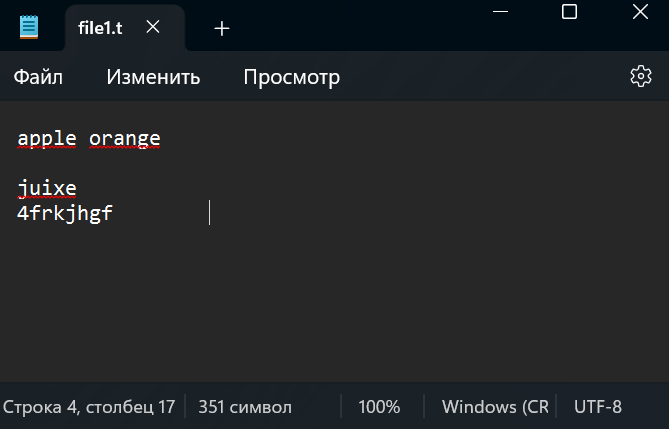


Рисунок 2 – Содержимое файла file1.txt

Результат выполнения программы представлен на рисунке 3.

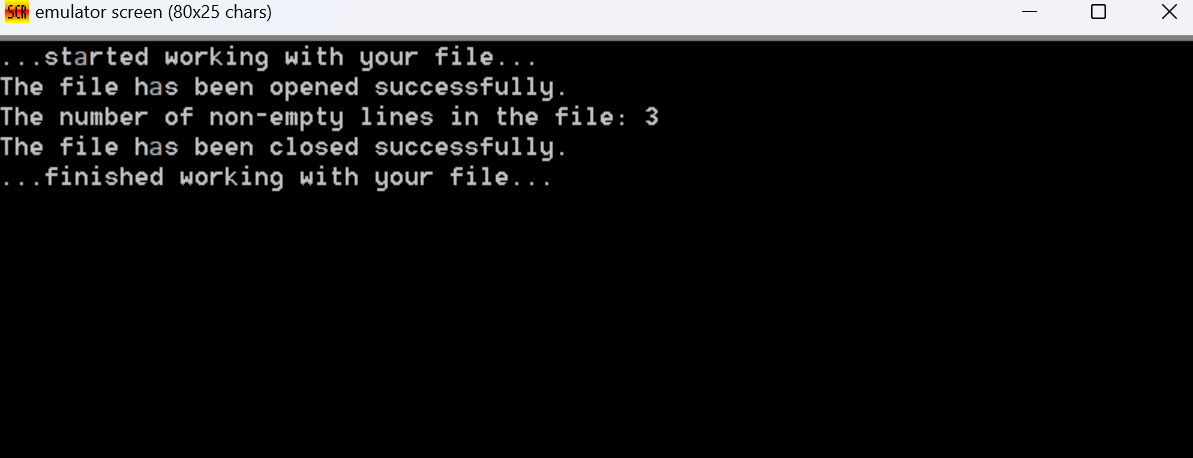


Рисунок 3 – Результат выполнения программы

На рисунке 4 представлен результат выполнения программы с файлом py.txt размером около 1 Кб.

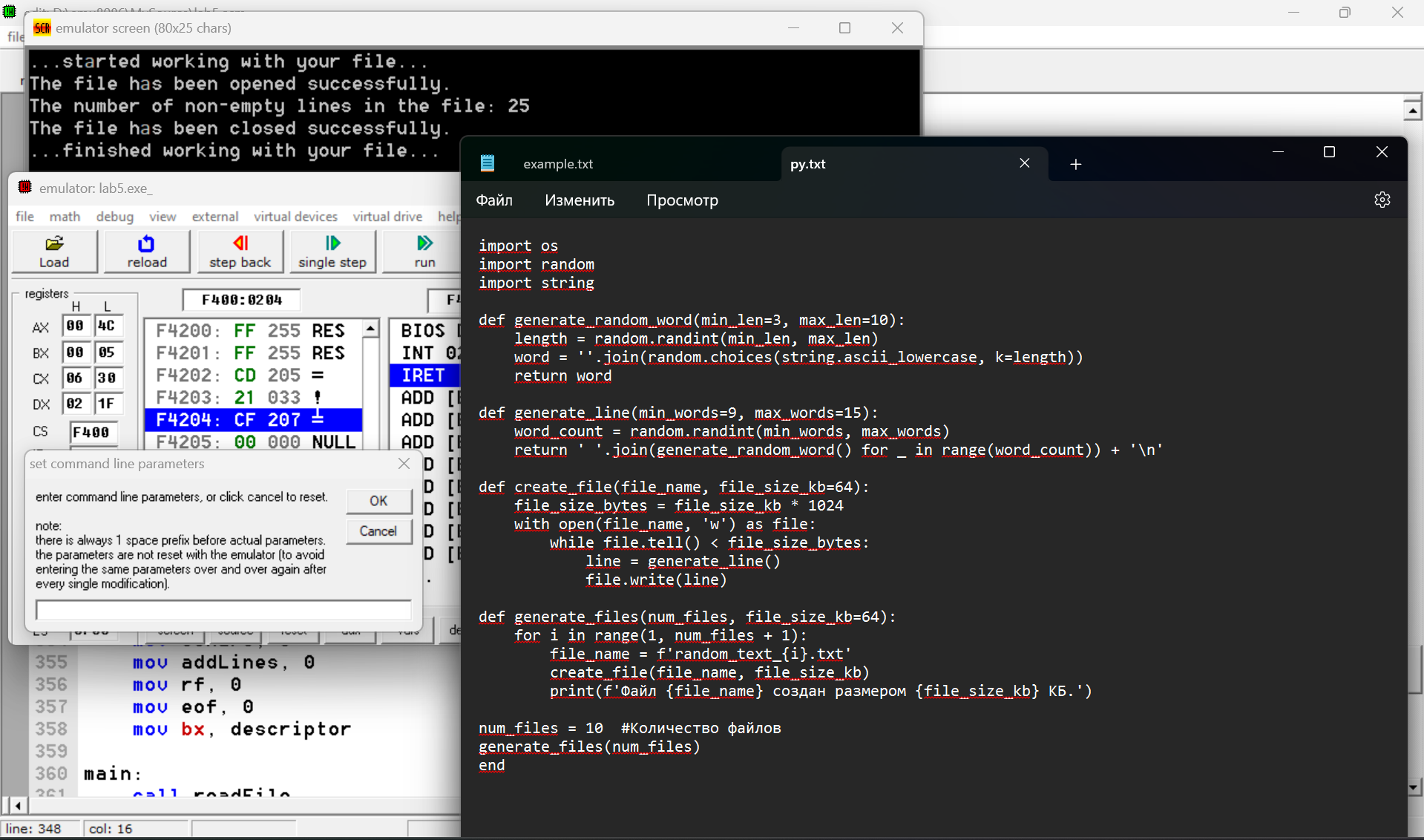


Рисунок 4 – Результат выполнения программы с файлом py.txt

На рисунке 5 представлен результат выполнения программы с файлом размером больше 64 Кб.

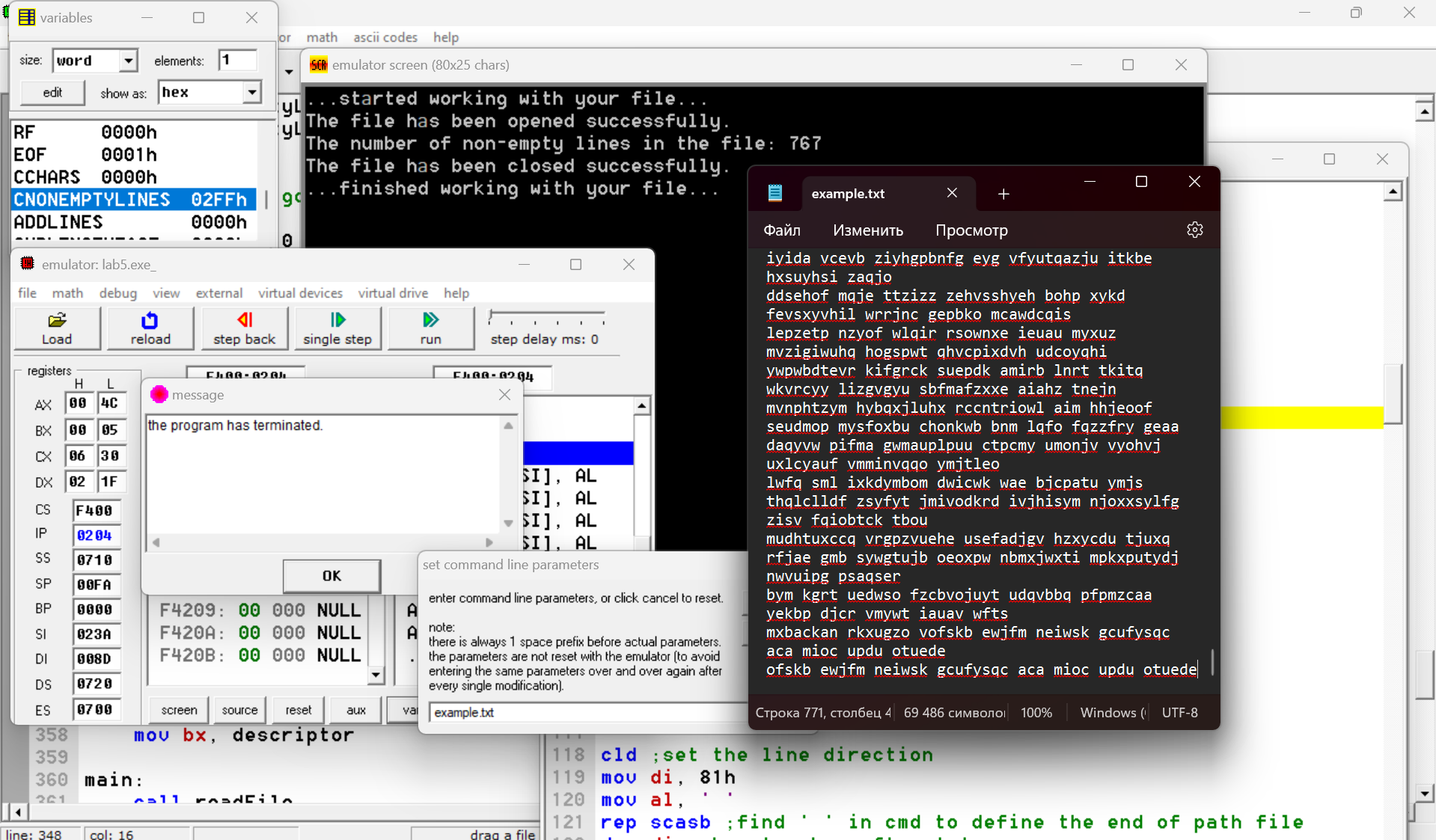


Рисунок 5 – Результат выполнения программы с файлом больше 64 Кб

# Вывод

В ходе лабораторной работы были изучены основные операции, связанные с обработкой файлов, такие как открытие, закрытие, чтение и запись. В процессе выполнения работы было получено понятие о работе с параметрами командной строки, а также изучены основные механизмы обработки файлов в ассемблере.