МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

Архитектура вычислительных систем

|  |
| --- |
| Работа с математическим сопроцессором |

Руководитель К. В. Богданов

подпись, дата инициалы, фамилия

Обучающийся БПИ19-01, 191219044 И.А. Фролов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2021 г.

Условие задачи:

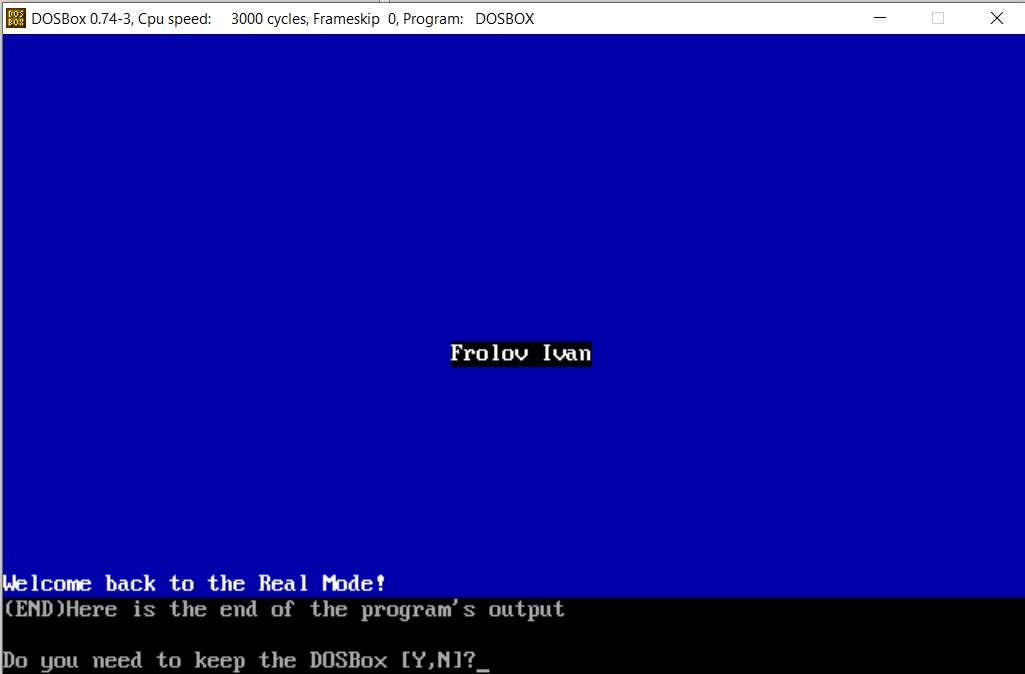
Перейти в защищенный режим. Вывести на экран в центре строки, соответствующей номеру варианта (отсчет сверху) свою фамилию. Вернуться в реальный режим. Вывести сообщение на экран о возврате в реальный режим (любым удобным способом). Завершить работу с кодом возврата 0.

Результаты проделанной работы:

Для перехода в защищенный режим, необходимо объявить структуру дескрипторов сегментов и дескрипторов ловушек, создать глобальную таблицу дескрипторов. Также необходимо запретить вызов аппаратных прерываний и заполнить регистры данных и стека.

Для вывода фамилии и имени в центре экрана нужно отсчитать отступ в 15 строк и потом на 15 строке отступить пол строки и отнять половину длины строки, в которой записана фамилия и имя, чтобы вывести информацию оп центру. Смещение по строке необходимо делать, чтобы отступить нужное количество строк. Так как символ составляет 2 бита, то необходимо умножить количество символов в строке на количество строк, которые нужно отступить, умножить на 2, отнять половину длины строки, умноженную на 2, а также отнять строку с данными, также умноженную на 2.

Для выхода из защищённого режима в режим реальных адресов, необходимо заполнить дескрипторы для реального режима, обновить регистр CS . Далее при переходе в режим реальных адресов, необходимо восстановить регистры на сегменты данных, стека, регистра IDTR реального режима, а также разрешить системные прерывания.



Код программы:

;вывод символов на экран в защищённом режиме

.386p   ;разрешение трансляции всех, в том числе и привилегированных команд МП 386 и 486

;описание дескрипторов сегмента

descr   struc

limit   dw  0   ;граница (биты 0…15)

base\_l  dw  0   ;база, биты 0…15

base\_m  db  0   ;база, биты 16…23

attr\_1  db  0   ;байт атрибутов 1

attr\_2  db  0   ;граница (биты 16…19) и атрибуты 2

base\_h  db  0   ;база, биты 24…31

descr   ends

;описание дескрипторов ловушек

trap    struc

offs\_1  dw  0

sel dw  16

rsrv    db  0

attr    db  8Fh

offs\_h  dw  0

trap    ends

data segment use16  ;16-разрядное приложение

;таблица глобальных дескрипторов GDT

gdt\_0   label   word

gdt\_null    descr<0,0,0,0,0,0>                          ;селектор 0-обязательный нулевой дескриптор

gdt\_data    descr<data\_size-1,0,0,92h,0,0>              ;селектор 8-сегмент данных

gdt\_code    descr<code\_size-1,0,0,98h,0,0>              ;селектор 16-сегмент команд

gdt\_stack   descr<255,0,0,92h,0,0>                      ;селектор 24, сегмент стека

gdt\_screen  descr<4095,8000h,0Bh,92h,0,0>               ;селектор 32, видеобуфер

gdt\_size=$-gdt\_null ;размер GDT

idt label   word

trap    32 dup ()

idt\_size=$-idt

pdescr  dq  0       ;псевдодескриптор для LGDT

sym db  97          ;символ для вывода на экран

stoka\_name db  "Frolov Ivan"

real\_sp dw  0       ;ячейка для хранения SP

real\_ss dw  0       ;ячейка для хранения SS

pos dw  950

mes db 'Welcome back to the Real Mode!$'

strlen dw 11

offsetY dw 15

string  db '++++++++++' ;строка

len=$-string    ;её размер

data\_size=$-gdt\_null    ;размер сегмента данных

data    ends    ;конец сегмента данных

text    segment 'code' use16    ;начало сегмента команд. Будем работать в 16-разрядном режиме

assume CS:text,DS:data

begin   label   word

;заглушка вместо обработчика всех исключений, которые у нас отсутствуют в защищенном режиме

dummy\_exc   proc

pop EAX

pop EAX

mov SI,offset string+5

mov AX,1111b

jmp home

dummy\_exc   endp

main    proc

xor EAX,EAX     ;очистка EAX

mov AX,data     ;инициализация DS

mov DS,AX       ;в реальном режиме

;вычислим 32-битовый линейный адрес сегмента данных и загрузим его в дескриптор

;сегмента данных в таблице GDT. В регистре АХ уже находится сегментный адрес.

shl EAX,4                   ;умножим его на 16

mov EBP,EAX                 ;сохраняем его в EBP

mov EBX, offset gdt\_data    ;на BX адрес дескриптора

mov [EBX].base\_l,AX         ;загрузим младшую часть базы

rol EAX,16                  ;обмен старшей и младшей частей EAX

mov [EBX].base\_m,AL         ;загрузим среднюю часть базы

;аналогично для линейного адреса сегмента команд

xor EAX,EAX             ;очищаем EAX

mov AX,CS               ;берем адрес сегмента команд

shl EAX,4               ;умножаем его на 16

mov EBX,offset gdt\_code ;адрес дескриптора

mov [EBX].base\_l,AX     ;загрузим младшую часть базы

rol EAX,16              ;обмен старшей и младшей частей EAX

mov [EBX].base\_m,AL     ;загружаем среднюю часть базы

;аналогично для линейного адреса сегмента стека

xor EAX,EAX     ;очищаем EAX

mov AX,SS       ;берем адрес сегмента стека

shl EAX,4       ;умножаем его на 16

mov EBX,offset gdt\_stack    ;адрес дескриптора

mov [EBX].base\_l,AX         ;загрузим младшую часть базы

rol EAX,16                  ;обмен старшей и младшей частей EAX

mov [EBX].base\_m,AL         ;загружаем среднюю часть базы

;подготовим псевдодескриптор pdescr и загрузим регистр GDTR

mov dword ptr pdescr+2,EBP      ;база GDT, биты 0…31

mov word ptr pdescr, gdt\_size-1 ;граница GDT

lgdt    pdescr                  ;загрузим регистр GDTR

;подготовимся к переходу в защищённый режим

cli             ;запрет аппаратных прерываний

;загрузим IDTR

mov word ptr pdescr, idt\_size-1     ;граница IDT

xor EAX,EAX                         ;EAX=0

mov AX,offset idt                   ;смещение idt в сегменте данных

add EAX,EBP                         ;плюс линейный адрес сегмента данных

mov dword ptr pdescr+2,EAX          ;адрес IDT в pdescr

lidt    pdescr                      ;загрузка IDTR

;переходим в защищённый режим

mov EAX,CR0 ;получим слово состояния машины

or  EAX,1   ;установим бит PE

mov CR0,EAX ;запишем назад слово состояния

;мы в защищённом режиме!

;загружаем в CS:IP селектор:смещение точки continue и заодно очищаем очередь команд

db  0Eah                ;код команды far jmp - подгядел в инете

dw  offset continue     ;смещение

dw  16                  ;селектор сегмента команд

continue:

;делаем адресуемыми данными

mov AX,8    ;селектор сегмента данных

mov DS,AX

;делаем адресуемым стек

mov AX,24   ;селектор сегмента стек

mov SS,AX

home:   mov si,offset string

mov si,offset string

mov cx,len

mov ah,74h

mov di,1600

scr:    lodsb

stosw

loop    scr

;инициализируем ES и выводим символы

mov AX,32   ;селектор сегмента видеобуфера

mov ES,AX

;Заполнение

mov BX, 0

mov CX, 80\*25   ;число выводимых символов

;mov    AX,word ptr stoka\_name  ;начальный символ с атрибутами

mov si,0

screen:

mov AX, 1f00h ;word ptr stoka\_name[si]

mov ES:[BX], AX ;вывод в видеобуфер

add BX,2    ;сместимся в видеобуфере

inc si  ;следующий символ

loop    screen  ;цикл вывода на экран

;вывод имени

mov BX, 4\*80\*2+2\*40-5\*2 ;высчитывание середины

mov CX, 11

mov SI, 0

printName:

mov AX,word ptr stoka\_name[si]

;Добавим белый цвет

and AX, 00FFh

add AX, 0F00h

mov ES:[BX], AX ;вывод в видеобуфер

add BX,2    ;сместимся в видеобуфере

inc si  ;следующий символ

loop    printName   ;цикл вывода на экран

;подготовим переход в реальный режим

;сформируем и загрузим дескрипторы для реального режима

mov gdt\_data.limit,0FFFFh   ;граница сегмента данных

mov gdt\_code.limit,0FFFFh   ;граница сегмента кода

mov gdt\_stack.limit,0FFFFh  ;граница сегмента стека

mov gdt\_Screen.limit,0FFFFh     ;граница доп. сегмента

mov AX,8    ;загружаем теневой регистр

mov DS,AX   ;сегмента данных

mov Ax,24   ;загружаем теневой регистр

mov SS,AX   ;сегмента стека

mov AX,32   ;загружаем теневой регистр

mov ES,AX   ;дополнительного сегмента

;выполним дальний переход для того, чтобы заново загрузить

;селектор в регистр CS и модифицируем его теневой регистр

db 0EAh         ;командой дальнего перехода

dw offset go    ;загрузим теневой регистр

dw 16           ;сегмента команд

;переключим режим процессора

go: mov EAX,CR0     ;получим содержимое CR0

and EAX,0fffffffeh  ;сбросим бит PE

mov CR0,EAX         ;запишем назад в CR0

db 0EAh ;код команды far jmp

dw offset return    ;смещение

dw text ;сегмент

;теперь процессор снова работает в реальном режиме

;восстановим операционную среду реального режима

return: mov AX,data ;восстановим адресуемость

mov DS,AX   ;данных

mov AX,stk  ;адресуемость

mov SS,AX   ;стека

mov SP,256

mov SS,real\_ss

;Восстановим состояние регистра IDTR реального режима (хотя можно и не делать)

mov ax,3ffh ;граница таблицы векторов

mov word ptr pdescr,AX

mov eax,0   ;смещение таблицы векторов

mov dword ptr pdescr+2,EAX

lidt    pdescr  ;загрузим pdescr IDTR

;разрешим аппаратные и немаскируемые прерывания

sti  ;разрешение прерываний

;проверим выполнение функций DOS после возврата в реальный режим

mov AH,09h ;функции вывода на экран строки

mov EDX,offset mes  ;адрес строки

int 21h ;вызов DOS

mov AX,4C00h    ;завершаем программу

int 21h ;обычным образом

main    endp    ;конец главной процедуры

code\_size=$-main    ;размер сегмента команд

text    ends    ;конец сегмента команд

stk segment stack 'stack'   ;начало сегмента стека

db  256 dup ('^')

stk ends    ;конец сегмента стека

end main    ;конец программы