**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Организация ЭВМ и систем»**

# **Тема: Написание собственного прерывания и работа со строками на языке ассемблера.**

|  |  |
| --- | --- |
| Студент гр. 3382 | Копасова К. А. |
| Преподаватель | Фирсов М. А. |

Санкт-Петербург

2024

**Цель работы:** Изучить обработку прерываний в DOS и научиться использовать команды для работы со строками для преобразования данных.

**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.

**Сценарии работы с прерываниями.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № сц. | Действия основной программы | Действия 1сh | Действия 9h | Действия 4bh |
| 6 | 2) замена 1ch;  3) ожидание ввода строки;  4) вызов 4bh. | после замены через 3 сек. заменяет 4bh и восст. себя. |  | 1) вып. работу;  2) восст. себя. |

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.

1. «Ожидание ввода строки» следует делать так, чтобы пользователь мог нажать множество клавиш, прежде чем решит завершить ввод нажатием Enter.
2. 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.
3. «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.
4. Перед запуском ожидания нажатия клавиши («ожидание ввода строки» в таблице) вывести сообщение об этом.
5. Для исключения возможного взаимного влияния системных и пользовательских прерываний рекомендуется отвести в программе под стек не менее 1Кбайт.

**Варианты преобразования строк:**

е) Исключение цифр, введённых во входной строке, при формировании выходной строки.

Замечания:

1. При выполнении преобразования обязательно использовать команды работы со строками.
2. Завершающие символы (0ah, 0dh, «$») рассматривать именно как завершающие, т.е. не подвергать преобразованиям; закончить ими выходную строку.

Шифр задания варианта 6: 6е.

**Теоретический материал**

**Прерывания**

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.

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 его как аппаратное прерывание, генерируемое системным таймером; генерируется автоматически 18,2 раза в сек.;

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

– 21H – программное прерывание для вызова сервисов DOS.

**Обработка строк**

Для обработки строковых данных в ассемблере моя программа имеет несколько команд обработки строк:

— STOSB — записывает байт в строку данных;

— LODSB — загрузить из памяти один байт в регистр AL.

**Выполнение работы**

1. ***Реализация сценария работы с прерываниями в соответствии с вариантом.***
2. Реализована замена прерываний в основном код

Сохранение прерывания 4bh:

С помощью функции 354bh прерывания int 21h был сохранен изначальный вектор прерывания 4bh.

Сохранение и замена прерывания 1ch:

С помощью функций 351ch и 251ch прерывания int 21h был сохранен изначальный вектор прерывания 1ch и установлен новый вектор, указывающий на процедуру new\_int\_1ch.

Процедура new\_int\_1ch:

Данная процедура является обработчиком прерывания таймера 1ch, который вызывается примерно 18,2 раза в секунду. Инкрементируются два счётчика: timer\_counter, который отслеживает общее количество вызовов прерывания, и tick\_counter, который фиксирует количество тиков, прошедших с момента установки текущего обработчика. Затем выполняется проверка, прошло ли три секунды (это приблизительно 55 тика (18.2 \* 3)). Если прошло меньше времени, процедура завершает работу, восстанавливая сохранённые регистры и передавая управление обратно вызвавшему процессу.

Если прошло три секунды, проверяется, был ли обработчик прерывания 4bh уже изменён (проверяется флаг int\_4bh\_change). Если он не был изменён, устанавливается новый обработчик 4bh, а флаг int\_4bh\_change обновляется, чтобы избежать повторной установки. Далее проверяется, был ли восстановлен исходный обработчик прерывания 1сh (проверяется флаг int\_1ch\_restored). Если он ещё не был восстановлен, процедура возвращает управление на сохранённый исходный обработчик и устанавливает флаг int\_1ch\_restored, указывая, что обработчик 1ch был возвращён в исходное состояние.

В завершение процедура восстанавливает сохранённые значения регистров AX и DS, чтобы вернуть процессор в состояние, в котором он находился до вызова прерывания. Завершается обработчик с помощью команды IRET, которая возвращает управление программе, вызвавшей прерывание.

Замена прерывания 4bh:

С помощью функции 254bh прерывания int 21h был установлен новый вектор 4bh, указывающий на процедуру new\_int\_4bh.

Ожидание ввода строки (сигнал к обработке строки):

На экран было выведено сообщение “Press enter for result: ”. Программа продолжала продолжала ввод, пока не была нажата клавиша Enter.

После этого срабатывало новое пользовательское прерывание 4bh, которое выполняло работу со строкой (удаляла из строки все цифры) и восстанавливало само себя.

Процедура new\_int\_4bh:

Данная процедура является обработчиком прерывания 4bh, который удаляет все цифры из заданной входной строки input\_str и сохраняет результат в output\_str. Сначала она сохраняет значения регистров, чтобы не нарушить работу программы, а затем получает длину строки и настраивает указатели для её обработки.

Процедура с помощью цикла обрабатывает символы строки: проверяет, является ли текущий символ цифрой. Если это цифра, она пропускается; если нет, символ копируется в выходную строку. Когда строка полностью обработана, в конец выходной строки добавляются завершающие символы (\r, \n, и $), после чего результат выводится на экран.

В завершение процедура восстанавливает обработчик прерывания 4bh, восстанавливает сохранённые регистры и завершает свою работу командой IRET.

1. ***Реализация преобразования строки с использованием команды работы со строками в соответствии с вариантом.***

В задании требовалось обработать входную строку так, чтобы на выходе получилась строка, которая не содержит цифр. Были инициализированы указатели на входную (input\_str) и выходную (output\_str) строки и счетчик символов.

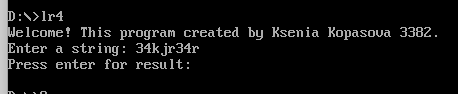
Программа проходилась по всем символам строки input\_str и проверяла следующие случаи:

1. Если символ меньше (больше) нуля (девяти) (т. е. код символа из таблицы ASCII меньше (больше), чем код нуля (девяти) из таблицы ASCII), то данный символ не является цифрой - а значит записываем этот символ в выходную строку и проходимся по символам дальше, пока они есть.
2. Если же символ больше кода нуля и меньше кода девятки, значит данный символ является числом - а значит декрементируем счетчик и переходим к следующему символу, никуда не записывая данную цифру.

Когда строка закончится, добавляем в конец символы возврата каретки (0dh), перевода курсора (0ah) и конца строки (‘$’). После выводим итоговую строку с помощью функции вывода строки 09h прерывания int 21h.

1. ***Тестирование сценария работы с прерываниями***

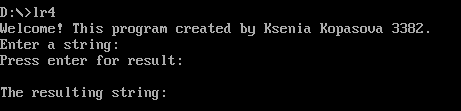
**Проверка восстановления обработчиков:**

Тестирование прерывания, когда прошло меньше 3-х секунд:

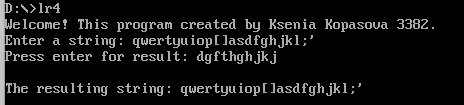
Тестирование прерывания, когда прошло больше 3-х секунд:

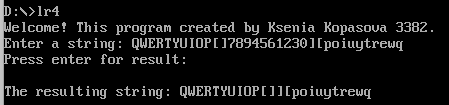
1. ***Тестирование преобразования строки***

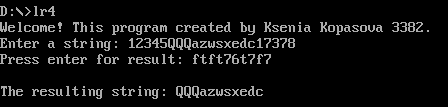
Программа была протестирована на разных входных данных, включая пустую строку, строку только из цифр, строку только из букв и строку из символов. Проверена корректность удаления цифр из входной строки.











|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Входные данные | Выходные данные | Комментарий |
| 1 |  |  | Верно |
| 2 | 1234567890123456789012345678901234567890 |  | Верно |
| 3 | qwertyuiop[]asdfghjkl;’ | qwertyuiop[]asdfghjkl;’ | Верно |
| 4 | QWERTYUIOP[]7894561230][poiuytrewq | QWERTYUIOP[]][poiuytrewq | Верно |
| 5 | 12345QQQazwsxedc17378 | QQQazwsxedc | Верно |

**Вывод**

В ходе лабораторной работы удалось изучить прерывания и функции со строками на языке ассемблера. Также был реализован сценарий работы с прерываниями, включая замену векторов прерываний и их восстановление по таймеру. Было реализовано преобразование строки, удаляющее цифры из входной строки.

**Приложение 1**

Файл LR4.asm:

.model small

.stack 1024 ; 1KB стека

.data

; строки приветствия, ожидания, ввода и вывода

msg\_greeting db 'Welcome! This program created by Ksenia Kopasova 3382.$'

msg\_enter\_string db 0Dh, 0Ah, 'Enter a string: $' ; для организации задержки, для ввода данных не используется

msg\_wait db 0Dh, 0Ah, 'Press enter for result: $' ; для организации задержки, для ввода данных не используется

input\_str db 80, ?, 83 dup(?) ;сохраняем введенную пользователем строку

output\_str db 83 dup('$') ; модифициаронная строка

result db 0Dh, 0Ah, 'The resulting string: $' ; перед выводом строки

int\_1ch\_restored db 0 ; флаг восстановления обработчика 1ch (0 - не восстановлен, 1 - восстановлен)

int\_4bh\_change db 0 ; флаг изменения обработчика 4bh (0 - не восстановлен, 1 - восстановлен)

timer\_counter dw 0 ; счетчик тиков таймера

tick\_counter dw 0 ; счетчик тиков

save\_int\_1ch dd ? ; хранение оригинального обработчика 60h

save\_int\_4bh dd ? ; хранение оригинального обработчика 1ch

.code

main:

; инициализация сегмента данных

mov ax, @data

mov ds, ax

mov es, ax ; инициализация доп сегмента es

; тк некоторые инструкции (например, stosb) используют дополнительный сегмент es

; для указания адреса назначения

; так же инициализация es позволяет корректно записывать данные в буфер

; вывод приветствия

mov ah, 09h ; функция вывода строки

lea dx, msg\_greeting ; строка-приветствие, которую надо вывести

int 21h ; вызов прерывание дос, которое выполняет функцию из ah

mov ah, 09h ; функция вывода строки

lea dx, msg\_enter\_string ; строка, которую нужно вывести

int 21h ; вызов прерывания дос, которое выполняет прерывание из ah

; сохраняем 4bh

mov ax, 354bh ; функция дос для получения адреса обработчика прерывания

int 21h ; вызов прерывания дос

; теперь адрес изначального обработчика 4bh содержится в bx (смещение) и cx (сегмент)

mov word ptr save\_int\_4bh, bx ; сохраняем смещение прерывания

mov word ptr save\_int\_4bh+2, es ; сохраняем сегмент прерывания

; сохранение и замена 1ch

mov ax, 351ch ; узнаем адрес текущего обработчика прерывания 1ch (таймер)

int 21h ; вызываем команду из ax

; теперь адрес изначального обработчика содержится в bx (смещение) и cx (сегмент)

mov word ptr save\_int\_1ch, bx ; сохраняем смещение обработчика 1ch

mov word ptr save\_int\_1ch+2, es ; сохраняем сегмент обработчика 1ch

push ds ; сохраняем текущий ds в стеке

mov dx, offset new\_int\_1ch ; записываем в dx смещение нового обработчика 1ch

mov ax, seg new\_int\_1ch ; записываем в ax сегмент нового обработчика 1ch

mov ds, ax ; помещаем сегмент нового обработчика 1ch в сегмент данных

mov ax, 251ch; функция дос для установки нового обработчика 1ch (25 - вектор постановки адреса обработчика)

int 21h ; вызов дос для установки нового обработчика 1ch

pop ds ; достаем обратно дата сегмент

; ввод строки с ограничением длины

mov ah, 0Ah ; команда для ввода строки с ограничением длины

lea dx, input\_str ; загружаем адрес буфера в регистр dx

int 21h ; вызов прерывание дос, которое выполняет программу из ah

; ввод строки

mov ah, 09h ; функция дос для ввода строки

lea dx, msg\_wait ; загружает адрес буфера в регистр dx

int 21h ;вызывает прерывание дос, которое выполняет функцию, указанную в ah

; ввоод до тох пор, пока нажатая клавиша != Enter

wait\_proc:

mov ah, 01h ; функция для проверки нажатой клавиши

int 21h ; вызов дос

cmp al, 0Dh ; проверяем, была ли нажата клаваша Enter

jne wait\_proc ; если al != Enter, то переходим по метке wait\_proc

; вызов 4bh

int 4bh

; завершение программы

mov ax, 4C00h ; функция дос для завершения программы

int 21h ; Вызов прерывания DOS, который завершит программу

;новый обработчик 1ch

new\_int\_1ch proc far

; far - это дальняя процедура и она может быть вызвана из другого сегмента кода

; обработчики прерываний должны быть дальними процедурами

; сохраняем, поскольку далее данные регистры будут менятся

push ax ; сохраняем ax в стеке

push ds ; сохраняем ds в стеке

mov ax, @data

mov ds, ax ; опредляем дс как дата сегмент

inc timer\_counter ; timer\_counter + 1 количество вызовов 1ch (inc - инкремент - добавляет к числу 1)

inc tick\_counter ; tick\_counter + 1 - количество тиков с момента установки нового обработчика

cmp tick\_counter, 55 ; ~ 3 секунды (18,2 \* 3), тк прерывание в 1ch генерируется 18.2 раз в секунду

jb check\_restore ; переход по метке check\_restore, если меньше 3 секунду

; проверям, был ли изменен 4bh до этого

cmp int\_4bh\_change, 0 ; сравниваем флаг int\_4bh\_change с 0

jne continue ; если флаг != 0, то переходим по метке continue

; меняем 4bh, если он не был изменен раньше

push ds ; сохраняем дата сегмент в стеке

mov dx, offset new\_int\_4bh ; загружаем адрес нового обработчика 4bh в bx

mov ax, seg new\_int\_4bh ; загружаем сегмент нового обработчика 4bh

mov ds, ax ; загружаем сегмент нового обработчика 4bh в дата сегмент

mov ax, 254bh ; функция дос для установки нового обработчика 4bh

int 21h ; вызов дос

pop ds ; достаем из стека дата сегмент

mov int\_4bh\_change, 1 ; обновляем флаг int\_4bh\_change - теперь 4bh изменен

continue:

; проверяем, был ли восстановлен 1ch до этого

cmp int\_1ch\_restored, 0 ; сравниваем флаг int\_1ch\_restored с 0

jne check\_restore ; если флаг не равен нулю, то 1ch восстановлен и выполняем переход на метку check\_restore

;восстанавливаем 1ch

push ax ; сохраняем ax, тк дальше будем его изменять

mov ax, 251ch ; уставновим вектор прерывания 1ch в регистр ax

lds dx, save\_int\_1ch ; загрузим адрес исходного(сохраненного) обработчика прерывания

; 1ch (сохранен в переменной save\_int\_1ch) в регистры ds:dx

int 21h ; выполняем фукнцию 25h, устанавливая вектор прерывания 1сh на адрес, сохраненный в ds:dx

pop ax ; восстанавливаем ax

mov int\_1ch\_restored, 1 ; устанавливаем флаг int\_1ch\_restored - теперь 1ch восстановлен

check\_restore:

; с момента установки нового обработчика прошло <3 секунд

; ничего не восстанавливаем

; восстанавливаем регистры в соответвующем порядке ds и ax

pop ds

pop ax

iret ; (interrupt return) завершаем обработчик прерывания

;извлекает cs:ip из стека: загружает значение cs и ip из стека,

; восстанавливая адрес возвратав прерванную прог.

;извлекает флаги из стека: восстанавливает регистр флага из стека

;возвращает управление: процессов продолжает выполнение программы с адреса,

;указанного в восстановленных cs:ip

new\_int\_1ch endp

;новый обработчик 4bh - выполняет обработку строки(удаление цифр в строке),

;выводит ее и восстанавливает себя

new\_int\_4bh proc far

; сохраняем значения регистров ax, bx, cx, dx, si, di, ds, ed на стеке

;чтобы не повредить их значение при выполнении процедуры

push ax

push bx

push cx

push dx

mov ax, @data

mov ds, ax

mov es, ax ; устанавливаем доп сегмент es

; получаем длину строки

xor cx, cx ; обнуляем регистр cx

mov cl, input\_str+1 ; загружаем длину введеной строки из input\_str+1

;(там хранится фактическая длина строки)

; настраиваем указатели

lea si, input\_str+2 ; источник - входная строка

lea di, output\_str ; назначение - выходная строка

processing\_cycle:

or cx, cx ; устанавливаем флаги в соответствии с содержимым в cx

; zf = 1, если cx = 0

jz finish\_proc ; переход к метке, если zf = 1 (cx = 0, те все символы обработаны)

lodsb ; загружает байт из адреса ds:si в al и инкрементирует в si

; проверяем, является ли символ цифрой

cmp al, '0' ; сравниваем al (текущий символ) с кодом символа '0'

jb not\_digit; если символ меньше нуля, то это не цифра

cmp al, '9' ; сравниваем al (текущий символ) с кодом символа '9'

ja not\_digit; если символ больше девятки, то это тоже не цифра

;если это все же цифра, то пропускаем ее и переходим к следующему символу

dec cx ; декрементируем cx (счетчик символов)

jmp processing\_cycle ; переходим к метке processing\_cycle

not\_digit: ; если символ != цифре

stosb ; сохраняем байт из al по адресу es:di и инкрементируем di

dec cx ; декрементируем cx (счетчик символов)

jmp processing\_cycle ; снова переходим по метке processing\_cycle

finish\_proc: ; если строка закончилась

; добавляем завершающие символы

mov al, 0Dh ; загружаем в al символ возврата каретки

stosb ; сохраняем байт из al по адресу es:di и инкрементируем di

mov al, 0Ah ; загружаем в al символ перевода курсора

stosb ; сохраняем байт из al по адресу es:di и инкрементируем di

mov al, '$' ; загружаем в al символ конца строки

stosb ; сохраняем байт из al по адресу es:di и инкрементируем di

; выводим результат

mov ah, 09h; загружаем функцию вывода строки в регистр ah

lea dx, result ; загружаем адрес строки result в регистр dx

int 21h ; вызов дос

mov ah, 09h ; загружаем фукнцию вывода строки в регистр ah

lea dx, output\_str ; загружаем адрес строки output\_str в регистр dx

int 21h ; вызов дос

; восстанавливаем старый обработчик 4bh

push ax ; сохраняем регистр ax

mov ax, 254bh ; уставновим вектор прерывания 4bh в регистр ax

lds dx, save\_int\_4bh ; ; загрузим адрес исходного(сохраненного) обработчика прерывания

; 4bh (сохранен в переменной save\_int\_4bh) в регистры ds:dx

int 21h ; выполняем фукнцию 25h, устанавливая вектор прерывания 4bh на адрес, сохраненный в ds:dx

pop ax ; восстанавливаем регистр ax

; восстанавливаем регистры в обратном порядке их сохранения

pop dx

pop cx

pop bx

pop ax

iret ; выходим из обработчика прерывания

new\_int\_4bh endp

end main