МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Организация ЭВМ и систем»
ТЕМА: Разработка собственного прерывания.

Студентка гр. 9382	_________________	Пя С.
Преподаватель	_________________	Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить команды для работы с прерываниями в ассемблере, написать собственное прерывание.

Теоретические сведения:

Прерывание - это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (обработка сигнала таймера, нажатия клавиши и т.д.). Когда возникает прерывание, процессор прекращает выполнение текущей программы (если ее приоритет ниже) и запоминает в стеке вместе с регистром флагов адрес возврата(CS:IP) - места, с которого будет продолжена прерванная программа.

Затем в CS:IP загружается адрес программы обработки прерывания и ей передается управление. Адреса 256 программ обработки прерываний, так называемые векторы прерывания, имеют длину по 4 байта (в первых двух хранится значение IP , во вторых - CS) и хранятся в младших 1024 байтах памяти. Программа обработки прерывания должна заканчиваться инструкцией IRET (возврат из прерывания), по которой из стека восстанавливается адрес возврата и регистр флагов.

Программа обработки прерывания - это отдельная процедура, имеющая структуру:

```
SUBR_INT PROC FAR

 PUSH AX ; сохранение изменяемых регистров

 <действия по обработке прерывания>

 POP AX ; восстановление регистров

 ...
```

```
MOV AL, 20H
OUT 20H,AL
IRET
SUBR_INT ENDP
```

Две последние строки обработчика прерывания, указанные перед командой IRET выхода из прерывания, необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное.

Замечание: в лабораторной работе действиями по обработке прерывания может быть вывод на экран некоторого текста, вставка цикла задержки в вывод сообщения или включение звукового сигнала.

Программа, использующая новые программы обработки прерываний при своем завершении должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H возвращает текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. В соответствии с этим, программа должна содержать следующие инструкции:

```
; -- в сегменте данных
KEEP_CS DW 0 ; для хранения сегмента
KEEP_IP DW 0 ; и смещения вектора прерывания
; -- в начале программы
MOV AH, 35H ; функция получения вектора
MOV AL, 1CH ; номер вектора
```

```
INT 21H
MOV KEEP_IP, BX ; запоминание смещения
MOV KEEP_CS, ES ; и сегмента вектора прерывания
```

Для установки адреса нового обработчика прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая помещает заданные адреса сегмента и смещения обработчика в вектор прерывания с заданным номером.

```
PUSH DS
MOV DX, OFFSET ROUT ; смещение для процедуры в DX
MOV AX, SEG ROUT ; сегмент процедуры
MOV DS, AX ; помещаем в DS
MOV AH, 25H ; функция установки вектора
MOV AL, 60H ; номер вектора
INT 21H ; меняем прерывание
POP DS
```

Далее может выполняться вызов нового обработчика прерывания. В конце программы восстанавливается старый вектор прерывания

```
CLI
PUSH DS
MOV DX, KEEP_IP
```

```
MOV AX, KEEP_CS
MOV DS, AX
MOV AH, 25H
MOV AL, 1CH
INT 21H ; восстанавливаем старый вектор прерывания
POP DS
STI
```

Задание:

10 Вариант – 4А

4 - 08h - прерывание от системного таймера - генерируется автоматически операционной системой 18 раз в сек.

А - Печать сообщения на экране;

Ход работы:

При разработке программы были использованы следующие команды:

Инструкция OUT выводит данные из регистра AL или AX (ИСТОЧНИК) в порт ввода-вывода. Номер порта должен быть указан в ПРИЁМНИКЕ.

Тестирование.

Вводные данные	Результат
	I love you!

	I love you! I love you! I love you! I love you! I love you! I love you! I love you!

Выводы.

В результате выполнения лабораторной работы был разработан код, определяющий собственное прерывание. Были улучшены навыки письма в ассемблере.

Приложение.

Текст файла HELLO1.LST

```
0000                      AStack    SEGMENT  STACK
0000  0400[                         DW 1024 DUP(?)
         ????
                    ]

0800                      AStack    ENDS
0000                      DATA SEGMENT
0000  0000                KEEP_CS DW 0
0002  0000                KEEP_IP DW 0
0004  49 20 6C 6F 76 65   message db 'I love you!',10,13,'$' ;строк�
                          � для сообщения
      20 79 6F 75 21 0A
      0D 24

0012                      DATA ENDS
0000                      CODE      SEGMENT
                                    ASSUME CS:CODE, DS:DATA, SS:AStack

0000                                Output PROC FAR
0000  EB 43 90                             jmp start

0003  0000                ST_SS DW 0000
0005  0000                ST_AX DW 0000
0007  0000                ST_SP DW 0000
0009  001E[               IStack DW 30 DUP(?)
         ????
                    ]


0045                         start:

0045  2E: 89 26 0007 R              mov ST_SP, SP
004A  2E: A3 0005 R          mov ST_AX, AX
004E  8C D0                  mov AX, SS
0050  2E: A3 0003 R          mov ST_SS, AX
0054  2E: A1 0009 R          mov AX, IStack
0058  8E D0                  mov SS, AX
005A  2E: A1 0005 R          mov AX, ST_AX

005E  50                     push ax
005F  52                      push dx
0060  B4 09                   mov  ah, 09h
0062  BA 0004 R               mov dx, offset message
0065  CD 21                   int  21h
0067  5A                      pop dx
0068  58                      pop ax

0069  2E: A3 0005 R           mov ST_AX,AX
006D  2E: A1 0003 R              mov AX,ST_SS
0071  8E D0                      mov SS,AX
0073  2E: 8B 26 0007 R                  mov SP,ST_SP
0078  2E: A1 0005 R              mov AX,ST_AX

007C  B0 20                   mov al,20h
```

```
007E  E6 20                         out 20h,al
0080  CF                             iret
0081                          Output ENDP

0081                          Main PROC FAR
0081  1E                      push ds
0082  2B C0                   sub ax,ax
0084  50                      push ax
0085  B8 ---- R               mov ax,data
0088  8E D8                   mov ds, ax

008A  B8 3523                        mov ax,3523h
008D  CD 21                    INT 21H
008F  89 1E 0002 R             MOV KEEP_IP, BX ; запоминание см�
                              �щения
0093  8C 06 0000 R             MOV KEEP_CS, ES ; и сегмента вект
                              ора прерывания

0097  1E                       PUSH DS
0098  BA 0000 R                MOV DX, OFFSET Output ; смещение дл�
                              � процедуры в DX
009B  B8 ---- R                MOV AX, SEG Output ; сегмент проце
                              дуры
009E  8E D8                    MOV DS, AX ; помещаем в DS
00A0  B8 2508                         mov ax,2508h
00A3  CD 21                    INT 21H ; меняем прерывание
00A5  1F                       POP DS

00A6                           waiting:
00A6  B4 01                    mov   ah,1h
00A8  CD 21                          int   21h
00AA  3C 1B                          cmp   al,27
00AC  75 13                          jne nextstep

00AE  FA                                    CLI
00AF  1E                       PUSH DS
00B0  8B 16 0002 R             MOV DX, KEEP_IP
00B4  A1 0000 R                MOV AX, KEEP_CS
00B7  8E D8                    MOV DS, AX
00B9  B8 2508                         mov AX,2508h
00BC  CD 21                    INT 21H ; восстанавливаем ст�
                              �рый вектор прерывания
00BE  1F                       POP DS
00BF  FB                       STI
00C0  CB                       ret

00C1                                 nextstep:
00C1  EB E3                          jmp waiting



00C3                          Main ENDP
00C3                          CODE ENDS
                               END Main
```

```
Segments and Groups:

                N a m e               Length  Align Combine Class

ASTACK . . . . . . . . . . . . .         0800   PARA   STACK
CODE . . . . . . . . . . . . . .         00C3   PARA   NONE
DATA . . . . . . . . . . . . . .         0012   PARA   NONE

Symbols:

                N a m e               Type    Value  Attr

ISTACK . . . . . . . . . . . . .         L WORD 0009   CODE   Length = 001E

KEEP_CS  . . . . . . . . . . . .         L WORD 0000   DATA
KEEP_IP  . . . . . . . . . . . .         L WORD 0002   DATA

MAIN . . . . . . . . . . . . . .         F PROC 0081   CODE   Length = 0042
MESSAGE  . . . . . . . . . . . .         L BYTE 0004   DATA

NEXTSTEP . . . . . . . . . . . .         L NEAR 00C1   CODE

OUTPUT . . . . . . . . . . . . .         F PROC 0000   CODE   Length = 0081

START  . . . . . . . . . . . . .         L NEAR 0045   CODE
ST_AX  . . . . . . . . . . . . .         L WORD 0005   CODE
ST_SP  . . . . . . . . . . . . .         L WORD 0007   CODE
ST_SS  . . . . . . . . . . . . .         L WORD 0003   CODE

WAITING  . . . . . . . . . . . .         L NEAR 00A6   CODE

@CPU . . . . . . . . . . . . . .         TEXT  0101h
@FILENAME  . . . . . . . . . . .         TEXT  HELLO1
@VERSION . . . . . . . . . . . .         TEXT  510


     94 Source  Lines
     94 Total   Lines
     20 Symbols

  48002 + 461305 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```