Федеральное государственное бюджетное образовательное учреждение высшего образования

**Национальный исследовательский университет**

**«МЭИ»**

**Институт информационных и вычислительных технологий**

**Кафедра Вычислительных машин, систем и сетей**

**Отчет по типовому расчету по курсу**

«**СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ**»

Создание программы просмотра произвольного файла в виде дампа

Вариант 22

**Выполнил**: студент гр. А-08-17

Шестёркин В. Д.

**Проверил**: доц. Чернов С. А.

**Дата**:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Оценка**:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись**:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2020

# **Назначение программы. Область ее применения**

Программное приложение, разработанное в рамках данного типового проекта, позволяет изучать дамп, то есть содержание выбранного пользователем файла.

Пользователь видит не только дамп в виде символов и номер строки , но также и содержимое каждого байта в шестнадцатеричной системе счисления .

Программа может оказаться полезной при изучении текстовых файлов.

Минимальный размер файла для исправной работы 65 Байт, максимальный 4ГБ

# **Список использованных файлов. Режимы компиляции и компоновки**

**tip.ASM** – исходный текст программы

**tip.EXE –** сама программа

Исполняемый файл будет иметь расширение EXE, поэтому этапы его компиляции и компоновки будут следующими:

tasm tip.asm

tlink tip.obj

Для запуска исполняемого файла в командной строке MS DOS нужно набрать tip.exe.

При этом на всех трех этапах соблюдение регистра букв необязательно. Так же можно отбросить расширения файлов.

# **Работа программы. Требования к аппаратуре**

Данное программное приложение требует наличия стандартного устройства ввода (клавиатура) и стандартного устройства вывода (дисплей). Каких-то дополнительных периферийных устройств не требуется.

Минимальные требования к ПЭВМ:

* Операционная система MS DOS
* Свободной памяти около 3 Кб
* Дисплей должен поддерживать видеорежим №3 – текст, 80\*25 знакомест.

# **Сценарий работы программы**

При запуске программа устанавливает видеорежим, изменяя его на третий, тем самым очищая видеостраницу. Выводится приглашение к вводу имени файла, дамп которого должен быть выведен.

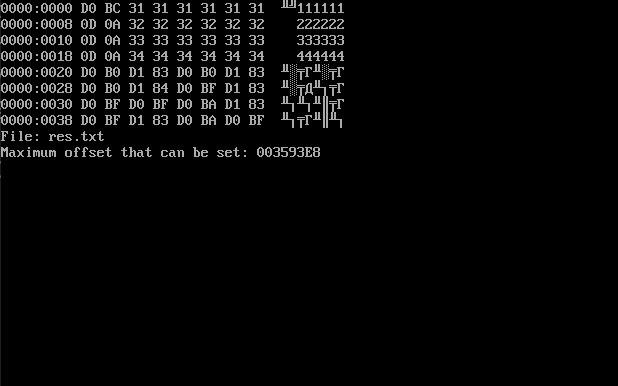
 После получения имени файла, программа выводит 8 строк дампа, имя файла и максимально допустимое смещение, которое можно задать . Далее ожидает действий пользователя: нажатия клавиш меню.

Рис. 1. Окно приложения

# **Интерфейс пользователя**

|  |  |
| --- | --- |
| Клавиша | Действие |
| q | Прокрутка окна дампа вверх на одну строку |
| a | Прокрутка окна дампа вниз на одну строку |
| o | Переход к заданному смещению |
| h | Изменение режима нумерации строк на hex |
| d | Изменение режима нумерации строк на dec (режим рекомендууется использовать для файлов размером не больше 9992 Байт |
| x | Выход из программы |

Табл. 1. Интерфейс пользователя

При нажатии клавиши ‘o’ выводится приглашение к вводу смещения, после чего пользователь вводит 1-8 цифры шестнадцатеричной системы счисления, неверные символы могут быть удалены клавишей Backspace. Символы могут быть цифрами 0-9 и буквами a,b,c,d,e,f, как в верхнем так и в нижнем регистрах

# **Состав программы**

Модуль TIP.ASM содержит логику программу 8 процедур и 3 макроса:

delenie\_DX\_AX – процедура деления DX:AX на 8.

input\_offset – процедура для ввода строки и преобразования ее в число (1-8 символов), принимает на вход цифры и буквы a,b,c,d,e,f как в верхнем так и в нижних регистрах, если был введен неверный символ, то выведет сообщение об ошибке.

izmenenie\_nomera\_stroki– процедура изменения вида нумерации строк (hex/dec).

vivod\_nomera\_stroki– процедура вывода номера строки.

PrintString– процедура вывода строки дампа. Формат: номер строки в формате 0000:0000, 8 символов в hex виде, 8 символов в dec виде.

zamena\_0Ah\_0Dh\_09h– процедура замены символов 0Ah,0Dh,09h на пробел.

Convert\_char\_hex– процедура переводит символ в hex и выводит его.

input– процедура ввода имени файла для дальнейшей работы с ним

vivod\_name\_file– процедура для вывод информации ниже дампа: имя файла и максимальное смешение, которое можно задать.

vivod\_simvola\_10– макрос для вывод символа в dec кодировке.

vivod\_simvola\_hex– макрос для вывод символа в hex кодировке.

polozenie\_kursora– макрос перемещения курсора на необходимую позицию

# **Структура данных программы. Назначение переменных**

**input\_msg db** - приглашение для ввода имени файла

**path db** - имя файла

**buf db** - символ считанный из файла

**nomer\_stroki dw** - младая часть номера строки

**dop\_nomer\_stroki dw** - старшая часть номера строки

**format\_2\_4 dw** - формат вывод чисел(2 или 4)

**skolko\_strok\_viveli db** - количества выведенных строк дампа

**last\_char db** - какая прокрутка была последней(вверх/вниз)

**hex\_dec dw** - формат вывода чисел(hex или dec)

**ostatok\_vivoda dw** - сколько символов осталось вывести в последней строке (если в ней не набралось 8 символов)

**zapret db** - запрещает прокручивать вниз, после достижения конца файла

**oshibka db** - если было некорректно введено смещение тогда 1, иначе 0

**zadali\_offset db** - если задавали смещение - 1. После прокрутки вверх/вниз - 0

**no\_offset db** - выводится, если смещение было задано больше разрешенного

**str1 db** - приграшение ввести смещение

**errmsg db** - выводится, если смещение было задано неккоректно

**namefile db** - выводится перед именем файла

**maxoffset db** - выводится перед максимально разрешенным смещением

**maxlen db** - введенная строка для смещения

**len db**

**string db**

**numberAX dw** - младшая часть введенного смещения

**numberDX dw** - старшая часть введенная смещения

**razmerAX dw** - младшая часть размера файла

- старшая часть размера файла

**Исходный текст программы**

## **tip.asm**

; КУРСОВАЯ РАБОТА ПО КУРСУ "СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ"

; СТУДЕНТА ГРУППЫ А-08-17 ШЕСТЕРКИН В.Д. Вар. 22

; ПРОГРАММА ПРОСМОТРА ПРОИЗВОЛЬНОГО ФАЙЛА В ВИДЕ ДАМПА

; ПРОГРАММА ПОЛУЧАЕТ НА ВХОД ИМЯ ФАЙЛА, ВЫВОДИТ ЕГО ДАМП. ПОЗВОЛЯЕТ ТАКЖЕ ПЕРЕМЕЩАТЬСЯ ПО ДАМПУ, ПЕРЕХОДИТЬ К ЗАДАННОМУ СМЕЩЕНИЮ

; Имя файла исходного текста tip.asm

; Компилятор - TASM, компоновщик - TLINK. OC - MS DOS

; Для прокрутки вверх/вниз нажмите клавиши q/a соответственно

; Для перехода к заданному смещению - клавишу 'o' и введите смещение в виде 1-8 цифр 16-ричной системы

; Для выхода из программы - клавишу x

; Для смены режима отображения смещения нажмите клавишу h/d - hex/dec (режим dec рекомендуется использовать для файлов размером не больше 9992 Байт)

; - Минимальный размер файла для исправной работы 65 Байт, максимальный 4ГБ

; - Программа тестировалась и отлаживалась при работе с файлом, размер котого составляет 3МБ

.model small

.stack 100h

.data

input\_msg db 'Vvedite nazvanie faila:', 13, 10, '$' ; приглашение для ввода имени файла

path db 30,?,29 dup (?),'$' ;имя файла

buf db ? ;символ считанный из файла

nomer\_stroki dw 0h ;младая часть номера строки

dop\_nomer\_stroki dw ;старшая часть номера строки

format\_2\_4 dw 2 ;формат вывод чисел(2 или 4)

skolko\_strok\_viveli db 0 ;количества выведенных строк дампа

last\_char db 'a' ;какая прокрутка была последней(вверх/вниз)

hex\_dec dw 16 ;формат вывода чисел(hex или dec)

ostatok\_vivoda dw ? ;сколько символов осталось вывести в последней строке (если в ней не набралось 8 символов)

zapret db 0 ;запрещает прокручивать вниз, после достижения конца файла

oshibka db 0 ;если было некорректно введено смещение тогда 1, иначе 0

zadali\_offset db 0 ;если задавали смещение - 1. После прокрутки вверх/вниз - 0

no\_offset db "offset too large(press Enter)",0dh,0ah,'$' ;выводится, если смещение было задано больше разрешенного

str1 db "enter the offset(for example: AbcD): ",0dh,0ah,'$' ;приграшение ввести смещение

errmsg db "entered incorrectly(press Enter)",0dh,0ah,'$' ;выводится, если смещение было задано неккоректно

namefile db "File: ",'$' ;выводится перед именем файла

maxoffset db "Maximum offset that can be set: ",'$' ;выводится перед максимально разрешенным смещением

maxlen db 9h ;введенная строка для смещения

len db 0h

string db 8h dup(20h)

numberAX dw 0000h ;младшая часть введенного смещения

numberDX dw 0000h ;старшая часть введенная смещения

razmerAX dw 0000h ;младшая часть размера файла

razmerDX dw 0000h ;старшая часть размера файла

.code

.386 ;используется исключительно для снятия ограничения с условных джампов

vivod\_simvola\_10 MACRO nomer ;макрос для вывод символа в dec кодировке

mov dx,[bp+nomer] ;параметры передаются через стек

call zamena\_0Ah\_0Dh\_09h ;замена символов 0Ah, 0Dh, 09h на пробелы для вывода в dec виде

call proc\_2\_21h ;процедура вывода символа

endm

vivod\_simvola\_hex MACRO nomer ;макрос для вывод символа в hex кодировке

mov ax,[bp+nomer] ;параметры передаются через стек

call Convert\_char\_hex ;переводит символ в hex вид и выводит его в заданном формате(2 или 4 символа)

mov dx,20h ;вывод пробела

call proc\_2\_21h

endm

polozenie\_kursora MACRO stroka ;макрос перемещения курсора на необходимую позицию

mov ah, 2 ;будет вызвана функция 2h int 10h - установка курсора

mov bh, 0 ;номер видеостраницы

mov dh, stroka ;номер строки

mov dl, 0 ;номер столбца

int 10h ;прерывание

endm

begin:

mov ax,@data

mov DS,ax ;настраиваем сегментный регистр данных

mov ax, 03 ;очищам окно после входа в программу

int 10h

call input ;ввод имени файла

mov ax, 03 ;очищаем окно для корректного вывода

int 10h

mov ah,01h ;выбор режима курсора

mov ch,20h ;курсор будет невидимым

int 10h

mov ax,3d00h ;открываем для чтения

lea dx,path + 2 ;DS:dx указатель на имя файла

int 21h ;в ax деcкриптор файла

jc exit ;если поднят флаг С, то ошибка открытия

push ax ;запоминаем указатель файла для дальнейшей корректной работы

mov bx,ax

mov ax,4202h ;получение размера файла

xor cx,cx

xor dx,dx

int 21h ;DX:AX - размер файла

mov razmerDX,dx ;запоминаем старшую часть размера файла

mov razmerAX,ax ;запоминаем младшую часть размера файла

call delenie\_DX\_AX ;процедура деления DX:AX на 8, сx:ax находится частное, в dx - остаток от деления

XCHG cx,dx ;меняем местами значения регистров

mov ax,razmerAX

mov dx,razmerDX ;вычитаем 38h + остаток от деления для ограничения задаваемого смещения

sub ax,cx

sub ax,38h

sbb dx,0

mov razmerAX,ax ;запоминаем исправленный размер файла

mov razmerDX,dx

call vivod\_name\_file ;выводим на экран имя файла и максимально разрешенное смещение в нем

pop bx ;копируем в bx указатель файла

xor cx,cx

xor dx,dx

mov ax,4200h

int 21h ;идем к началу файла

xor di,di

;==============================================================================

; Побайтовое чтение из файла и вывод строки (после прочтения 8 байт)

;==============================================================================

out\_str:

xor ax,ax

mov ah,3fh ;будем читать из файла

mov cx,1 ;1 байт

lea dx,buf ;в память buf

int 21h

cmp ax,cx ;если достигнуть EoF

jnz finish\_file ;то выводим последнюю строку (если в ней не набралось 8 символов) и запрещаем прокрутку вниз

mov dl,buf

push dx ;помещаем символ в стек для передачи его в процедуру вывод строки

add di,1 ;считаем символы в строке (до 8)

cmp di,8

JNZ out\_str ;продолжаем считывать символы, если не 8

call PrintString ;если символов в строке набралось 8, то передаем управление процедуре вывода строки

xor di,di ;обляем счетчик символов в строке

mov al,skolko\_strok\_viveli

add al,1

cmp al,8 ;строк дампа на экране должно быть 8

jz waiting ;после вывод 8-ой строки переходим на метку для ожидания нажатия клавиши

mov skolko\_strok\_viveli,al

jmp out\_str ;если строк на экране не 8, считываем из файла дальше

;==============================================================================

; Ожидаем нажатие клавиши для дальнейших действий

;==============================================================================

waiting:

mov skolko\_strok\_viveli,7 ;можно будет вывести еще одну строчки при прокрутке вверх/вниз

mov oshibka,0 ;обнуляем ошибку для дальнейших попыток ввода

mov ah, 08h ;вводим символ без эха

int 21h

cmp al,'q'

JZ proverka\_up ;при нажатии q переходим на метку прокрутки вверх

cmp al,'a'

JZ proverka\_down ;при нажатии a переходим на метку прокрутки вниз

cmp al,'x'

JZ close ;при нажатии x переходим на выхода из программы

cmp al,'h'

JZ nomer\_v\_hex ;при нажатии h меняем формат вывода номера строк на hec

cmp al,'d'

JZ nomer\_v\_dec ;при нажатии d меняем формат вывода номера строк на dec

cmp al,'o'

JZ zadat\_smeshc ;при нажатии o задаем смещение дампа

jmp waiting ;ждем пока не введен нужный символ

;==============================================================================

; Изменение смещения дампа

;==============================================================================

zadat\_smeshc:

push bx ;запоминаем указатель файла

mov zadali\_offset,1 ;запоминаем, что задавали смещение для дальнейшей корректной работы программы

call input\_offset ;процедура ввода смещения

xor ax,ax

mov al,oshibka

cmp al,1

jz nazali\_enter1 ;если смещение ведено с ошибкой, то выводим файл сначала и возвращаемся в режим ожидания

xor cx,cx

mov cl,zapret

cmp cl,0 ;проверяем, дошли до конца файла или нет

jz esli\_ne\_zapret1

mov zapret,0 ;если дошли до конца файла, то необходимо скорректировать смещение в файле

mov ax,4201h

mov dx,ostatok\_vivoda ;корректируем на количество байт, которое выводили после достижения конца файла(<8)

xor cx,cx

int 21h

esli\_ne\_zapret1:

mov ax,razmerDX

mov dx,numberDX

cmp dx,ax

JA big\_offset ;сравниваем введенное смещение и максимально допустимое, если больше, то выводим файл сначала и возврщаемся в режим ожидания

cmp ax,0

jnz suda ;если razmerDX не равен нулю, то дополнительных проверок не требуется

mov ax,razmerAX

mov dx,numberAX

cmp dx,ax

JA big\_offset ;иначем сравниваем младшие части введеного смещения и максимально допустимого

suda: ;если введенное смещение допустимо, то продолжаем

call vivod\_name\_file ;выводим на экран имя файла и максимально разрешенное смещение в нем

mov ah,01h ;выбор режима курсора

mov ch,20h ;курсор будет невидимым

int 10h

xor di,di ;обнуляем количетсво символов встроке

mov skolko\_strok\_viveli,0 ;обнуляем количетсво выведенных строк

mov ax,numberAX

mov dx,numberDX

mov nomer\_stroki,ax

mov dop\_nomer\_stroki,dx ;теперь смещение дампа = заданному смещению

polozenie\_kursora 0 ;переносим курсор на 0-ю строку

pop bx ;помещаем в bx указатель файла

xor cx,cx

xor dx,dx

mov ax,4200h

int 21h ;идем к началу файла

dviz: ;смещаем указатель файла от начала до введенного смещения

mov ax,numberAX

mov dx,numberDX

cmp dx,0

jnz proveryaem\_skolko\_smestitsya ;если старшая часть смещения не ровна 0, то сначала вычитаем из нее

vichitaem:

cmp ax,0 ;0, если дошли до нужного смещения

jz out\_str ;тогда начинаем выводить с заданного смещения

sub ax,8

sbb dx,0

mov numberAX,ax

mov numberDX,dx ;иначе продолжаем вычитать

mov ax,4201h ;переместить указатель файла от текущей позиции

mov dx,8 ;на 8 байт вперед

xor cx,cx ;0 т.к. смещение положительное

int 21h

jmp dviz ;продолжаем, пока не дойдем до заданного смещения

proveryaem\_skolko\_smestitsya: ;если старшая часть смещения не ровна 0, то сначала вычитаем из нее

sub ax,0fff0h

sbb dx,0 ;вычитаем знаение на которое будем смещать указатель в файле

mov numberAX,ax

mov numberDX,dx

mov ax,4201h ;переместить указатель файла от текущей позиции

mov dx,0fff0h ;на fff0h байт вперед

xor cx,cx ;0 т.к. смещение положительное

int 21h

jmp dviz ;продолжаем, пока не дойдем до заданного смещения

big\_offset: ;если ввели смещение больше допустмого, то выводим файл с начала и переходим в ожидание

mov ah,9h

lea dx,no\_offset

int 21h ;выводим сообщение, что смещение превышает допустиоме

mov ah,01h ;выбор режима курсора

mov ch,20h ;курсор будет невидимым

int 10h

ozidanie1:

mov ah, 08h

int 21h

cmp al,0DH

JZ nazali\_enter1

jmp ozidanie1

nazali\_enter1: ;ожидаем нажатие Enter

xor di,di ;обнуляем счетчик символов в строке

mov skolko\_strok\_viveli,0 ;обнуляем счетчик строк дампа на экране

mov nomer\_stroki,0

mov dop\_nomer\_stroki,0 ;будет выводить файл сначала

polozenie\_kursora 0 ;переносим курсор на 0-ю строку

pop bx ;восстанавливаем указатель файла

mov ax,4201h ;переместить указатель файла от текущей позиции

mov dx,-8 ;на 8 байт назад

mov cx,-1 ;т.к. смещение отрицательное

int 21h

xor cx,cx

xor dx,dx

mov ax,4200h

int 21h ;идем к началу файла

call vivod\_name\_file ;выводим на экран имя файла и максимально разрешенное смещение в нем

jmp out\_str ;переходим к считываю и выводу из файла

;==============================================================================

; изменение режима отображения нумерации строк (hex/dec)

;==============================================================================

nomer\_v\_hex: ;изменение режима отображания нумерации строк в hex

mov hex\_dec,16 ;помещаем в переменную режим отображения

call izmenenie\_nomera\_stroki;передаем управление функции для изменения режима на видимом дампе

jmp waiting ;возвращаемся в режим ожидания

nomer\_v\_dec: ;изменение режима отображания нумерации строк в dec

mov hex\_dec,10 ;помещаем в переменную режим отображения

call izmenenie\_nomera\_stroki;передаем управление функции для изменения режима на видимом дампе

jmp waiting ;возвращаемся в режим ожидания

;==============================================================================

; проверка и корректировка смещения при прокрутке вверх

;==============================================================================

proverka\_up: ;при нажатии на q в режиме ожидания переходим сюда(прокрутка вверх)

xor cx,cx

mov cl,zapret

cmp cl,0 ;проверяем, дошли до конца файла или нет

jz esli\_ne\_zapret

mov zapret,0 ;если дошли до конца файла, то необходимо скорректировать смещение в файле

mov ax,4201h

mov dx,ostatok\_vivoda ;корректируем на количество байт, которое выводили после достижения конца файла(<8)

xor cx,cx

int 21h

esli\_ne\_zapret:

cmp al,last\_char ;необходимо проверить, какая прокрутка была последней(вверх/вниз)

JZ up ;если последняя прокрутка также была вверх, то дополнительно ничего смещать не надо

mov cl,zadali\_offset ;если последним действием мы задавали смещение(клавиша o),

cmp cl,0 ;то необходимо установить в переменной 0, для дальнейшей корректной работы

jz offset\_ne\_zadavali3

mov zadali\_offset,0

offset\_ne\_zadavali3:

mov ax,nomer\_stroki ;если последния прокрутка была вниз, то необходимо сделать дополнительные действия

sub ax,56

mov nomer\_stroki,ax ;вычитаем из номера строки 56, чтобы при прокрутке вверх, нумерация отображалось верно

mov ax,4201h ;переместить указатель файла от текущей позиции

mov dx,-56 ;на 56 байт назад

mov cx,-1 ;-1 т.к. вычитаем

int 21h

mov last\_char,'q' ;запоминаем, что последним дейсвтием будет прокрутка вверх

jmp up ;переходим к прокрутке вверх и выводу соответствующей строки

;==============================================================================

; проверка и корректировка смещения при прокрутке вниз

;==============================================================================

proverka\_down: ;при нажатии на a в режиме ожидания переходим сюда(прокрутка вниз)

xor cx,cx

mov cl,zapret ;проверяем, дошли до конца файла или нет

cmp cl,1 ;если дошли до конца файла, но не делали прокрутку вверх, то прокрутка вниз запрещена.

jz waiting ;возврат в режим ожижания

cmp al,last\_char ;необходимо проверить, какая прокрутка была последней(вверх/вниз)

JZ down ;если последняя прокрутка также была вниз, то дополнительно ничего смещать не надо

mov cl,zadali\_offset ;если последним действием мы задавали смещение(клавиша o), то необходимо скорректировать смещения

cmp cl,0

jz offset\_ne\_zadavali2

mov ax,4201h ;переместить указатель файла от текущей позиции

mov dx,-56 ;на 56 байт назад

mov cx,-1 ;-1 т.к. вычитаем

int 21h

mov dx,dop\_nomer\_stroki

mov ax,nomer\_stroki

sub ax,38h ;уменьшить номер строк на 38

sbb dx,0

mov nomer\_stroki,ax

mov dop\_nomer\_stroki,dx

mov zadali\_offset,0

offset\_ne\_zadavali2: ;если последним действием мы не задавали смещение, то переходим сразу сюда

mov ax,nomer\_stroki ;если последния прокрутка была вврех, то необходимо сделать дополнительные действия

add ax,56

mov nomer\_stroki,ax ;добавляем к номеру строки 56, чтобы при прокрутке вниз, нумерация отображалось верно

mov ax,4201h ;переместить указатель файла от текущей позиции

mov dx,56 ;на 56 байт вперед

xor cx,cx ;0, т.к. прибаляем

int 21h

mov last\_char,'a' ;запоминаем, что последним дейсвтием будет прокрутка вниз

jmp down ;переходим к прокрутке вниз и выводу соответствующей строки

;==============================================================================

; прокрутка вверх

;==============================================================================

up: ;метка с которой начинается прокрутка вверх и вывод соответсвующей строки

cmp nomer\_stroki,8

JnZ mozno\_vichitat

cmp dop\_nomer\_stroki,0 ;если номер строки равен 8, то последняя строка выводилась с номером 0, вычитать больше нельзя

jz waiting ;возвращаемся в режим ожидания

mozno\_vichitat:

mov cl,zadali\_offset

cmp cl,0

jz offset\_ne\_zadavali ;если последним действием мы задавали смещение(клавиша o), то необходимо скорректировать смещения

mov ax,4201h ;переместить указатель файла от текущей позиции

mov dx,-56 ;на 56 байт назад

mov cx,-1 ;-1 т.к. вычитаем

int 21h

mov dx,dop\_nomer\_stroki

mov ax,nomer\_stroki

sub ax,38h ;уменьшить номер строк на 38

sbb dx,0

mov nomer\_stroki,ax

mov dop\_nomer\_stroki,dx

mov zadali\_offset,0

offset\_ne\_zadavali: ;если последним действием мы не задавали смещение, то переходим сразу сюда

mov dx,dop\_nomer\_stroki

mov ax,nomer\_stroki

sub ax,16 ;вычитаем 16 из номера строки

sbb dx,0

mov nomer\_stroki,ax

mov dop\_nomer\_stroki,dx

mov ah,7

mov al,1

mov bh,7

mov ch,0

mov cl,0

mov dh,7

mov dl,43

int 10h ;делаем прокрутку вверх

polozenie\_kursora 0 ;переносим курсор на 0-ю строку

mov ax,4201h ;переместить указатель файла от текущей позиции

mov dx,-16 ;на 16 байт назад

mov cx,-1 ;-1 т.к. вычитаем

int 21h

jmp out\_str ;переходим к считываю и выводу из файла

;==============================================================================

; прокрутка вниз

;==============================================================================

down: ;метка с которой начинается прокрутка вниз и вывод соответсвующей строки

mov ah,6

mov al,1

mov bh,7

mov ch,0

mov cl,0

mov dh,7

mov dl,43

int 10h ;делаем прокрутку вниз

polozenie\_kursora 7 ;переносим курсор на 7-ю строку

jmp out\_str ;переходим к считываю и выводу из файла

;==============================================================================

; дошли до конца файла

;==============================================================================

finish\_file: ;если в процессе считывания из файла, мы дошли до его конца, то переходим сюда

mov zapret,1 ;запрещаем делать прокрутку вниз, пока не сделаем прокрутку вверх или не зададим смещение

cmp di,0

jz waiting ;если достигли конца файла, но в стеке нет символов, то возвращаемся в режим ожидания

mov cx,8

sub cx,di ;иначе узнаем сколько символов не хватило до полной строки (полная строка 8 символов)

mov ostatok\_vivoda,cx;

rr:

push ''

loop rr ;и дополняем ее пустыми символами

xor di,di ;обнуляем счетчик символов в строке

call PrintString ;передаем управление процедуре вывода строки

jmp waiting ;возвращаемся в режим ожидания

;==============================================================================

; ПРОЦЕДУРЫ

;==============================================================================

delenie\_DX\_AX proc ;процедура деления DX:AX на 8

xor cx, cx

mov bx,8

cmp dx, bx ;старшая часть делимого меньше частного - переполнения не будет

jbe div2

mov cx, ax ;сохраняем младшую часть

mov ax, dx

xor dx, dx

div bx

xchg cx, ax ;AX = младшая часть делимого, CX = старшая часть частного

;В DX остался остаток от деления

div2:

div bx

done: ;в сx:ax находится частное, в dx - остаток от деления

ret

endp

input\_offset proc ;процедура для ввода смещения(1-8 символов)

push si

push bx

mov ax, 03 ;очищаем экран

int 10h

polozenie\_kursora 0 ;переносим курсор на 0-ю строку

MOV AH,01 ;Установить размер курсора

MOV CH,6 ;Верхняя линия сканирования

MOV CL,7 ;Нижняя линия сканирования

INT 10H ;Вызвать BIOS

;Приглашение

mov ah,9h

lea dx,str1

int 21h

;Ввод числа

mov ah,0ah

lea dx,maxlen

int 21h

xor dx,dx

;Подготовка к циклу

xor ax,ax ;обнуляется регистр

lea di,string ;di - индексный регистр

mov si,16 ;si содержит множитель 16, т.к. надо получить число в формате hex

xor bh,bh ;обнуляется регистр

xor ch,ch ;обнуляется регистр

mov cl,len ;Число цифр в буфере

cmp cl,4

ja snachalaDX ;если длина введенной строки превышает 4 символа, то переходим

m1:

mul si ;умножить ax на si(16)

mov bl,[di] ;к произвдению добавить число

cmp bl, 30h ;сравнение

jl err1 ;если меньше, то введен недопустимый символ

cmp bl, 39h ;сравнение

jg bukvaAX ;если больше, то переходим на метку(возможно веденный символ - A,B,C,D,E,F,a,b,c,d,e,f)

sub bl,30h ;иначе цифра, отнимаем 30h

goAX:

add ax,bx ;добачить число к сумме ax

inc di ;инкремент di

loop m1 ;повтор цикла

mov numberAX,ax ;переместить регистр ax numberAX

mov ax,numberAX

mov dx,numberDX

delim\_dalshe: ;вычитаем из веденного смещения, пока он не кратен 8

mov numberDX,dx

mov numberAX,ax

call delenie\_DX\_AX

cmp dx,0

jz bez\_ostatka

mov dx,numberDX

mov ax,numberAX

sub ax,1

jmp delim\_dalshe

bez\_ostatka: ;если кратно 8, то восстанавливаем все регистры и выходим из процедуры

pop bx

pop si

xor di,di

ret

snachalaDX: ;если длинна веденного смещения превышает 4 символа, то сначала помещаем их в numberDX

sub cl,4 ;вычитаем 4, чтобы узнать сколько символов из веденной строки помещать в numberDX

mDX:

mul si ;умножить ax на si(16)

mov bl,[di] ;к произвдению добавить число

cmp bl, 30h ;сравнение

jl err1 ;если меньше, то введен недопустимый символ

cmp bl, 39h ;сравнение

jg bukvaDX ;если больше, то переходим на метку(возможно веденный символ - A,B,C,D,E,F,a,b,c,d,e,f)

sub bl,30h ;иначе цифра, отнимаем 30h

goDX:

add ax,bx ;добачить число к сумме ax

inc di ;инкремент di

loop mDX ;повтор цикла

mov numberDX,ax ;переместить регистр ax numberDX

xor ax,ax ;очищаем ax, для numberAX

mov cl,4 ;обрабатываем оставшиеся 4 символа

jmp m1

bukvaDX: ;если символ - A,B,C,D,E,F для numberDX

cmp bl, 65

jl neZaglavDX

cmp bl, 70

jg neZaglavDX

sub bl,55

jmp goDX

neZaglavDX: ;если символ - a,b,c,d,e,f для numberDX

cmp bl, 97

jl err1

cmp bl, 102

jg err1

sub bl,87

jmp goDX

bukvaAX: ;если символ - A,B,C,D,E,F для numberAX

cmp bl, 65

jl neZaglavAX

cmp bl, 70

jg neZaglavAX

sub bl,55

jmp goAX

neZaglavAX: ;если символ - a,b,c,d,e,f для numberAX

cmp bl, 97

jl err1

cmp bl, 102

jg err1

sub bl,87

jmp goAX

err1: ;если введен некорректный символ

pop bx ;восстанавливаем регистры

pop si

xor di,di

mov ah,9h ;выводим сообщение об ошибке

lea dx,errmsg

int 21h

mov ah,01h ;выбор режима курсора

mov ch,20h ;курсор будет невидимым

int 10h

mov oshibka,1 ;запоминаем, что была ошибка

ozidanie: ;ждем нажатия Enter

mov ah, 08h

int 21h

cmp al,0DH

JZ nazali\_enter

jmp ozidanie

nazali\_enter:

ret

endp

izmenenie\_nomera\_stroki proc;процедура изменения формата выводу нумерации строк(hex/dec)

xor cx,cx

mov cl,last\_char

cmp cl,'a'

jz was\_last\_char1 ;если последняя прокрутка была вниз, то переходим

mov dx,dop\_nomer\_stroki

mov ax,nomer\_stroki

add ax,56 ;иначе необходимо скорректировать номера строк

adc dx,0

mov nomer\_stroki,ax

mov dop\_nomer\_stroki,dx

was\_last\_char1:

polozenie\_kursora 0 ;переносим курсор на 0-ю строку

mov skolko\_strok\_viveli,0 ;обнуляем счетчик вывыденных строк

mov dx,dop\_nomer\_stroki

mov ax,nomer\_stroki

sub ax,64 ;вычитаем 64 из номера, т.к. выводим все номера на экране

sbb dx,0

mov nomer\_stroki,ax

mov dop\_nomer\_stroki,dx

mov cx,8 ;выводим 8 строк (номеров)

m2:

call vivod\_nomera\_stroki

mov dx,0Ah

call proc\_2\_21h

mov dx,0Dh

call proc\_2\_21h

loop m2

xor cx,cx

mov cl,last\_char

cmp cl,'a' ;если последняя прокрутка была вниз, то переходим

jz was\_last\_char2

mov dx,dop\_nomer\_stroki

mov ax,nomer\_stroki

sub ax,56 ;иначе необходимо скорректировать номера строк обратно

sbb dx,0

mov nomer\_stroki,ax

mov dop\_nomer\_stroki,dx

was\_last\_char2:

ret

endp

vivod\_nomera\_stroki proc ;процедура вывода номера строки

mov format\_2\_4, 4 ;выводятся в формате 0000:0000

mov ax,dop\_nomer\_stroki

call Convert\_char\_hex

mov dx,':'

call proc\_2\_21h

mov ax,nomer\_stroki

call Convert\_char\_hex

xor dx,dx

mov ax,nomer\_stroki

add ax,0008h ;прибаляем 8, чтобы следующий номер строки отличался

adc dx,0

mov nomer\_stroki,ax

mov ax,dop\_nomer\_stroki

add ax,dx

mov dop\_nomer\_stroki,ax

mov dx,20h

call proc\_2\_21h

ret

endp

PrintString PROC ;процедура вывода строки

push bp ;подготовка для передачи параметров через стек

mov bp,sp

call vivod\_nomera\_stroki ;выводим номер строки

mov cx,hex\_dec ;вывод символов в hex

mov hex\_dec,16

mov format\_2\_4, 2 ;по 2 символа

vivod\_simvola\_hex 18 ;выводим первый символ в строке в hex

vivod\_simvola\_hex 16

vivod\_simvola\_hex 14

vivod\_simvola\_hex 12

vivod\_simvola\_hex 10

vivod\_simvola\_hex 8

vivod\_simvola\_hex 6

vivod\_simvola\_hex 4 ;выводим восьмой символ в строке в hex

mov dx,20h

call proc\_2\_21h

;mov hex\_dec,cx

vivod\_simvola\_10 18 ;выводим первый символ в строке в dec

vivod\_simvola\_10 16

vivod\_simvola\_10 14

vivod\_simvola\_10 12

vivod\_simvola\_10 10

vivod\_simvola\_10 8

vivod\_simvola\_10 6

vivod\_simvola\_10 4 ;выводим восьмой символ в строке в dec

mov dx,0Ah ;переходим на следующую строчку

call proc\_2\_21h

mov dx,0Dh

call proc\_2\_21h

pop bp

ret 16 ;удаляем 8 элементов в стека и выходим из процедуры

endp

proc\_2\_21h PROC ;процедура вывод символа

mov ah,02h

int 21h

ret

endp

zamena\_0Ah\_0Dh\_09h proc ;процедура замены символов 0Ah,0Dh,09h на пробел

cmp dx,0Ah

JNZ perenos\_probel1

mov dx,' '

perenos\_probel1:

cmp dx,0Dh

JNZ perenos\_probel2

mov dx,' '

perenos\_probel2:

cmp dx,09h

JNZ perenos\_probel3

mov dx,' '

perenos\_probel3:

ret

endp

Convert\_char\_hex proc ;процедура перевода в hex символа и его вывод в дамп

push dx

push bx

push cx

xor dx, dx ; зануляем ргистр

mov bx,hex\_dec ;система счисления

mov cx, format\_2\_4 ;формат вывода 2 или 4 символа

vivod:

; В DX остался остаток от деления

div bx ;число поделили на систему счисления

;Если ЧИСЛО - это БАЙТ, то AL = AX / ЧИСЛО

;Если ЧИСЛО - это СЛОВО, то AX = (DX AX) / ЧИСЛО

;При этом остаток от деления, если таковой имеется, будет записан:

;В регистр АН, если ЧИСЛО - это байт

;В регистр DX, если ЧИСЛО - это слово

push dx ;заносим остаток от деления (цифру десятичной записи) в стек

xor dx,dx

loop vivod ;если да, то продолжаем разбор числа на цифры

mov ah,02h ;вывод

mov cx, format\_2\_4

vivodvkons:

pop dx ;достает цифры из стека последовательно

cmp dl,10

jl next

add dl,7

next:

add dl, '0' ;добавляем смещение на 30h

int 21h

loop vivodvkons

pop cx

pop bx

pop dx

ret

endp

input proc ;процедура ввода имени файла для дальнейшей работы с ним

mov ah,09h

mov dx,offset input\_msg ;выводим приглашение

int 21h

mov ah,0Ah

mov dx,offset path ;вводим имя файла

int 21h

xor bx,bx

mov bl,path+1

mov byte ptr path[bx+2],0 ;добавляем в конец имени 0, для успешного открытия файла

mov ax, 03 ;очищаем экран

int 10h

ret

endp

vivod\_name\_file proc

polozenie\_kursora 8 ;переносим курсор на 8-ю строку

mov ah,09h

mov dx,offset namefile

int 21h

mov ah,09h

mov dx,offset path + 2 ;выводим имя файла

int 21h

mov dx,0Ah

call proc\_2\_21h

mov dx,0Dh

call proc\_2\_21h

mov ah,09h

mov dx,offset maxoffset

int 21h

mov format\_2\_4,4 ;выводим максимально дупустимое смещение в файле, которое можно задать в формате 0000 0000

mov ax,razmerDX

call Convert\_char\_hex

mov ax,razmerAX

call Convert\_char\_hex

mov format\_2\_4,2

polozenie\_kursora 0 ;переносим курсор на 8-ю строку

ret

endp

close: ;закрываем файл, после чтения

xor ax,ax

mov ah,3eh

int 21h

MOV AH,01 ;Установить размер курсора

MOV CH,6 ;Верхняя линия сканирования

MOV CL,7 ;Нижняя линия сканирования

INT 10H ;Вызвать BIOS

exit: ;завершаем программу

mov ax,3

int 10h

xor ax,ax

mov ah,4ch

int 21h

end begin

# **Заключение**

Данное программное приложение отлажено и работоспособно в установленных данной спецификации пределах. Программа тестировалась на текстовом файле 3Мб и файле asm. В процессе отладки основные проблемы возникали со смещением в файле и нумерации строк после определенных действий, но все проблемы были выловлены и исправлены с помощью дополнительных условий и проверок, которые реализованы в алгоритме программы. Минимальный размер файла, который можно корректно просматривать в программе равен 65 Байт, максимальный 4Гб. Так же ограничение наложено для вывода нумерации строк в виде dec, в таком виде максимальный размер файла рекомендуется использовать 9992 Байта, иначе нумерация строк будет некорректной. Так же в процессе отладки был ускорен переход на большие смещения.

# **Оглавление**

[**Назначение программы. Область ее применения** 2](#_Toc39950304)

[**Список использованных файлов. Режимы компиляции и компоновки** 2](#_Toc39950305)

[**Работа программы. Требования к аппаратуре** 3](#_Toc39950306)

[**Сценарий работы программы** 3](#_Toc39950307)

[**Интерфейс пользователя** 4](#_Toc39950308)

[**Состав программы** 4](#_Toc39950309)

[**Структура данных программы. Назначение переменных** 5](#_Toc39950310)

[**tip.asm** 6](#_Toc39950311)

[**Заключение** 38](#_Toc39950312)

[**Оглавление** 39](#_Toc39950313)

[**Литература** 40](#_Toc39950314)

# **Литература**

1. Абель П. Язык ассемблера для IBM PC и программирования. – М.:Высш. шк., 1992. – 447 c.
2. Лю Ю-Чжен, Гибсон Г. Микропроцессоры семейства 8086/8088. Архитектура, программирование и проектирование микрокомпьютерных систем: Пер. с англ. – М.: Радио и связь, 1987. – 512 с.
3. Юров В. И. Assembler. – СПб.: Питер, 2003. – 624 с.
4. Калашников О. А. Ассемблер – это просто. – 2-е изд., перераб. и доп. – СПб: БХВ-Петербург, 2011. – 336 с.