# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №4**

# по дисциплине «Организация систем и ЭВМ»

**Тема:**

«**Написание собственного прерывания и работа со строками**»

Студент гр. 3388 Потоцкий С.С.

Преподаватель Фирсов М.А.

Санкт-Петербург 2024

# Цель работы.

Изучить работу с прерываниями и обработку строк в ассемблере 8086. Написать программу, реализующую собственное прерывание 9h с обработкой строки.

# Задание.

Состоит из двух основных задач:

1) Реализация сценария работы с прерываниями в соответствии с вариантом.

2) Реализация преобразования строки с использованием команд работы со строками в соответствии с вариантом.

Вариант 16: 16о

|  |
| --- |
| 2) вып. работу.  3) Экспериментально выяснить, используется ли прерывание 16h (ввод символа с клавиатуры) при вводе строки с помощью прерывания 21h или наоборот. |

о) Символы входной строки передаются в выходную строку непосредственно, но если встретился символ "Q", то обработка строки завершается (вместо "Q" в выходную строку передаются завершающие байты).

Замечания:

1) В сценариях опущены:

– Вывод приветственного сообщения и ввод строки пользователем. Это 1-ый пункт основной программы каждого сценария (поэтому основная программа во всех сценариях начинается с пункта 2). Слова́ «Ожидание ввода строки», которые встречаются в таблице сценариев, – это всего лишь организация задержки с ожиданием нажатия клавиш, для ввода данных не используется.

– Действия по восстановлению изменённых прерываний, если они не требуются по сценарию специально. После завершения программы все изменённые прерывания всегда должны быть восстановлены (независимо от сценария). Перед завершением программы следует восстановить в том числе и те вектора прерываний, которые могли бы восстановиться по таймеру. После завершения программы DOS освобождает память, занимаемую программой, что может помешать корректному восстановлению векторов. Поэтому рассчитывать на восстановление по таймеру после завершения программы будет неправильно.

2) «Ожидание ввода строки» следует делать так, чтобы пользователь мог нажать множество клавиш, прежде чем решит завершить ввод нажатием Enter.

4) «вып. работу» означает, что надо на основе введённой строки (п. 1 основной программы) создать модифицированную строку и вывести её на экран. При выполнении преобразования нельзя портить исходную строку, результат преобразования должен записываться в выходную строку.

5) Перед запуском ожидания нажатия клавиши («ожидание ввода строки» в таблице) вывести сообщение об этом.

6) Перед заменой 9h следует сделать небольшую задержку (см. 3.2 «Задержка во времени»), чтобы предшествующая активность пользователя была обработана до того, как 9h будет изменён;

7) Для исключения возможного взаимного влияния системных и пользовательских прерываний рекомендуется отвести в программе под стек не менее 1Кбайт.

# Основные теоретические положения.

1. Краткие сведения о прерываниях.

Прерывание – это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств. Когда возникает прерывание, процессор прекращает выполнение текущей программы (если её приоритет ниже) и запоминает в стеке вместе с регистром флагов адрес возврата (CS:IP) – ме́ста, с которого будет продолжена прерванная программа. Затем в CS:IP загружается адрес программы обработки прерывания и ей передаётся 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.

В лабораторной работе среди действий по обработке прерывания могут быть:

– преобразование строки;

– вывод сообщения на экран;

– замена или восстановление вектора прерывания (другого или самого́ себя);

Программа, использующая новые программы обработки прерываний, при своём завершении должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H возвращает текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX.

Для задания адреса собственного прерывания с заданным номером в таблицу векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес нового обработчика.

При использовании функции 25H прерывания 21H DOS знает, что вы делаете, и гарантирует, что в момент записи прерывания будут заблокированы.

Прерывания бывают аппаратные (вызываемые в результате сигналов от оборудования) и программные (вызываемые в коде). В лабораторной работе предлагаются к замене вектора́ следующих прерываний:

– 09H – аппаратное прерывание, возникающее в результате нажатия клавиш клавиатуры;

– 16H – программное прерывание для ожидания ввода символа с клавиатуры;

– 21H – программное прерывание для вызова сервисов DOS.

Заменённое тело 9h следует завершать не выходом из прерывания (iret), а переходом к выполнению старого тела 9h (использовать команду jmp dword ptr), иначе обработка сигналов клавиатуры будет нарушена. То же самое касается прерываний 16h и 21h.

2. Краткие сведения о командах обработки строк.

Для обработки строковых данных ассемблер имеет пять групп команд обработки строк:

— MOVS — переслать один байт или одно слово из одной области памяти в другую;

Каждая команда имеет модификации, указывающие размер операнда: байт (B), слово (W), двойное слово (D). Например: MOVSB, MOVSW, MOVSD.

Эти команды предполагают, что регистры DI и SI содержат относительные адреса, указывающие на необходимые области памяти (для загрузки можно использовать команду LEA). Регистр SI обычно связан с регистром сегмента данных – DS:SI. Регистр DI всегда связан с регистром дополнительного сегмента – ES:DI. Следовательно, команды MOVS, STOS, CMPS и SCAS требуют инициализации регистра ES (обычно адресом в регистре DS).

Префиксы REP/REPE/REPZ/REPNE/REPNZ позволяют этим командам обрабатывать строки любой длины.

# Выполнение работы

1. **Основная процедура Main**  
   Программа начинается с вывода на экран приветственного сообщения, после чего пользователь вводит строку. Для корректной обработки происходит небольшая задержка. Затем оригинальные обработчики прерываний 16h и 21h сохраняются в специальные переменные (Oldint16H, Oldint21H и их сегменты), после чего заменяются на пользовательские обработчики (Newint16H и Newint21H).  
   Программа организует основной цикл, в котором пользователь вводит строку, она обрабатывается процедурой ProcessString, а результат выводится на экран. Также отображается статистика вызовов прерываний. Счетчики вызовов обнуляются перед повторением цикла, и буферы очищаются для корректной работы на следующей итерации. Выход из программы сопровождается восстановлением оригинальных обработчиков прерываний.

* **Обработчики прерываний Newint16H и Newint21H**  
  Пользовательские обработчики прерываний выполняют подсчет вызовов соответствующих прерываний. Для этого они увеличивают значения счетчиков callCount16H и callCount21H. После подсчета вызовов управление передается оригинальным обработчикам через сохраненные адреса.
* **Процедура ProcessString**  
  Эта процедура отвечает за обработку строки, введенной пользователем. Она копирует входную строку из буфера InputString в буфер OutputString, добавляя символ конца строки ($). Также реализована проверка на завершение обработки при обнаружении символа Q в строке. Для строковых операций используются регистры SI и DI, а их направление задается через команду CLD.
* **Процедура Delay**  
  Для предотвращения некорректной работы программы из-за быстрого выполнения предыдущих операций добавлена небольшая задержка, реализованная через BIOS-прерывание int 15h.
* **Процедура PrintNumber**  
  Эта процедура выводит число в десятичном формате. Она преобразует число в последовательность ASCII-символов и выводит их на экран с помощью функции DOS (int 21h, функция 02h).
* **Процедура Restoreinterrupts**  
  При завершении работы программы происходит восстановление оригинальных обработчиков прерываний int 16h и int 21h. Это гарантирует, что система возвращается в изначальное состояние.
* **Тестирование программы**

# 

# Рисунок 1 — Остановка программы в ожидании нажатия клавиши

# 

# Рисунок 2 — Успешное продолжение и завершение выполнения после нажатия клавиши

# 4. Тестирование различных входных данных

|  |  |
| --- | --- |
| Input | Output |
| Hello World | Hello World |
| Hello Q World | Hello |
| Hello WorldQ | Hello World |
| QLQLQL |  |
| LQLQLQ | L |
| QQQQQ |  |
| abrala | abrala |

**Вывод**

В ходе лабораторной работы были изучены прерывания и работа со строками на языке Ассемблер Intel 8086. А также эксперементально выяснено, что при вводе строки с помощью прерывания используется прерывание 16h (ввод символа с клавиатуры), но наоборот не происходит.

# ПРИЛОЖЕНИЕ

# ИСХОДНЫЙ КОД ПРОГРАММЫ

1. lr4.asm

­

.MODEL SMALL

.STACK 400h

.DATA

WelcomeMessage DB 0Dh, 0ah, 'Write input string: $' ; Сообщение приветствия

ResultMessage DB 0Dh, 0ah, 'Result: $' ; Сообщение результата

InputString DB 80h, 0, 80 DUP('$') ; Входная строка

OutputString DB 80 DUP('$') ; Выходная строка

callCount16H DW 0 ; Счетчик вызовов прерывания 16H

callCount21H DW 0 ; Счетчик вызовов прерывания 21H

Oldint16H DW ? ; Старый адрес обработчика прерывания 16H

Oldint16HSeg DW ? ; Сегмент старого обработчика прерывания 16H

Oldint21H DW ? ; Старый адрес обработчика прерывания 21H

Oldint21HSeg DW ? ; Сегмент старого обработчика прерывания 21H

callCountsMessage DB 0Dh, 0ah, 'Call counts: $' ; Сообщение о количестве вызовов

Count16Message DB 0Dh, 0ah, '16H: $' ; Сообщение о количестве вызовов 16H

Count21Message DB 0Dh, 0ah, '21H: $' ; Сообщение о количестве вызовов 21H

.CODE

START:

mov ax, @DATA

mov ds, ax ; Инициализация сегмента данных

; Сохранение старого обработчика прерывания 16H

mov ax, 3516h

int 21h

mov Oldint16H, bx

mov Oldint16HSeg, es

; Сохранение старого обработчика прерывания 21H

mov ax, 3521h

int 21h

mov Oldint21H, bx

mov Oldint21HSeg, es

call Delay ; Вызов процедуры задержки

; Установка нового обработчика прерывания 16H

push ds

mov dx, OFFSET Newint16H

mov ax, SEG Newint16H

mov ds, ax

mov ax, 2516h

int 21h

pop ds

; Установка нового обработчика прерывания 21H

push ds

mov dx, OFFSET Newint21H

mov ax, SEG Newint21H

mov ds, ax

mov ax, 2521h

int 21h

pop ds

; Вывод сообщения приветствия

mov ah, 09h

lea dx, WelcomeMessage

int 21h

; Ввод строки

mov ah, 0ah

lea dx, InputString

int 21h

call ProcessString ; Обработка строки

; Вывод сообщения результата и обработанной строки

mov ah, 09h

lea dx, ResultMessage

int 21h

lea dx, OutputString

int 21h

; Вывод количества вызовов прерываний

mov ah, 09h

lea dx, callCountsMessage

int 21h

lea dx, Count16Message

int 21h

mov ax, callCount16H

call PrintNumber

mov ah, 09h

lea dx, Count21Message

int 21h

mov ax, callCount21H

call PrintNumber

; Вывод новой строки

mov ah, 02h

mov dl, 0Dh

int 21h

mov dl, 0Ah

int 21h

; Восстановление старых обработчиков прерываний и завершение программы

pushf

call RestoreInterrupts

popf

mov ax, 4C00h

int 21h

Delay PROC NEAR

push cx

push dx

push ax

mov cx, 0eh

mov dx, 0ffffh

mov ah, 86h

int 15h

pop ax

pop dx

pop cx

ret

Delay ENDP

RestoreInterrupts PROC

pushf

push ax

push bx

push cx

push dx

push ds

push es

; Восстановление старого обработчика прерывания 21H

pushf

push ds

mov dx, Oldint21H

mov ax, Oldint21HSeg

mov ds, ax

mov ah, 25h

mov al, 21h

int 21h

pop ds

; Восстановление старого обработчика прерывания 16H

push ds

mov dx, Oldint16H

mov ax, Oldint16HSeg

mov ds, ax

mov ah, 25h

mov al, 16h

int 21h

pop ds

popf

mov ax, @DATA

mov ds, ax

pop es

pop ds

pop dx

pop cx

pop bx

pop ax

popf

ret

RestoreInterrupts ENDP

Newint16H PROC FAR

pushf

push ax

push ds

mov ax, @DATA

mov ds, ax

inc callCount16H ; Увеличение счетчика вызовов прерывания 16H

pop ds

pop ax

popf

jmp dword ptr cs:[Oldint16H] ; Переход к старому обработчику прерывания 16H

Newint16H ENDP

Newint21H PROC FAR

pushf

push ax

push ds

mov ax, @DATA

mov ds, ax

inc callCount21H ; Увеличение счетчика вызовов прерывания 21H

pop ds

pop ax

popf

jmp dword ptr cs:[Oldint21H] ; Переход к старому обработчику прерывания 21H

Newint21H ENDP

ProcessString PROC

push ds

pop es

mov cl, [InputString + 1] ; Длина входной строки

lea si, InputString + 2 ; Указатель на начало входной строки

lea di, OutputString ; Указатель на начало выходной строки

cld

cmp cl, 0

je EndProcessing

NextChar:

lodsb

cmp al, 'Q' ; Проверка на символ 'Q'

je EndProcessing

stosb

loop NextChar

EndProcessing:

mov al, '$'

stosb

ret

ProcessString ENDP

PrintNumber PROC

push ax

push bx

push cx

mov bx, 10

xor cx, cx

cmp ax, 0

JNE ConvertLoop

mov dl, '0'

mov ah, 02h

int 21h

JMP DonePrinting

ConvertLoop:

xor dx, dx

div bx

push dx

inc cx

cmp ax, 0

JNE ConvertLoop

PrintLoop:

pop dx

ADD dl, '0'

mov ah, 02h

int 21h

LOOP PrintLoop

DonePrinting:

pop cx

pop bx

pop ax

ret

PrintNumber ENDP

END START