**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 и работа со строками.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Лексин М.В. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2024

## Цель работы

Изучить теоретическую информацию о прерываниях и строках в языке Ассемблера, научиться работать с прерываниями и обрабатывать строки. Создать программу, в которой будет обработка строки согласно заданному условию и замена прерывания с последующим восстановлением.

## Задание.

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 12.

Шифр 12к.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № сц. | Действия основной программы | Действия 1сh | Действия 9h | Действия 4bh |
| 12 | 2) замена 9h;  3) ожидание ввода строки. | а) 82 запуска;  б) вып. работу;  в) восст. себя. | 1) inc счётчик вызовов 9h;  2) Если счётчик нечётный, то заменить или восст. 1ch.\*\* |  |

\*\* Замена 1ch происходит на вызовах 1, 5, 9, 13 и т.д., а восстановление 1ch – на вызовах 3, 7, 11, 15 и т.д. При повторной замене 1ch состояние не сбрасывается (т.е. сохраняется счётчик выполненных запусков 1ch).

к) Заменить введённые во входной строке латинские буквы на числа, соответствующие их номеру по алфавиту, представленному в шестнадцатеричной СС.

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.

3) Для 1ch описан не алгоритм работы одного вызова, а последовательность действий, выполняемая в результате множества вызовов. Время должно измеряться через подсчёт количества выполненных вызовов, функцию задержки использовать не следует.

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.

8) При выполнении преобразования обязательно использовать команды работы со строками.

9) В заданиях а, г, й, к, т, у, э следует разделять пробелом результирующие числа.

10) Завершающие символы (0ah, 0dh, «$») рассматривать именно как завершающие, т.е. не подвергать преобразованиям; закончить ими выходную строку.

**Основные теоретические положения**

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 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.

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.

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

– 1CH – можно рассматривать их как аппаратное прерывание, генерируемое системным таймером; генерируется автоматически 18,2 раза в сек.;

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

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.

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

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

— LODS — загрузить из памяти один байт в регистр AL или одно слово в регистр AX;

— STOS — записать содержимое регистра AL или AX в память;

— CMPS — сравнить содержимое двух областей памяти, размером в один байт или в одно слово;

— SCAS — сравнить содержимое регистра AL или AX с содержимым памяти.

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).

**Ход работы**

1. Реализация сценария работы с прерываниями.

В начале программа загружает сегмент данных в регистр DS с помощью инструкции mov ax, @data. Затем с помощью процедуры writeMsg выводится приветственное сообщение, используя функцию DOS int 21h с функцией ah = 9h, которая выводит строку на экран. После приветствия программа переходит к вводу строки, который осуществляется через процедуру stringRead, использующую функцию DOS int 21h с ah = 0Ah для чтения строки из ввода. Введенная строка сохраняется, и оригинальный обработчик прерывания 9h сохраняется. Затем происходит задержка на 14 \* 65535 мкс, после чего устанавливается новый обработчик для прерывания 9h.

Обработчик прерывания 9h (handler9h) выполняет следующие действия:

1. При каждом вызове обработчик 9h увеличивает счетчик вызовов 9h.
2. Если счетчик вызовов нечетный, производится либо замена, либо восстановление обработчика 1Ch: Замена обработчика 1Ch происходит на 1-м, 5-м, 9-м, 13-м и т.д. вызовах 9h. Восстановление обработчика 1Ch выполняется на 3-м, 7-м, 11-м, 15-м и т.д. вызовах 9h.

Обработчик прерывания 1Ch (handler1Ch) выполняет такие действия:

а) Выполняется 82 запуска обработчика.

б) После 82-го запуска выполняется заданная работа.

в) Затем обработчик восстанавливает себя для дальнейшего использования.

Для замены обработчика прерывания используется прерывание int 21h с функцией ah = 25h, позволяющее установить новый обработчик. Оригинальные векторы прерывания сохраняются заранее через int 21h с ah = 35h.

1. Реализация преобразования строки с использованием команд работы со строками.

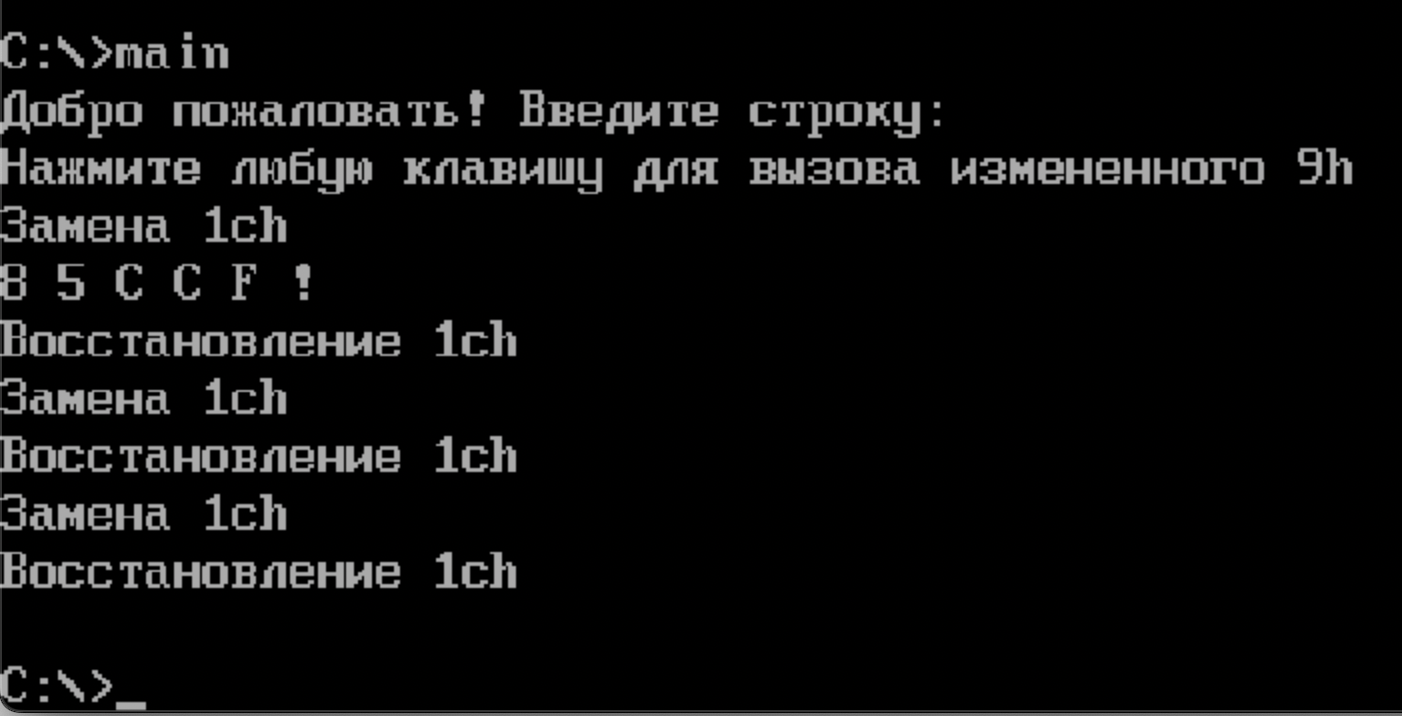
Процедура convertLettersDig преобразует строку по следующему принципу:

1. Проходит по каждому символу строки.
2. Если символ — буква от 'a' до 'i', он преобразуется в соответствующее число от 0 до 9.
3. Если символ — буква от 'o' до 'z', он преобразуется в число от 10 до 26.
4. Прочие символы остаются без изменений.

Для преобразования символов в числа используется операция sub, которая вычитает из кода символа значение, соответствующее нужной цифре. Например, 'a' преобразуется в 1, 'b' в 2 и так далее. После завершения преобразования преобразованная строка выводится на экран с помощью writeMsg, которая использует int 21h с ah = 9h для отображения строки.

1. Тестирование сценария работы с прерываниями.

После ввода строки, нажатия любой клавиши и ожидания 82-го вызова прерывания 1Ch строка будет преобразована и выведена на экран. При нажатии клавиши enter программа завершится, восстановив все прерывания. Если после ввода строки и нажатия клавиши не дожидаться 82-го вызова 1Ch и восстановить прерывание нажатием еще раз на клавишу, то строка не будет преобразована.



1. Тестирование преобразования строки. Результаты тестирования представлены в Таблице 1.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Ожидаемые результаты | Полученный результат |
| 1 | Hello! | 8 5 C C F ! | 8 5 C C F ! |
| 2 | abcde | 1 2 3 4 5 | 1 2 3 4 5 |
| 3 | ZzZhcfe | 1A 1A 1A8 3 6 5 | 1A 1A 1A8 3 6 5 |
| 4 | Ako!@\_! | 1 B F !@\_! | 1 B F !@\_! |

## Выводы

В ходе выполнения лабораторной работы был изучен теоретический материал о прерываниях и работе со строками на языке Ассемблер. Также разработано собственное прерывание и реализована работа со строкой согласно заданию.

# Приложение

Название файла: main.asm

.model small

.stack 400h

.data

keep\_ip\_1ch DW 0 ; для хранения смещения исходного 1ch

keep\_cs\_1ch DW 0 ; для хранения сегмента исходного 1ch

keep\_ip\_9h DW 0 ; для хранения смещения исходного 09h

keep\_cs\_9h DW 0 ; для хранения сегмента исходного 09h

EOFLine EQU '$' ; Определение символьной константы 'Конец строки';

message db "Добро пожаловать! Введите строку: ", 0ah, 0dh, EOFLine

change db "Нажмите любую клавишу для вызова измененного 9h", 0ah, 0dh, EOFLine

waiting db "Замена 1ch", 0ah, 0dh, EOFLine

recovering db "Восстановление 1ch", 0ah, 0dh, EOFLine

input\_head DB 64H, 0 ; заголовок строки: можно ввести максимум 64h=100 символов.

; В след байте вместо 0 появится фактическое количество введённых символов

input DB 64H DUP('\*'), 0ah, 0dh, EOFLine ; Буфер памяти для введённых символов

; плюс байты для корректного завершения вывода

output DB 64H DUP('\*'), 0ah, 0dh, EOFLine ; Буфер памяти для обработанных символов

; плюс байты для корректного завершения вывода

counter1ch dw 0 ; счетчик вызовов 1ch

counter9h dw 0 ; счетчик вызовов 9h

.code

mov ax, @data

mov ds, ax

; вывод приветствия

mov dx, OFFSET message

call writeMsg

; ввод строки пользователем

mov dx, OFFSET input\_head

call stringRead

; сохранение позиций старого обработчика прерывания 9h

mov ah, 35h ; функция получения вектора

mov al, 9h ; номер вектора

int 21h

mov keep\_cs\_9h, es ; запоминание сегмента оригинального 9h

mov keep\_ip\_9h, bx ; запоминание смещения оригинального 9h

; задержка во времени

mov cx, 0eh ; 14 \* 65535 мкс задержки

mov dx, 0ffffh ; ещё 65535 мкс задержки

mov ah, 86h ; функция "ждать"

int 15h ; вызов функции ожидания

; замена обработчика прерывания 9h

push ds

mov dx, offset handler9h ; смещение до процедуры в dx handler9h (нового обработчика 9h)

mov ax, seg handler9h ; сегмент процедуры в ax

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 9h ; номер вектора

int 21h ; замена прерывания

pop ds

; сообщение об ожидании

lea dx, change

call writeMsg

; ожидание нажатия Enter

wait\_for\_enter:

mov ah, 0h ; функция "проверка клавиши"

int 16h ; вызываем прерывание для чтения клавиши

cmp ah, 1Ch ; проверяем, что нажата клавиша Enter (скан-код 1c)

jne wait\_for\_enter ; если нет, продолжаем ожидание

key\_pressed:

; восстановление 9h

push ds

mov dx, keep\_ip\_9h ; смещение для оригинального обработчика 9h

mov ax, keep\_cs\_9h ; сегмент оригинального обработчика

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 9h ; номер вектора

int 21h ; замена прерывания

pop ds

; восстановление 1ch

push ds

mov dx, keep\_ip\_1ch ; смещение до оригинального обработчика 1ch

mov ax, keep\_cs\_1ch ; сегмент оригинального обработчика

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 1ch ; номер вектора

int 21h ; замена прерывания

pop ds

; завершение программы

mov ah, 4ch

int 21h

StringRead PROC

; запоминание изменеяемых регистров

push ax

push bp

push bx

mov ah, 0ah ; функция ввода строка

push dx ; смещение заголовка строки

int 21h ; вызов функции DOS ввода строки

pop bp ; поместить в bp

xor bx, bx ; обнуление bx

mov BL, ds:[bp + 1] ; теперь в bx количество введенных символов

add bx, bp ; теперь bx указывает на конечный введённый символ

add bx, 2 ; теперь bx указывает на байт, следующий за финальным 0dh

mov word ptr [bx + 1], 240ah ; добавить в конец 0ah и '$'

; возвращение сохраненных регистров

pop bx

pop bp

pop ax

RET

StringRead ENDP

writeMsg proc

push ax

mov ah, 9h

int 21h

pop ax

ret

writeMsg ENDP

convertLettersDig proc

; запоминание изменеяемых регистров

push si

push di

push bx

push cx

push ax

push es

mov ax, @data

mov ds, ax

mov ES, AX

cld ; Сброс флага DF для направления слева направо (для последовательного перебора строки)

lea di, output ; Адрес области "куда"

lea si, input ; Адрес области "откуда"

xor cx, cx ; обнуление регистра cx

mov cl, input\_head[1] ; сх - размер строки

add cl, 3 ; + дополнительные символы

convert\_loop:

lodsb ; загрузить из памяти один байт в регистр al

mov bl, al ; bl = al

or bl, 20h ; логическое или: если буква в верхнем регистре, то она приведется к нижнему

; остальные символы свдинутся на 32 бита

cmp bl, 'a' ; проверка символа на латинскую букву

jb next ; если ниже a - то не буква, переход к next

cmp bl, 'z'

ja next ; если выше z - то не буква, переход к next

or al, 20h ; логическое или: буква в верхнем регистре, приводится к нижнему

cmp al, 'i' ; если буква выше i, то переход к метке second

ja second

sub al, 30h ; если буква от a до i, то вычитание 30h от a до i - соответствие цифрам, начиная с 1

jmp store\_char ; переход к записи числа

second:

cmp al, 'o' ; если буква выше o, то переход к метке third

ja third

sub al, 29h ; если буква от j до o то вычесть 29h (от j до o - числа от A до F)

jmp store\_char ; переход к записи числа

third:

mov bl, al ; от p до z число содержит первой цифрой "1"

mov al, '1'

stosb ; запись "1"

mov al, bl

cmp al, 'z' ; если z, то переход к метке z

jz z

sub al, 40h ; если буква от p до y, то вычесть 40h, то есть вторая цифра числа от 1 до 9

jmp store\_char ; переход к записи числа

z:

mov al, 'A' ; у номер буквы 'z' вторая цифра - A

jmp store\_char ; переход к записи числа

store\_char:

stosb ; число записывается с пробелом, здесь происходит запись числа из регистра al

mov al, ' ' ; в al - пробел

next:

stosb ; запись в память из регистра al

loop convert\_loop ; переход к следующей итерации

; вывод преобразованной строки

mov dx, offset output

call writeMsg

; возвращение сохраненных регистров

pop es

pop ax

pop cx

pop bx

pop di

pop si

ret

convertLettersDig endp

; обработчик прерывания 1Ch

handler1ch proc

inc counter1ch ; увеличение количество запусков

cmp counter1ch, 82 ; если не 82 запуска, то переход к концу

jne skip

call convertLettersDig ; если не 82 запуска, то вызов преобразования строки

; восстановление 1ch

push ds

mov dx, keep\_ip\_1ch ; смещение для оригинального обработчика 1ch

mov ax, keep\_cs\_1ch ; сегмент оригинального обработчика

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 1ch ; номер вектора

int 21h ; замена прерывания

pop ds

skip:

iret

handler1ch endp

; обработчик прерывания 9h

handler9h proc

; сохранение изменяемых регистров

push ax

push es

push bx

push ds

push dx

inc counter9h ; увеличение количество запусков

; Проверяем остаток от деления на 4

mov ax, counter9h

and ax, 3 ; остаток от деления на 4 (маска 0x3)

cmp ax, 1

je changer ; если остаток 1, выполняем замену 1ch

cmp ax, 3

je rec ; если остаток 3, восстанавливаем 1ch

jmp skip\_to\_ret ; иначе пропускаем

changer:

; сохранение позиций старого обработчика прерывания 1ch

mov ah, 35h ; функция получения вектора

mov al, 1ch ; номер вектора

int 21h

mov keep\_cs\_1ch, es ; запоминание сегмента оригинального обработчика

mov keep\_ip\_1ch, bx ; запоминание смещения оригинального обработчика

; замена обработчика прерывания 1ch

push ds

mov dx, offset handler1ch ; смещение для процедуры в dx handler1ch - новый обработчик прерывания 1ch

mov ax, seg handler1ch ; сегмент процедуры в ax

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 1ch ; номер вектора

int 21h ; замена прерывания

pop ds

; сообщение об ожидании

lea dx, waiting

call writeMsg

jmp skip\_to\_ret

rec:

; восстановление 1ch

push ds

mov dx, keep\_ip\_1ch ; смещение для оригинального обработчика 1ch

mov ax, keep\_cs\_1ch ; сегмент оригинального обработчика

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 1ch ; номер вектора

int 21h ; замена прерывания

pop ds

; сообщение об ожидании

lea dx, recovering

call writeMsg

skip\_to\_ret:

; возвращение сохраненных регистров

pop dx

pop ds

pop bx

pop es

pop ax

jmp dword ptr [keep\_ip\_9h] ; переход к выполнению оригинального кода обработки прерывания 9h

handler9h endp

end