**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 |  | Лутфулин Д.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 13.

Шифр 13л.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № сц. | Действия основной программы | Действия 1сh | Действия 9h | Действия 4bh |
| 13 | 2) замена 9h;  3) ожидание ввода строки. | а) восст. 9h;  б) 63 запуска;  в) вып. работу;  г) восст. себя. | заменить 1ch |  |

л) Все цифры, введённые во входной строке, заменить на 0 и 1: 0 - для цифр 0, 2, 4, 6, 8; 1 - для 1, 3, 5, 7, 9.

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.

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.

В конце программы восстанавливается старый вектор прерывания.

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 используются следующие команды обработки строк:

— MOVSB — переслать один байт из одной области памяти в другую;

— STOSB — записать содержимое регистра AL в память;

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

Используемые прерывания:

* DOS Прерывания (INT 21h):
  + 09h: Вывод строки.
  + 0Ah: Ввод строки с предварительным заголовком.
  + 02h: Вывод символа.
  + 25h: Установка нового обработчика прерывания.
  + 35h: Получение текущего адреса обработчика прерывания.
  + 4Ch: Завершение программы.
* BIOS Прерывания:
  + 16h (Клавиатура):
  + 00h: Проверка ввода с клавиатуры.
  + 15h (Таймер):
  + 86h: Задержка на основе таймера.
* Прерывание таймера (INT 1Ch): Таймер вызывается периодически (обычно 18.2 раза в секунду). Обработчик New1chHandler изменяет его поведение.
* Вызывается при каждом нажатии клавиши. Обработчик New9hHandler выполняет пользовательскую логику.

Используемые процедуры:

* StringRead:
  + Ввод строки с клавиатуры через функцию DOS 0Ah. Сохраняет длину строки в STR1HEAD и вводит символы в STR1. Добавляет в конец строки CR, LF, и $.
* ProcessInput:
  + Преобразует строку из STR1 в OUTSTR с помощью TransformString и выводит результат через функцию DOS 09h.
* TransformString:
  + Проходит по символам строки STR1 и копирует их в OUTSTR. Если символ - цифра, заменяет четные цифры на 0, нечетные - на 1. Все остальные символы копируются без изменений.
* Delay:
  + Создает паузу с помощью функции BIOS 15h, которая обеспечивает задержку на основе таймера.
* WriteEnter:
  + Выводит символы CR и LF (перенос строки) через функцию DOS 02h.
* Set\_new\_9h и Set\_new\_1ch:
  + Устанавливают новые обработчики для прерываний 9h (клавиатура) и 1Ch (таймер) через функцию DOS 25h.
* New9hHandler: Новый обработчик прерывания клавиатуры. Устанавливает обработчик таймера 1Ch при первом вызове.
* New1chHandler: Новый обработчик таймера. Восстанавливает 9h, считает количество вызовов и при достижении 63 вызовов обрабатывает введенную строку через ProcessInput. Затем восстанавливает стандартные обработчики.
* Restore9h и Restore1ch:
  + Восстанавливают оригинальные обработчики прерываний 9h и 1Ch.

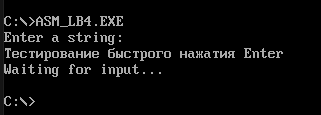
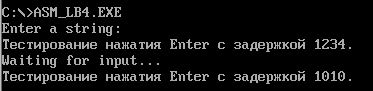
Основная программа:

Сначала инициализируется сегмент данных, стек. Выводится сообщение «Enter a string:», затем запрашивается ввод строки. Сохраняются в памяти оригинальные векторы 1ch и 9h. Выполняется замена 9h. Вызывается процедура wait for enter – ожидание ввода клавиши enter. После ввода восстанавливаются оригинальные вектора прерываний и выполняется выход из программы.

## Тестирование

1. Тестирование сценария работы с прерываниями.

После ввода строки, нажатия любой клавиши и ожидания 63-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 досрочно нажать клавишу enter, то программа не выведет строку на экран и завершится, восстановив прерывания.

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

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Ожидаемые результаты | Полученный результат |
| 1 | «Тестовый ввод без цифр.» | «Тестовый ввод без цифр.» | «Тестовый ввод без цифр.» |
| 2 | «Testing digits 0123456789» | «Testing digits 0101010101» | «Testing digits 0101010101» |
| 3 | «» | «» | «» |
| 4 | «checking max string length 12314421085798374987345983722435982739572349572395732» | «checking max string lenght 10110001001110110101101101100011100111110101110111110» | «checking max string lenght 10110001001110110101101101100011100111110101110111110» |

## Выводы

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

# Приложение

Название файла: lr4.asm

DATA\_SEG  SEGMENT

    INPUT\_MESSAGE  DB 'Enter a string:', 0Dh, 0Ah, '$'         ; Заголовок строки

    INT\_MESSAGE    DB 'Waiting for input...', 0Dh, 0Ah, '$'

    STR1HEAD       DB  79, 0

    STR1           DB  83 DUP('\*'), '$'

    OUTSTR         DB  83 DUP('\*')        ; Выходная строка для преобразования

    ; Сохранение старого вектора прерывания

    KEEP\_IP\_9H     DW  0

    KEEP\_CS\_9H     DW  0

    KEEP\_IP\_1CH    DW  0

    KEEP\_CS\_1CH    DW  0

    CallCount      DW  0

DATA\_SEG  ENDS

CODE\_SEG  SEGMENT

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

START:

    ; Инициализация сегмента данных

    MOV  AX, DATA\_SEG

    MOV  DS, AX

    MOV  ES, AX

    MOV  AX, AStack

    MOV  SS, AX

    ; Приветственное сообщение

    MOV  AH, 09h                  ;Функция DOS 09H: Выдать строку. Выдает строку находящуюся в DS:DX

    MOV  DX, OFFSET INPUT\_MESSAGE

    INT  21h

    CALL StringRead ; Ввод строки в STR1

    ; Cохраняем оригинальный вектор 1CH

    MOV  AH, 35H ; функция получения вектора: записывает в ES:BX адрес обработчика прерывания написанный в AH

    MOV  AL, 1CH ; номер вектора

    INT  21H

    MOV  KEEP\_IP\_1CH, BX ; запоминание смещения

    MOV  KEEP\_CS\_1CH, ES ; и сегмента

    ; Cохраняем оригинальный вектор 9H

    MOV  AL, 9H ; номер вектора

    INT  21H

    MOV  KEEP\_IP\_9H, BX ; запоминание смещения

    MOV  KEEP\_CS\_9H, ES ; и сегмента

    MOV  AX,  DS        ; возвращаем изначальное значение в ES

    MOV  ES,  AX        ;

    CALL Delay          ; Задержка перед заменой 9h

    CALL Set\_new\_9h     ; ставим новый в

    ; Ожидание ввода строки

    MOV  AH, 09h               ;Функция DOS 09H: Выдать строку. Выдает строку находящуюся в DS:DX

    MOV  DX, OFFSET INT\_MESSAGE

    INT  21h

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

    ; возврат стандартных векторов прерывания

    CALL Restore9h

    CALL Restore1ch

    ; Завершение программы

    MOV  AH, 4Ch

    INT  21h

ProcessInput PROC

    ; Преобразование строки

    CALL TransformString

    MOV  AH, 09h

    MOV  DX, OFFSET OUTSTR

    INT  21h

    RET

ProcessInput ENDP

StringRead PROC

    MOV  DX, OFFSET STR1HEAD

    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

    MOV  BL,  DS:[BP+1]    ; теперь в bx количество введённых символов

    ADD  BX,  BP           ; теперь bx указывает на конечный введённый символ(не считая последнего CR)

    ADD  BX, 2               ; Также помещаем во 2 строку cr, перенос строки и $

    MOV  WORD PTR [BX+1],240AH       ;добавить в конец(после STR1HEAD, STR1 и cr) 0ah и '$'

    call WriteEnter

    RET

StringRead ENDP

; Функция задержки

Delay PROC

    MOV  CX, 0Eh

    MOV  DX, 0FFFFh

    MOV  AH, 86h

    INT  15h

    RET

Delay ENDP

WriteEnter PROC

    MOV AH, 02h          ; Функция DOS: Вывод символа

    MOV DL, 0Dh          ; Код символа "возврат каретки" (CR)

    INT 21h              ; Вызов DOS прерывания

    MOV AH, 02h          ; Функция DOS: Вывод символа

    MOV DL, 0Ah          ; Код символа "перевод строки" (LF)

    INT 21h              ; Вызов DOS прерывания

    RET                  ; Возврат из процедуры

WriteEnter ENDP

Set\_new\_9h PROC

    PUSH DS

    MOV  DX, OFFSET  New9hHandler

    MOV  AX, SEG     New9hHandler

    MOV  DS, AX

    MOV  AH, 25H

    MOV  AL, 9H

    INT  21h

    POP  DS

    RET

Set\_new\_9h ENDP

Set\_new\_1ch PROC

    PUSH DS

    PUSH DX

    PUSH AX

    MOV  DX, OFFSET  New1chHandler

    MOV  AX, SEG     New1chHandler

    MOV  DS, AX

    MOV  AH, 25H

    MOV  AL, 1CH

    INT  21h

    POP  AX

    POP  DX

    POP  DS

    RET

Set\_new\_1ch ENDP

; Новый обработчик для прерывания 9h

New9hHandler PROC

    PUSH AX

    CALL Set\_new\_1ch

    MOV AL, 20H

    OUT 20H, AL

    POP AX

    JMP DWORD PTR [KEEP\_IP\_9H]

New9hHandler ENDP

New1chHandler PROC

    ; Сохраняем состояние для дальнейшего восстановления

    PUSH  AX

    PUSH  BX

    PUSH  CX

    PUSH  DX

    MOV   AX, [CallCount]

    CMP   AX, 0

    JNE  restored\_9h

first\_int\_1ch:          ;первое вхождение в измененное прерывание 1ch. Восстанавливаем 09h

    CALL  Restore9h

restored\_9h:

    INC  WORD PTR [CallCount]  ; Увеличиваем количество вызовов на 1

    ; Проверка, если вызовов было 63, то выполнить работу

    MOV  AX, [CallCount]

    CMP  AX, 63

    JNE  end\_1ch\_handler           ; Если 63 или больше, выходим из обработчика

    CALL ProcessInput

    CALL Restore1ch

end\_1ch\_handler:

    MOV AL, 20H

    OUT 20H, AL

    POP  DX

    POP  CX

    POP  BX

    POP  AX

    IRET

New1chHandler ENDP

; Преобразование строки: Замена цифр на 0 и 1

TransformString PROC

    CLD

    LEA SI, STR1            ; Адрес ввода

    LEA DI, OUTSTR          ; Адрес выхода

    XOR CH, CH

    MOV CL, [STR1HEAD+1]    ; Помещаем в счётчик число - длина введенной строки

    add cl, 3               ; Также помещаем во 2 строку cr, перенос строки и $

    ; CMP CL, 3               ; Проверяем была ли пустой

    ; JNE  TransformLoop

    ; RET

TransformLoop:

    LODSB                   ; Загружаем байт в AL

    CMP AL,  '0'            ; Проверяем, является ли символ цифрой

    JB StoreChar

    CMP AL,  '9'

    JA StoreChar

    ; Преобразуем цифры

    SUB AL,  '0'            ; Преобразуем символ в число

    TEST AL,  1             ; Проверяем младший бит (четность)

    JZ   EvenDigit          ; Если четное (младший бит 0), идем на EvenDigit

    MOV  AL, '1'            ; Если нечетное, ставим '1'

    JMP  StoreChar

EvenDigit:

    MOV  AL, '0'            ; Чётная цифра

StoreChar:

    STOSB                   ; Сохраняем в выходной строке

    LOOP TransformLoop      ; Переходим к следующему символу

    ; MOV  AL, DH

    ; STOSB

    ; MOV  AL, 0AH

    ; STOSB

    RET

TransformString ENDP

Restore9h PROC

    PUSH DS

    MOV  DX, KEEP\_IP\_9H

    MOV  AX, KEEP\_CS\_9H

    MOV  DS, AX

    MOV  AH, 25H

    MOV  AL, 9H

    INT  21H

    POP  DS

    RET

Restore9h ENDP

Restore1ch PROC

    PUSH DS

    MOV  DX, KEEP\_IP\_1CH

    MOV  AX, KEEP\_CS\_1CH

    MOV  DS, AX

    MOV  AH, 25H

    MOV  AL, 1CH

    INT  21H

    POP  DS

    RET

Restore1ch ENDP

CODE\_SEG ENDS

END START