**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. 3344 |  | Тукалкин В.А. |
| 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 20.

Шифр 4т.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № сц. | Действия основной программы | Действия 1сh | Действия 9h | Действия 4bh |
| 4 | 2) замена 9h;  3) ожидание ввода строки. |  | 1) вып. работу;  2) восст. себя. |  |

т) Для каждого встретившегося символа “V” вывести максимальное число символов после этого “V”, образующих последовательность, являющуюся префиксом входной строки. Например: "aVooVaVoGGVVak"->"0 3 0 0 1".

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**

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:

–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).

**Выполнение 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 (SUBR\_INT) выполняет следующие действия:

1) При каждом вызове обработчик 9h выполняет обработку строки.

2) Восстанавливает на изначальный обработчик.

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.

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

Процедура process\_string преобразует строку по следующему принципу:

1) Проходит по каждому символу строки.

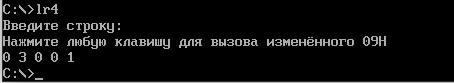
2) Если символ — “V”, начинает проверку символов, идущих далее.

3) Если символ — не символ “$” и равен символу с из префикса, то счётчик увеличивается и переходит к следующей итерации.

4) После проверки результат записывается в выходной буфер и строковом представлении.

**Тестирование сценария работы с прерываниями.**

После ввода строки, нажатия любой клавиши строка будет преобразована и выведена на экран. При нажатии клавиши enter программа завершится, восстановив все прерывания.



Тестирование преобразования строки. Результаты тестирования представлены в Таблице 1.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Ожидаемые результаты | Полученный результат |
| 1 | aVooVaVoGGVVak | 0 3 0 0 1 | 0 3 0 0 1 |
| 2 | VVVV | 3 2 1 0 | 3 2 1 0 |
| 3 | aVVVa | 0 0 1 | 0 0 1 |
| 4 | aasfsVVasdVssfVasdV | 0 1 0 1 0 | 0 1 0 1 0 |

## Выводы

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

# Приложение

Название файла: lr4.asm

.model small

.stack 400h

.data

KEEP\_IP DW 0 ; Для хранения смещения

KEEP\_CS DW 0 ; Для хранения сегмента

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

welcome\_str db 'Введите строку:', 0ah, 0dh, EOFLine ; Привествующая строка

change db "Нажмите любую клавишу для вызова изменённого 09H", 0ah, 0dh, EOFLine ; Сообщения о замене 9h

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

.code

mov ax,@data ; Загрузка сегментного

mov ds,ax ; регистра данных

; Вывод привествия

mov dx, OFFSET welcome\_str

mov ah, 09h ; Функция DOS для вывода строки

int 21h

; 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, es ; Запоминание сегмента оригинального обработчика

mov KEEP\_IP, bx ; Запоминание смещения оригинального обработчика

; 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 SUBR\_INT ; Смещение для процедуры в DX новый обработчик SUBR\_INT

mov ax, SEG SUBR\_INT ; Сегмент процедуры

mov ds, ax ; Помещаем в DS

mov ah, 25h ; Функция установки вектора

mov al, 09h ; Номер вектора

int 21h ; Меняем прерывание

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 (код 0x0D)

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

; Восстановление 9h

push ds

mov dx, KEEP\_IP ; Смещение для оригинального обработчика 9h

mov ax, KEEP\_CS ; 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

; Завершение программы

mov ah, 4ch

int 21h

; Процедура считывания строки

StringRead PROC NEAR

; Запоминание изменяемых 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

; Процедура вывода сообщения

writeMsg PROC NEAR

push ax ; Запоминание изменяемого регистра

mov ah, 09h ; Функция DOS для вывода строки

int 21h ; Вызов DOS

pop ax ; Выгрузка изменяемого регистра

ret

writeMsg ENDP

; Процедура обработки строки

process\_string PROC NEAR

; Запоминание изменяемых регистров

push si

push di

push bx

push cx

push dx

push ax

push es

; Загрузка сегмента данных в регистр es

mov ax, @data

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

cmp cl, 0 ; если строка пустая, то закончить

je leave

; Инициализация переменных

xor dx, dx ; bx будет использоваться для подсчета символов после "V"

xor bx, bx

process\_loop:

lodsb ; Загрузить символ из si в al

cmp al, 'V' ; Проверка, является ли символ "V"

jne next\_char ; Если нет, перейти к следующему символу

; Нашли "V", теперь нужно найти максимальное число символов после него, образующих префикс

push si ; Сохранить текущую позицию в строке

mov dx, 0 ; Сбросить счетчик символов после "V"

mov bx, 0 ; Сбросить смещение

mov al, [si] ; Загрузить символ из si в al

while:

cmp al, '$' ; Проверка на конец строки

je append

cmp al, input[bx] ; Проверка на совпадение символов

jne append

inc bx ; Увеличения смещения

inc dx ; Увеличение счётчика

mov al,[si+bx] ; Загрузить символ из si в al

jmp while

append:

pop si ; Восстановить текущую позицию в строке

mov al, dl ; Загрузить число символов в al

cmp al,9 ; Если число двухзначное

jle less\_than\_10

; Преобразование числа больше 9 в строку

mov bx, 10 ; Делитель

xor dx, dx ; Очистка dx для деления

mov ah, 0 ; Очистка ah для верного результата деления

div bx ; Делим ax на 10, частное в ax, остаток в dx

add al, '0' ; Преобразовать частное в символ

mov [di], al ; Сохранить символ в output

inc di ; Перейти к следующей позиции в строке

add dl, '0' ; Преобразовать остаток в символ

mov [di], dl ; Сохранить символ в output

inc di ; Перейти к следующей позиции в строке

mov al, ' ' ; Добавить пробел после числа

stosb ; Сохранить пробел в output

jmp next\_char ; Перейти к следующему символу

less\_than\_10:

add al, '0' ; Преобразовать число в символ

stosb ; Сохранить символ в output

mov al, ' ' ; Добавить пробел после числа

stosb ; Сохранить пробел в output

jmp next\_char ; Перейти к следующему символу

next\_char:

loop process\_loop ; Продолжить обработку следующего символа пока cx>0

leave:

; Завершение строки output

dec di ; Убрать последний пробел

mov al, '$' ; Добавить символ конца строки

stosb ; Сохранить символ конца строки в output

; Вывод преобразованной строки

mov dx, offset output

call writeMsg

; Возвращение сохраненных регистров

pop es

pop ax

pop dx

pop cx

pop bx

pop di

pop si

ret

process\_string ENDP

; Обработчик прерывания 9h

SUBR\_INT proc

; Сохранение изменяемых регистров

push ax

push es

push bx

push ds

push dx

; Преобразование строки

call process\_string

; Восстановление 9h

push ds

mov dx, KEEP\_IP ; Смещение для оригинального обработчика 9h

mov ax, KEEP\_CS ; 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

; Возвращение сохраненных регистров

pop dx

pop ds

pop bx

pop es

pop ax

jmp dword ptr [KEEP\_IP] ; Переход к выполнению оригинального кода обработки прерывания 9h

SUBR\_INT endp

end