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

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Организация ЭВМ и систем»
Тема: Написание собственного прерывания.

Студент гр. 3384 _____ Рудаков А.Л.

Преподаватель _____ Ковалев А.Д.

Санкт-Петербург
2024

Цель работы.

Познакомиться с теорией о собственных прерываниях в Ассемблере. Написать программу, вызывающую собственное прерывание.

Задание.

Шифр 19c

Номер и назначение вектора прерывания: c - 23h - прерывание, генерируемое при нажатии клавиш Control+C; иначе программа должна завершаться.

Действие, реализуемое программой обработки прерываний: 19. Исключение русских букв и цифр, введенных во входной строке, при формировании выходной строки.

Основные теоретические положения.

Прерывание - это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (нажатие клавиши, обработка сигнала таймера, ввод-вывод строки 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 ;  SUBR_INT – имя, задаваемое пользователем
   PUSH AX  ; сохранение изменяемых регистров
    <действия по обработке прерывания>
   MOV AL, 20H ; корректное завершение
   OUT 20H, AL   ; обработчика
   POP AX ; восстановление регистров
IRET
SUBR_INT ENDP
```

Две строки перед POP AX и IRET (завершение обработчика) необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное.

Программа, использующая новую программу обработки прерывания, при своем завершении должна восстанавливать оригинальные векторы прерываний. Для этого используется функция 35 прерывания 21H, которая сохраняет текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. В этом случае головная программа должна содержать инструкции (для примера заменяется вектор прерывания 1Ch):

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

Для задания адреса собственного обработчика прерывания с заданным номером в таблицу векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес нового обработчика (сегмент – в DS, смещение – в DX)

```
PUSH DS
MOV DX, OFFSET SUBR_INT ; смещение для процедуры в DX
MOV AX, SEG SUBR_INT    ; сегмент процедуры в AX
MOV DS, AX     ; помещаем в DS
MOV AH, 25H    ; функция установки вектора
MOV AL, 1CH    ; номер вектора
INT 21H        ; меняем прерывание
POP DS
```

В конце программы нужно восстановить 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 вектор
POP DS
STI
```

Выполнение работы.

В начале программы задаются расположение сегментов под *DOS*, модель памяти *small* и размер стека в 1Кб.

Далее в сегменте *.data* Объявляются переменные типа *db: input_string, output_string,* в которых записано “приглашение” для ввода и вывода строки, а также поля для ввода типа *db*:

```
input db 255
      db ?
      dp 255 dup(0)
```

и

output db 255

db ?

dp 255 dup(0)

Они разделены по трем строкам, так как в первой строке содержится максимальная длина строки, во второй фактическая длина строки, и в третьей сама строка, заполненная 0.

Кроме того есть переменные типа *dw: out_length* (будет хранить длину новой строки), *KEEP_CS, KEEP_IP* (нужны для хранения сегмента и смещения прерывания соответственно).

Далее идет сегмент кода *.code*. Внутри первой и основной процедурой является *main proc*. В ней инициализируется сегмент данных, после чего идет сохранение оригинального вектора прерывания *23h* и установка своего обработчика прерывания.

Внутри блока сохранения оригинального вектора прерывания происходит получение самого вектора и его номера, после чего вызов прерывания *int 21h* и запоминание смещения и сегмента при помощи выделенных под это переменных.

Внутри блока установки своего обработчика прерывания происходит передача смещения и сегмента процедуры *SUBR_INT*, в которой обрабатывается новое прерывание, в регистры *dx* и *ax*(после в *ds*) соответственно, далее записываем функцию установки вектора и номер вектора в *ah* и *al* соответственно, после чего вызывается *int 21h* и происходит смена прерывания.

Далее вызывается само прерывание *int 23h*, после чего действие переходит к обработчику прерывания (процедуре *SUBR_INT*).

Внутри происходит вывод строки *input_string*, ввод строки *input*, после чего обработка полученной строки. Обработка происходит в цикле *process_loop*, пока поданная строка не закончится. Внутри происходит поэлементное сравнение символов строки вначале с русскими буквами, если это не русские буквы, идет переход в блок *check_digit*, внутри которого

осуществляется проверка на цифры. Если символ оказался не цифрой и не русской буквой, то переход к блоку *add_output*, в котором происходит запись символа в новую строку, попутно увеличивая размер *out_length*, после чего переходит к блоку *skip*. Если символ оказывается русской буквой или цифрой, действие так же переходит к блоку *skip*, в котором уменьшается счетчик длины строки и переходит к следующему символу строки, после чего возвращается к *process_loop*. После выполнения цикла идет блок print_output, внутри которого в новую строку добавляется символ конца строки *'$'*, выводится символ перехода на новую строку, выводится строка *output_string* и затем выводится строка *output*, после чего в строках:

mov al, 20h

out 20h, al

происходит корректное завершение обработчика и отправление команды завершения. Далее идет *iret* – возврат из прерывания.

После вызова прерывания в процедуре main выполняется блок *exit_program*, внутри которого через *cli* отключается прерывание для восстановления вектора, далее восстанавливается вектор *23h*, и прерывание включается через *sti*.

После этого программа завершается.

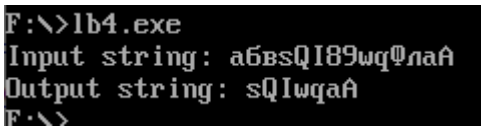

Рисунок 1. Пример ввода/вывода

Программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Input string: абвsQI89wqФлаА	Output string: sQIwqaA	ok
2.	Input string: 123456	Output string:	ok
3.	Input string:	Output string:	ok

	АбвГДУвьады		
4.	Input string: AsdflISDJDMmf	Output string: AsdflISDJDMmf	ok
5.	Input string: 123ыовлы99воф83окьв	Output string:	ok

Выводы.

В ходе лабораторной работы была написана программа на языке ассемблер, заменяющая прерывание 23h.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: LR4.asm

```
DOSSEG
.model small
.stack 400h

.data
  input_string db 'Input string: $'
  output_string db 'Output string: $'
  input db 255           ; строка для ввода
     db ?
     db 255 dup(0)
  output db 255          ; строка для вывода
     db ?
     db 255 dup(0)

  out_length dw 0
  KEEP_CS dw 0
  KEEP_IP dw 0


.code
main PROC
  mov ax, @data
  mov ds, ax

  ; сохранение оригинального вектора прерывания 23h
  mov ah, 35h              ; получение веткора
  mov al, 23h              ; номер вектор
  int 21h
  mov KEEP_IP, bx            ; запоминание смещение
  mov KEEP_CS, es            ; запоминание сегмент

  ; устанавление своего обработчика прерывания
  push ds
  mov dx, OFFSET SUBR_INT         ; смещение для процедуры в dx
  mov ax, SEG SUBR_INT          ; сегмент процедуры в ax
  mov ds, ax               ; перемещение в ds
  mov ah, 25h               ; функция установки вектора
  mov al, 23h               ; номер вектора
  int 21h                 ; смена прерывания
  pop ds

  int 23h

exit_program:
  cli                    ; отключение прерывания перед восстановлением
вектора
  push ds                ; сохранение ds для восстановления

  ; Восстановление 23h
  mov dx, KEEP_IP            ; смещение старого вектора
  mov ax, KEEP_CS            ; сегмент старого вектора
  mov ds, ax
  mov ah, 25h               ; функция восстановления вектора
  mov al, 23h               ; номер вектора
  int 21h
```

```
  pop ds                  ; восстановление ds
  sti                    ; включение прерывания

  mov ax, 4C00h              ; завершение программы
  int 21h

main ENDP

; обработчик прерывания Control+C 23h
SUBR_INT PROC FAR
  push ax                 ; сохранение регистров

  ; ввод строки
  mov ah, 09h               ; вывод строки
  lea dx, input_string
  int 21h

  mov ah, 0Ah               ; ввод строки
  lea dx, input
  int 21h

  lea si, input+2             ; указатель на начало введенной строки
  mov cl, [input]             ; длина введенной строки
  lea di, output             ; указатель на выходную строку

  ; обработка
process_loop:
  cmp cl, 0                ; проверка на конец строки
  je print_output             ; если конец, переход к печати

  mov al, [si]              ; получение символа
  cmp al, 'A'               ; сравнение с русскими буквами
  jb check_digit             ; если меньше, проверка цифр
  cmp al, 'я'               ; сравнение с последней русской цифрой
  ja check_digit             ; если больше, проверка цифр

  jmp skip                ; пропуск, если русская буква

check_digit:
  cmp al, '0'               ; проверка на цифры
  je add_output              ; если меньше, добавление в строку
  cmp al, '9'               ; проверка на последнюю цифру
  ja add_output              ; если больше, добавление в строку
  jmp skip

skip:
  inc si                  ; переход к следующему символу
  dec cl                  ; усеньшение сетчика длины исходной строки
  jmp process_loop

add_output:
  mov [di], al              ; добавление символа в выходную строку
  inc di                  ; увеличение указатель для ввода
  inc out_length             ; увеличение счетчика выходной строки
  jmp skip

print_output:
  mov byte ptr [di], '$'          ; завершение строки символом $

  mov dl, 10               ; символ Line Feed
  mov ah, 2
  int 21h
```

```
  ; вывод строки
  mov ah, 09h
  lea dx, output_string
  int 21h

  mov ah, 09h
  lea dx, output
  mov bx, out_length
  int 21h

  mov al, 20h               ; корректное завершение обработчика
  out 20h, al               ; отправление команды завершения

  pop ax
  iret                  ; возврат из прерывания

SUBR_INT ENDP

END main
```