**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 |  | Павлов А.Р. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2024

## **Цель работы.**

Изучить прерывания и работу со строками в языке Ассемблер. Написать собственное прерывание.

## **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 14.

Шифр 14м.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № сц. | Действия основной программы | Действия 1сh | Действия 9h | Действия 4bh |
| 14 | 2) замена 9h;  3) замена 4bh;  4) ожидание (5 сек.);  5) вызов 4bh. |  | отметить, что был вызван. | вып. работу, если ранее был вызван 9h. |

м) В выходную строку передаются сначала все цифры из входной строки, а потом все остальные символы. Порядок цифр сохраняется, а порядок остальных символов инвертируется.

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.

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.

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.

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:

– 4BH – зарезервированное прерывание, которое при выполнении лабораторной работы можно использовать в качестве пользовательского программного прерывания;

– 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).

**ПРОТОКОЛ**

1. Реализация сценария работы с прерываниями.

В основной процедуре Main происходит вывод вспомогательных сообщений и ввод строки пользователем. Далее с помощью кода, данного в теоретических положениях, происходит сохранение оригинальных прерываний 9h и 4bh в переменные памяти, также происходит их замена на пользовательские, реализованное в функциях handler9h и handler4bh, соответствующее условию варианта. Первое записывает в память отметку о своём вызове, а второе — если отметка о вызове есть, вызывает процедуру работы со строкой stringReverse. После этого вызывается функция 83h прерывания 15h для асинхронного ожидания ввода клавиши в течении 5 секунд, если оно проихошло — то выхывается прерывание 9h. После обработки и вывода изменённой строки восстанавливаются оригинальные прерывания и происходит выход их программы.

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

В процедуре stringReverse происходит обработка строки, введенной пользователем. В регистры ES и DS сохраняется адрес сегмента данных, это необходимо, так как регистры SI и DI, необходимые для использования LOADB и STOSB, связаны с ними. После этого за один проход формируется новая строка: если текущий символ — число, то он сразу записывается в новую строку, в противном случае — мы сохраняем его в стек. После обработки всех строки на входе, строка на выходе последовательно дополняется символами из стека: так как они сохранялись в прямом порядке, то на выходе будут в обратном (принцип рабоыт стека FILO).

1. Тестирование прерывания и обработки строки.

После ввода строки и ожидания в 5 секунд происходит вывод изменённой строки согласно заданию.

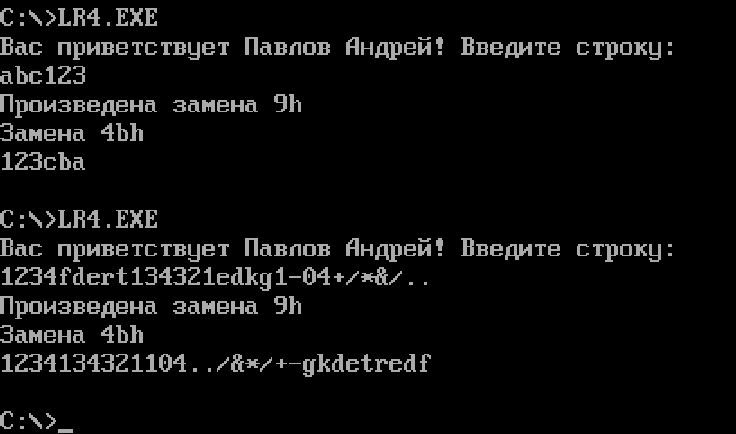


Рисунок 1 — Обработка строки (2 варианта входных данных)

Прерывание 4bh не отработает, согласно условию, если предварительно не было вызвано 9h. Ниже приведён пример рабоыт программы в таком случае.

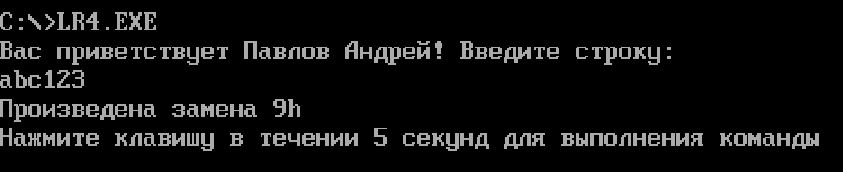


Рисунок 2 — Работа прерывания 4bh без предварительного вызова 9h

Примером успешной работы прерываний является корректная работа программы.

1. Тестирование функции обработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Ожидаемые результаты | Полученный результат |
| 1 | abc | cba | cba |
| 2 | 123 | 123 | 123 |
| 3 | abc123 | 123cba | 123cba |
| 4 | 123abc456\*&^ | 123456^&\*cba | 123456^&\*cba |

## **Выводы.**

В ходе лабораторной работы были изучены прерывания и работа со строками на языке Ассемблер. Разработаны собственные прерывания.

# **ПРИЛОЖЕНИЕ**

Название файла: lr4.asm

ASSUME CS:CODE, SS:AStack, DS:DATA

AStack SEGMENT STACK

DW 400h DUP(0) ; 1 КБ стека

AStack ENDS

DATA SEGMENT

keep\_ip\_4bh DW 0 ; для хранения смещения

keep\_cs\_4bh DW 0 ; для хранения сегмента

keep\_ip\_9h DW 0 ; для хранения смещения

keep\_cs\_9h DW 0 ; для хранения сегмента

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

message db "Вас приветствует Павлов Андрей! Введите строку: ", 0ah, 0dh, EOFLine

change db "Произведена замена 9h", 0ah, 0dh, EOFLine

change4 db "Замена 4bh", 0ah, 0dh, EOFLine

new\_line DB 0Dh, 0Ah, '$'

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

counter9h dw 0 ; счетчик вызовов 9h

DATA ENDS

CODE SEGMENT

StringRead PROC FAR

; запоминание изменеяемых регистров

push ax

push bp

push bx

mov ah, 0ah ; функция ввода строки

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 NEAR

; запоминание изменеяемых регистров

push ax

mov ah, 9h

int 21h

pop ax

; возвращение сохраненных регистров

ret

writeMsg ENDP

; ----------------------------------------------------------

stringReverse PROC FAR

; запоминание изменеяемых регистров

push ax

push bx

push cx

push dx

push si

push di

push es

push ds

; загрузка сегмента данных в регистр dx

mov ax, DATA

mov ds, ax

mov es, ax ; ???

cld ; Сброс флага DF для направления слева направо

lea si, input ; Адрес области "откуда"

lea di, output ; Адрес области "куда"

xor cx, cx ; обнуление регистра cx

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

;add cl, 3 ; + дополнительные символы

xor bx, bx ; обнуление регистра bx

convert\_loop:

lodsb ; загрузить из памяти один байт в регистр al

; проверка символа на цифру

cmp al, 30h

jb store ; если ниже 0 - то не цифра, в стек

cmp al, 39h

ja store ; если выше 9 - то не цифра, в стек

stosb ; запись в память из регистра al

next:

loop convert\_loop ; переход к следующей итерации

jmp sr

store:

inc bx

push ax

jmp next

sr:

; возвращаем в строку все НЕцифры из стека

; так как заполняли прямым ходом -- вернётся обратным

mov cx, bx

stack\_return:

pop ax

stosb ; запись в память из регистра al

loop stack\_return ; переход к следующей итерации

exit:

; Помещаем сивол конца строки

mov al, 0Dh

stosb

mov al, 0Ah

stosb

mov al, EOFLine

stosb

; вывод преобразованной строки

mov dx, offset output

call writeMsg

; восстановление содержимого регистров

pop ds

pop es

pop di

pop si

pop dx

pop cx

pop bx

pop ax

ret

stringReverse ENDP

; ----------------------------------------------------------

; обработчик прерывания 4Bh

handler4bh PROC FAR

cmp counter9h, 1 ; если 9h не запускался, то переход к концу

jne skip

call stringReverse

skip:

iret

handler4bh ENDP

; обработчик прерывания 9h

handler9h PROC FAR

inc counter9h ; увеличение количество запусков

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

handler9h ENDP

; ----------------------------------------------------------

Main PROC FAR

mov ax, DATA

mov ds, ax ; соответствие регистру ds - сегмента данных

; вывод приветствия

mov dx, OFFSET message

call writeMsg

; ввод строки пользователем

mov dx, OFFSET input\_head

call StringRead

mov dx, OFFSET new\_line

call writeMsg

; 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 обработчика

mov keep\_ip\_9h, bx ; запоминание смещения оригинального обработчика

; задержка во времени

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

mov dx, 424fh ; ещё 16975 мкс задержки (суммарно 1с)

mov ah, 86h ; функция "ждать"

int 15h ; вызов функции ожидания

; замена обработчика прерывания 9h

push ds

mov dx, offset handler9h ; смещение для процедуры в dx handler9h - новый обработчик прерывания 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 о замене

lea dx, change

call writeMsg

; замена обработчика прерывания 4bh

push ds

mov dx, offset handler4bh ; смещение для процедуры в dx handler4bh - новый обработчик прерывания 4bh

mov ax, seg handler4bh ; сегмент процедуры в ax

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 4bh ; номер вектора

int 21h ; замена прерывания

pop ds

; сообщение о замене

lea dx, change4

call writeMsg

mov cx, 4ch ; 76 \* 65535 мкс задержки

mov dx, 4bc8h ; ещё 19340 мкс задержки (суммарно 5с)

mov ah, 86h ; функция "ждать"

int 15h ; вызов функции ожидания

; Основная функция

int 9h

int 4bh

; 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 4bh

push ds

mov dx, keep\_ip\_4bh ; смещение для оригинального обработчика 4bh

mov ax, keep\_cs\_4bh ; сегмент оригинального обработчика

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 4bh ; номер вектора

int 21h ; замена прерывания

pop ds

; завершение программы

mov ah, 4ch

int 21h

Main ENDP

CODE ENDS

END Main