Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 8

по дисциплине «Программирование на языке Ассемблера»

Вариант 2

Выполнил студент гр. 150502: Альхимович Н.Г.

Проверил:        Туровец Н.О.

Минск 2022

Цель работы:

Получить понятие об обработчиках прерываний и изучить осо- бенности создания резидентных программ.

Вариант задания:

Написать программу вывода строк текста из файла с обработчиком прерывания 5 (Print-Screen). При нажатии Print-Screen программа должна записать содержимое экрана в файл print.txt.

Теоретические сведения:

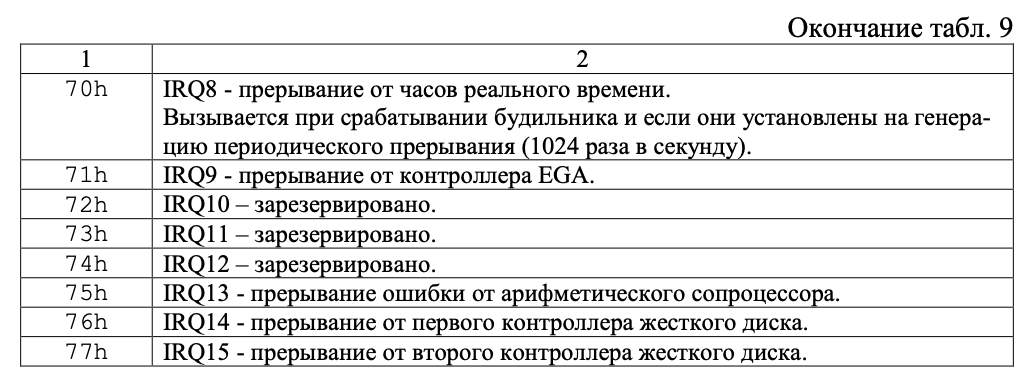
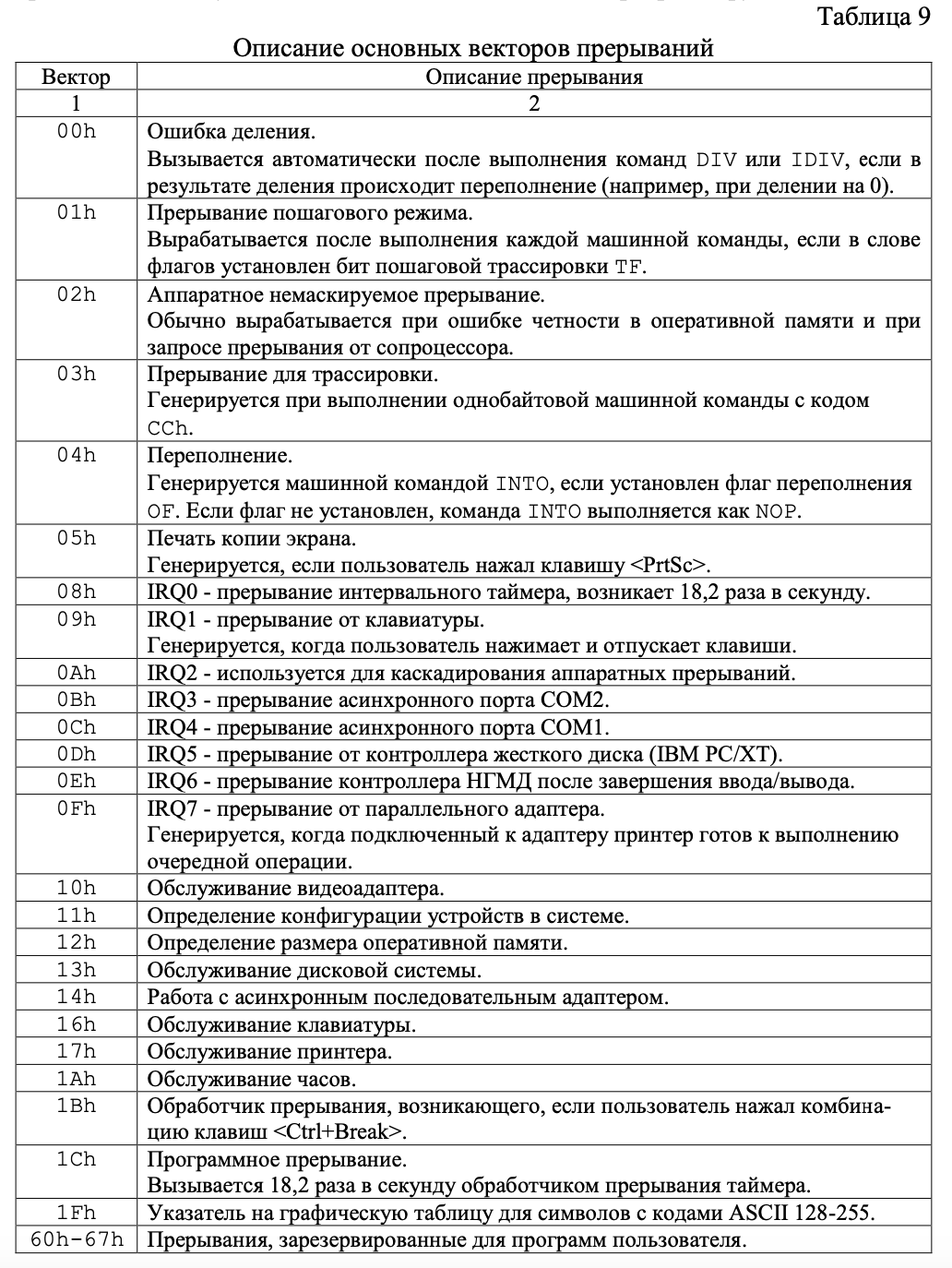
1. Обработчики прерываний.

Для обработки событий, особенно происходящих асинхронно по отноше- нию к выполнению программы, лучше всего подходит механизм прерываний – особых событий в системе, которые требуют немедленной обработки.

С каждым прерыванием связано уникальное число – номер прерывания, соответствующий определенному событию, а также соответствующий обработ- чик этого события. Для организации связи адреса обработчика прерывания с номером прерывания используется таблица векторов прерываний – блок памяти в диапазоне адресов от 0000:0000 до 0000:03FFh, состоящий из 256 эле- ментов – дальних адресов обработчиков прерываний. В первом слове элемента таблицы записано смещение, а во втором – сегмент адреса обработчика преры- вания. Вектор прерывания с номером 0 находится по адресу 0000:0000, с но- мером 1 – по адресу 0000:0004 и т. д. Инициализация таблицы векторов пре- рываний выполняется сначала BIOS перед началом загрузки операционной си- стемы, а затем DOS (см. таблицу 9).

Аппаратные прерывания (обозначаются IRQ0-IRQ15) вырабатываются устройствами компьютера, как правило, при завершении ими операций обмена данными или при изменении состояния и обрабатываются программируемым контроллером прерываний.

Программы могут сами вызывать прерывания с помощью команды INT – программные прерывания. Программные прерывания удобно использовать для организации доступа к отдельным, общим для всех программ функциям.



2. Резидентные программы.

Резидентной программой называют программу, остающуюся в памяти после того, как она вернула управление операционной системе, и выполняю- щую обработку данных при обращении к ней.

Резидентную программу можно разделить на две части: инсталлятор и обработчик.

Инсталлятор обычно выполняет следующие действия:  
-- проверяет можно ли установить обработчик;  
-- устанавливает обработчик на определенный вектор прерывания; -- завершается, оставляя обработчик резидентным.

При повторном запуске инсталлятора, он может быть также использован для выгрузки резидентной части из памяти с восстановлением старых обработчиков.

Для установки вектора обработчика рекомендуется использовать следу- ющие функции DOS:

-- Функция DOS 25h (INT 21h) – установить адрес обработчика прерыва- ния:

Ввод:

АН = 25h,  
AL =номер прерывания,

DS = сегментный адрес обработчика,

DX = смещение обработчика в сегменте.

-- Функция DOS 35h (INT 21h) – получить адрес обработчика прерывания:

Ввод:

АН = 35h,

AL =номер прерывания.  
Вывод:

ES = сегментный адрес обработчика прерывания,

BX = смещение обработчика прерывания.

-- Функция DOS 31h (INT 21h) – оставить программу резидентной: Ввод: АН = 31h,

AL = код возврата,  
DX = размер резидента в 16-байтных параграфах (больше 06h), считая от начала PSP.

Обработчик резидентной программы – процедура, выполняющая необхо- димые действия. Эта процедура обычно заменяет стандартный обработчик про- граммного или аппаратного прерывания.

Принято разделять резидентные программы на активные и пассивные, в зависимости от того, перехватывают ли они прерывания от внешних устройств или получают управление, только если программа специально вызовет команду INT с нужным номером прерывания и параметрами. Кроме этого, обработчик прерывания может быть заменен и обычной (нерезидентной) программой для поддержки специфичных действий (по окончанию такой программы должен быть восстановлен старый обработчик прерывания).

Пример простейшего обработчика прерывания, полностью заменяющего старый обработчик:

int\_handler proc far

; сохранение всех модифицируемых регистров в стеке

...

; инициализация сегментных регистров

...

; выполнение необходимых действий

...

; восстановление используемых регистров из стека

...

iret ; возврат из прерывания

int\_handler endp

При вызове команды INT выполняются следующие действия:  
1) В стеке сохраняются регистры FLAGS, CS, IP; сбрасываются флаги IF и TF.  
2) В регистр CS из таблицы векторов прерываний заносится значение сегмента обработчика прерывания, а в регистр IP – смещение обработчика преры- вания.

3) Выполняется передача управления на обработчик программного прерыва- ния (CS:IP).

При вызове команды IRET выполняются следующие действия:  
1) Из стека восстанавливаются регистры IP, CS, FLAGS.  
2) Передается управление в прерванную программу, на команду, находящуюся непосредственно за командой программного прерывания INT.  
Для сохранения возможностей стандартного обработчика можно восполь- зоваться следующими вариантами формирования своего обработчика (адрес старого обработчика должен быть сохранен до установки нового обработчика): -- вызов стандартного обработчика в начале функции обработчика (ниже приведенные команды эквивалентны команде INT):

pushf

call old\_handler

-- вызов стандартного обработчика в конце функции обработчика:

jmp cs:old\_handler

При этом образуются цепочки обработчиков прерываний, когда один век- тор прерывания имеет несколько обработчиков, вызываемых друг за другом.

Для обработчиков аппаратных прерываний рекомендуется обязательно вызывать стандартный обработчик, т.к. кроме возврата из аппаратного прерывания требуется выполнить ряд действий с аппаратурой, в том числе и с контроллером прерываний. Ниже приведенный фрагмент кода предназначен для указания контроллеру прерываний, что обработка аппаратного прерывания завершена:

Во время работы обработчика прерываний, может произойти повторный вызов этого же прерывания, при этом возможны ошибки обработки. Чтобы за- щитить обработчик от таких ситуаций (или сделать так чтобы он являлся по- вторно входимым), можно временно запретить прерывания в критических участках кода:

cli ; запретить прерывания

mov al,byte ptr counter

cmp al,100

jb counter\_ok

sub al,100

mov byte ptr counter,al

counter\_ok:

sti ; разрешить прерывания

Следует помнить, что, пока прерывания запрещены, система не отслежи- вает изменения часов реального времени, не получает данных с клавиатуры, так что прерывания надо обязательно, при первой возможности, разрешать.

В DOS обработчик INT 21h не является повторно входимым. В отличие от прерываний BIOS, обработчики которых используют стек прерванной про- граммы, обработчик системных функций DOS записывает в SS:SP адрес дна одного из трех внутренних стеков DOS. Если функция была прервана аппарат- ным прерыванием, обработчик которого вызвал другую функцию DOS, она бу- дет пользоваться тем же стеком, затирая все, что туда поместила прерванная функция. Когда управление вернется в прерванную функцию, в стеке окажется мусор и произойдет ошибка. Лучший выход – вообще не использовать преры- вания DOS из обработчиков прерываний, но если от этого нельзя отказаться, то нужно принять необходимые меры предосторожности.

Если прерывание произошло в тот момент, когда не выполнялось никаких системных функций DOS, ими можно пользоваться. Чтобы определить, занята DOS или нет, надо сначала, до установки собственных обработчиков, опреде- лить адрес флага занятости DOS:

-- Функция DOS 34h (INT 21h) – определить адрес флага занятости DOS:

Ввод: АН = 34h  
Вывод: ES:BX = адрес однобайтного флага занятости DOS,

ES:BX – 1 = адрес однобайтного флага критической ошибки DOS.

Теперь обработчик прерывания может проверять состояние этих флагов и, если оба флага равны нулю, разрешается свободно пользоваться функциями DOS.

Если флаг критической ошибки не нуль, никакими функциями DOS поль- зоваться нельзя.

Если флаг занятости DOS не нуль, можно пользоваться только функция- ми 01h – 0Ch, а чтобы воспользоваться какой-нибудь другой функцией, при- дется отложить действия до тех пор, пока DOS не освободится. Чтобы это сде- лать, надо сохранить номер функции и параметры в каких-нибудь переменных в памяти и установить обработчик прерывания 08h или 1Ch. Этот обработчик будет при каждом вызове проверять флаги занятости и, если DOS освободи- лась, вызовет функцию с номером и параметрами, оставленными в переменных в памяти. Кроме того, участок программы после проверки флага занятости – критический, и прерывания на этом участке должны быть запрещены.

Не все функции DOS завершаются быстро – функция чтения символа с клавиатуры может оставаться в таком состоянии до нажатия клавиши, и все это время флаг занятости DOS будет установлен в 1. В DOS предусмотрена и такая ситуация.ВсефункциивводасимволовсожиданиемвызываютINT 28hвтом же цикле, в котором они опрашивают клавиатуру, так что, если установить об- работчик прерывания 28h, из него можно вызывать все функции DOS, кроме 01h – 0Ch.

Функции BIOS также часто оказываются не повторно входимыми (к ним следует отнести обработчики программных прерываний 05, 08, 09, 0Bh, 0Ch, 0Dh, 0Eh, 10h, 13h, 14h, 16h, 17h). Так как нет флага занятости BIOS, можно создать его самим:

int10\_handler proc far

inc cs:byte ptr int10\_busy ; увеличить флаг занятости

pushf ; передать управление старому

; обработчику INT 10h,

call cs:dword ptr old\_int10 ; эмулируя команду INT,

dec cs:byte ptr int10\_busy ; уменьшить флаг занятости

iret

int10\_busy db 0

int10\_handler endp

Теперь обработчики аппаратных прерываний могут пользоваться коман- дой INT 10h, только тогда когда флаг занятости int10\_busy равен нулю, и это не приведет к ошибкам, если не найдется чужой обработчик прерывания, который тоже будет обращаться к INT 10h и не будет ничего знать о нашем флаге занятости.

Ниже приведен пример обобщенной структуры резидентной программы.

; резидентная программа для перехвата прерывания с номером V .model tiny

start:

.code

.386

org 100h

; для команд pusha, popa

; размер PSP для COM программы

; переход на инсталлятор

; адрес старого обработчика

; другие данные резидентной части

pushf ; сохранение флагов

call cs:old\_int ; вызов старого обработчика

; сохранение регистров в стеке

pusha

push ds

push es

; настройка регистра ds на данные резидентной программы

push cs

pop ds

; обработка данных

...

; восстановление регистров из стека

pop es

pop ds

popa

; возврат из обработчика

iret

int\_handler endp

; инсталлятор обработчика прерываний

install\_handler:

; сохранить адрес старого обработчика

jmp install\_handler

old\_int dd 0

...

; процедура обработчика прерываний

int\_handler proc far

mov ah,35h  
mov ah,V  
int 21h  
mov word ptr old\_int,bx  
mov word ptr old\_int+2,es ; сегмент обработчика ; установить новый обработчик

mov ah,25h ; установить адреса обработчика mov al,V ; V = номер вектора обработчика int 21h  
; завершить программу и оставить резидентную часть

mov ax,3100h ; оставить программу резидентной

; DX = размер резидентной части программы в параграфах

mov dx,(install\_handler – start + 10Fh) / 16

int 21h

end start ; конец программы

Код программы:

.model tiny

org 100h

.data

file\_name db "file.txt", 0

handler dw 0000h

new\_file\_name db "print.txt", 0

new\_handler dw 0000h

num\_of\_bytes dw 0000h

size\_of\_string dw 0000h

invite db ": $"

farewell db "The End.$"

;buffers

empty\_line db 0ah, 0dh, "$"

string db 50, 52 dup ('$')

file\_buffer db 202 dup ('$')

screen\_buffer db 202 dup ('$')

old\_int dd 0

old\_int\_offset dw 0 ;the address of the old interrupter

old\_int\_segment dw 0

.code

start:

call saveOldInt

call installNewInt

open\_file:

mov ah, 3Dh ;3Dh opens a file

mov al, 02h ;for reading and writing

lea dx, file\_name

int 21h

mov handler, ax ;get the file handler

mov dx, offset invite

mov ah, 09h

int 21h

mov dx, offset string

mov ah, 0Ah

int 21h

mov dx, offset empty\_line

mov ah, 09h

int 21h

read\_from\_file:

mov bx, handler

mov ah, 3Fh ;3Fh-function reads sth from the file

mov cx, 00C8h ;00C8h = 200 bytes

mov dx, offset file\_buffer

int 21h

mov num\_of\_bytes, ax

mov ax, num\_of\_bytes

cmp ax, 0000h ;if everything's been read

je done\_reading

print\_on\_the\_screen:

mov si, offset file\_buffer

mov dx, offset file\_buffer

mov ah, 09h

int 21h

jmp read\_from\_file

done\_reading:

mov dx, offset empty\_line

mov ah, 09h

int 21h

lea dx, farewell

mov ah, 09h

int 21h

mov dx, offset empty\_line

mov ah, 09h

int 21h

close\_file:

mov bx, handler

xor ax, ax

mov ah, 3Eh ;3Eh-function closes the file

int 21h

check\_Prt\_Scr:

mov ah, 00h ;if key pressed, compare it with the code of PrintScreen

int 16h

cmp al, 2ch ;PrintScreen - 2ch

int 05h

call setOldInt

exit:

int 20h

saveOldInt proc

push es

push ax

push bx

mov ah, 35h ;35h-function gets the address of the interrupter

mov al, 05h ;number of the interruption

int 21h

mov word ptr old\_int, bx

mov word ptr old\_int + 2, es

mov old\_int\_offset, bx

mov old\_int\_segment, es

pop bx

pop ax

pop es

ret

saveOldInt endp

installNewInt proc

cli ;interruptions are forbidden

push ax

push dx

push es

push ds

pop es

mov dx, offset newInt

mov ah, 25h ;25h-function sets the address of the new interrupter

mov al, 05h ;number of the interruption

int 21h

pop es

pop ax

pop dx

sti ;interruptions are allowed

ret

installNewInt endp

setOldInt proc

cli ;interruptions are forbidden

push bx

push es

push ax

push ds

pop es

mov dx, old\_int\_offset

mov ds, old\_int\_segment

mov ah, 25h ;25h-function installs the address of the interruption

mov al, 05h ;number of the interruption

int 21h

pop ax

pop es

pop bx

sti ;interruptions are allowed

ret

setOldInt endp

newInt proc

cli ;interruptions are forbidden

pushf

pusha

push ds

push es

push cs

pop ds

mov ah, 3Dh ;3Dh opens a file

mov al, 02h ;for reading and writing

lea dx, new\_file\_name

int 21h

mov new\_handler, ax

push 0B800h ;the segment of display in videomode

pop es

mov cx, 0000h

mov di, offset screen\_buffer

xor ax, ax

get\_the\_contents:

cmp ax, 200

jae screen\_to\_file ;if 200 >= ax

cmp cx, 047FFh ;size of the screen buffer

jae done

mov bx, cx

push es:[bx]

pop cs:[di]

inc cx

inc cx

inc di

inc ax

jmp get\_the\_contents

screen\_to\_file:

mov ax, 200

pusha

lea dx, screen\_buffer

mov bx, new\_handler

mov cx, ax ;number of bytes to write

mov ah, 40h ;40h-function writes sth to the file

int 21h

popa

xor ax, ax

mov di, offset screen\_buffer

jmp get\_the\_contents

done:

mov ax, 200

pusha

lea dx, screen\_buffer

mov bx, new\_handler

mov cx, ax ;number of bytes to write

mov ah, 40h ;40h-function writes sth to the file

int 21h

popa

pop es

pop ds

popa

popf

sti ;interruptions are allowed

push ax

mov al, 20h

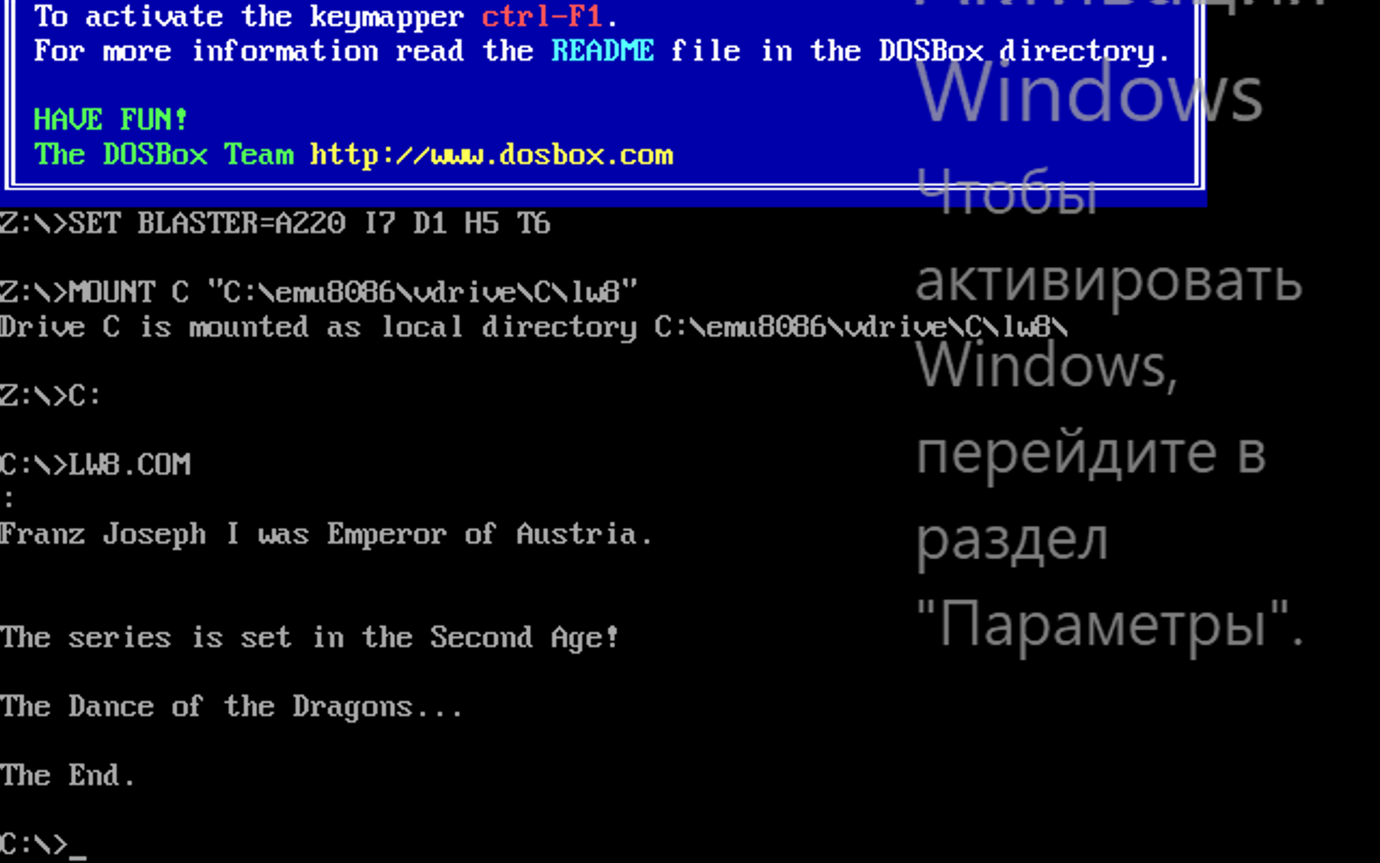
out 20h, al ;indicates that interruption is over

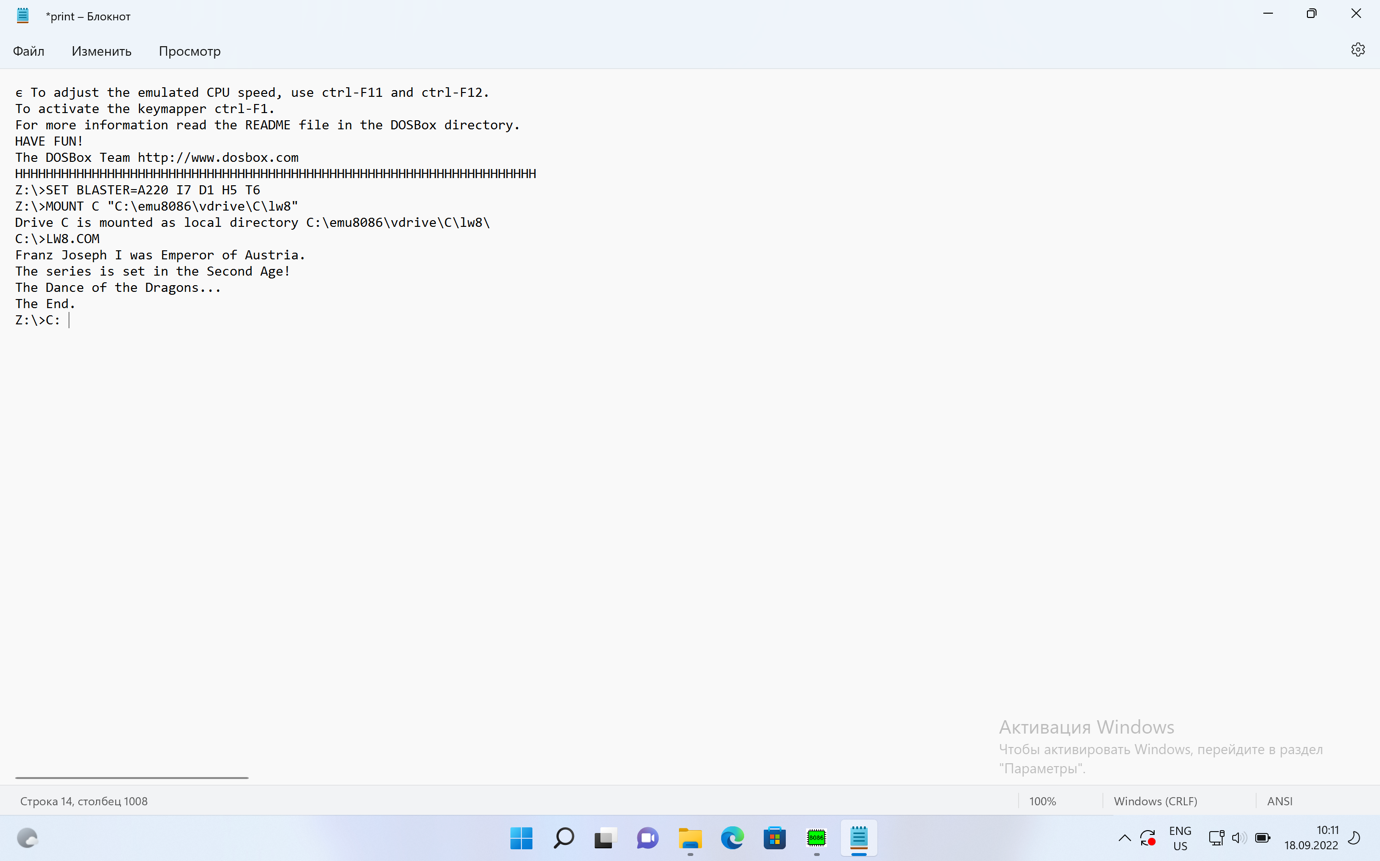
pop ax

iret

newInt endp

Примеры работы программы:





Вывод:

В ходе лабораторной работы было написано приложение, записывающее содержимое экрана в файл print.txt при нажатии клавиши Print-Screen. Я получила понятие об обработчиках прерываний и изучила особенности создания резидентных программ.