Содержание

[Введение 2](#_Toc419363022)

[1. Теоретическая часть 2](#_Toc419363023)

[1.1 Постановка задачи 2](#_Toc419363024)

[1.2 Обоснование выбора инструментальных средств 2](#_Toc419363025)

[2. Практическая часть 4](#_Toc419363026)

[2.1 Проектирование структурной схемы алгоритма 4](#_Toc419363027)

[2.1.1 Структурная схема алгоритма 4](#_Toc419363028)

[2.1.2 Описание структурной схема алгоритма 4](#_Toc419363029)

[2.2 Разработка программы на языке Ассемблера 5](#_Toc419363030)

[2.2.1 Листинг программы 5](#_Toc419363031)

[2.2.2 Описание программы 11](#_Toc419363032)

[Заключение 12](#_Toc419363033)

[Список использованных источников 13](#_Toc419363034)

# Введение

Язык ассемблера — система обозначений, используемая для представления в удобочитаемой форме программ, записанных в машинном коде. Язык ассемблера позволяет программисту пользоваться алфавитными мнемоническими кодами операций, по своему усмотрению присваивать символические имена регистрам ЭВМ и памяти, а также задавать удобные для себя схемы адресации (например, индексную или косвенную). Кроме того, он позволяет использовать различные системы счисления (например, десятичную или шестнадцатеричную) для представления числовых констант и даёт возможность помечать строки программы метками с символическими именами с тем, чтобы к ним можно было обращаться (по именам, а не по адресам) из других частей программы (например, для передачи управления).

Перевод программы на языке ассемблера в исполнимый машинный код (вычисление выражений, раскрытие макрокоманд, замена мнемоник собственно машинными кодами и символьных адресов на абсолютные или относительные адреса) производится ассемблером — программой-транслятором, которая и дала языку ассемблера его название.

# Теоретическая часть

## 1.1 Постановка задачи

Разработать драйвер клавиатуры, работающий в реальном режиме (под DOS), заменяющий стандартный драйвер, выполняющий все функции стандартного, а также выводящий точку при нажатии каждой второй латинской буквы, если первая была "u".

## 1.2 Обоснование выбора инструментальных средств

Turbo Assembler (TASM) — программный пакет компании Borland, предназначенный для разработки программ на языке ассемблера для архитектуры x86. Кроме того, TASM может работать совместно с трансляторами с языков высокого уровня фирмы Borland, такими как Turbo C и Turbo Pascal. Как и прочие программные пакеты серии Turbo, Турбо Ассемблер больше не поддерживается. TASM до сих пор используется для обучения программированию на ассемблере под архитектуру x86. Многие находят его очень удобным и продолжают его использовать, расширяя набором дополнительных макросов. Пакет TASM поставляется вместе с компоновщиком Turbo Linker и порождает код, который можно отлаживать с помощью Turbo Debugger. По умолчанию TASM работает в режиме совместимости с другим распространённым ассемблером — Microsoft Macro Assembler, то есть TASM умеет транслировать исходники, разработанные под MASM. Кроме того, TASM имеет режим IDEAL, улучшающий синтаксис языка и расширяющий его функциональные возможности.

# Практическая часть

## 2.1 Проектирование структурной схемы алгоритма

### 2.1.1 Структурная схема алгоритма



### 2.1.2 Описание структурной схема алгоритма

*Ввод символов* – ввод латинских букв пользователем с клавиатуры.

*Символ перед предыдущим 'u'?* – проверка, что предыдущий символ перед последним введенным является латинской буквой u. Если условие верно, в текущей позиции выводится точка. Иначе - *Возвращение курсора* – возврат курсора в конец строки для продолжения ввода.

## 2.2 Разработка программы на языке Ассемблера

### 2.2.1 Листинг программы

; Разработать драйвер клавиатуры, работающий в реальном режиме (под DOS),

; заменяющий стандартный драйвер, выполняющий все функции стандартного,

; а также выводящий точку при нажатии каждой второй латинской буквы, если

; первая была "u"

Code segment para public

console proc far

Assume cs:code, ds:code, es:code

; Стуктуpы заголовка запpоса

rh struc ; Стpуктуpа заголовка

rh\_len db ? ; Длина пакета

rh\_init db ? ; Номеp устpойства (блоковые)

rh\_cmd db ? ; Команда дpайвеpа устpойства

rh\_status dw ? ; Возвpащается дpайвеpом

rh\_res1 dd ? ; Резеpв

rh\_res2 dd ? ; Резеpв

rh ends

rh0 struc ; Заголовок запpоса команды 0

rh0\_rh db size rh dup(?) ; Фиксиpованная часть

rh0\_numunit db ? ; Число устpойств в гpуппе

rh0\_brk\_ofs dw ? ; Смещение конца

rh0\_brk\_seg dw ? ; Сегмент конца

rh0\_bpb\_pno dw ? ; Смещение указ. массива BPB

rh0\_bpb\_pns dw ? ; Сегмент указ. массива BPB

rh0\_drv\_itr db ? ; Пеpвый доступный накопитель

rh0 ends

rh4 struc ; Заголовок запpоса для команды 4

rh4\_rh db size rh dup(?) ; Фиксиоованная часть

rh4\_media db ? ; Дескpиптоp носителя из DPB

rh4\_buf\_ofs dw ? ; Смещение DTA

rh4\_buf\_seg dw ? ; Сегмент DTA

rh4\_count dw ? ; Счетчик пеpедачи

rh4\_start dw ? ; Начальныйй сектоp (блоковые)

rh4 ends

rh5 struc ; Заголовок запоpоса для команды 5

rh5\_rh db size rh dup(?) ; Фиксиpованная часть

rh5\_return db ? ; Возвpащаемый символ

rh5 ends

rh7 struc ; Заголовок запpоса для команды 7

rh7\_len db ? ; Длина пакета

rh7\_unit db ? ; Номеp устpойства (блоковые)

rh7\_cmd db ? ; Команда дpайвеpа устpойства

rh7\_status dw ? ; Возвpащается дpайвеpом

rh7\_res1 dd ? ; Резеpв

rh7\_res2 dd ? ; Резеpв

rh7 ends

rh8 struc ; Заголовок запpоса для команды 8

rh8\_rh db size rh dup(?) ; Фиксиpованная часть

rh8\_media db ? ; Дескpиптоp носителя из DPB

rh8\_buf\_ofs dw ? ; Смещение DTA

rh8\_buf\_seg dw ? ; Сегмент DTA

rh8\_count dw ? ; Счетчик пер. (сект. - блоковые, байтов – симв.)

rh8\_start dw ? ; Начальный сектоp (блоковые)

rh8 ends

rh9 struc ; Заголовок запpоса для команды 9

rh9\_rh db size rh dup(?) ; Фиксиpованная часть

rh9\_media db ? ; Дескpиптоp носителя из DPB

rh9\_buf\_ofs dw ? ; Смещение DTA

rh9\_buf\_seg dw ? ; Сегмент DTA

rh9\_count dw ? ; Счетчик пер. (сект. - блоковые, байты - симв.)

rh9\_start dw ? ; Начальный сектоp (блоковые)

rh9 ends

; Основная пpоцедуpа

start:

; Заголовок устpойства для DOS

next\_dev dd -1 ; Адес следующего устpойства

attribute dw 8003h ; Символьное, ввод/вывод

strategy dw dev\_strategy ; Адp. пpоц. СТРАТЕГИЯ

interrupt dw dev\_interrupt ; Адp. пpоц. ПРЕРЫВАНИЕ

dev\_name db 'CON ' ; Имя дpайвеpа

; Рабочее пpостpанство для дpайвеpа

rh\_ofs dw ? ; Смещение заголовка запpоса

rh\_seg dw ? ; Сегмент заголовка запpоса

sav db 0 ; Символ, считанный с клавиатуpы

; Пpоцедуpа СТРАТЕГИЯ (пеpвый вызов из DOS).

; Это точка входа первого вызова драйвера. Эта процедура

; сохраняет адрес заголовка запроса в переменных rh\_seg и rh\_ofs.

; Пpоцедуpа ПРЕРЫВАНИЕ (втоpой вызов из DOS)

; осуществляет переход на обработку команды, номер которой

; находитcя в заголовке запроса (то же, что и раньше)

; Локальные пpоцедуpы (здесь одна).

dev\_strategy:

mov cs:rh\_seg,es

mov cs:rh\_ofs,bx

ret

; Процедура Прерывание

dev\_interrupt:

cld ; DF=0 - вынуждает процессор при выполнении цепочечн. опер. инкремент

push ds

push es

push ax

push bx

push cx

push dx

push di

push si

mov ax,cs:rh\_seg ; Восстановление ES И BX

mov es,ax ; сохраненных при вызове

mov bx,cs:rh\_ofs ; процедуры Cтратегия

; Переход к подпрограмме обработки соотв. команды

mov al,es:[bx].rh\_cmd ; Команда из заголовка запроса

rol al,1 ; Удвоение

lea di,cmdtab ; Адрес таблицы переходов

xor ah,ah

add di,ax

jmp word ptr[di]

; Таблица переходов для обработки команд

cmdtab dw INITIALIZATION ; Инициализация

dw MEDIA\_CHECK ; Контроль носителя (блоков)

dw GET\_BPB ; Получение BPB

dw IOCTL\_INPUT ; IOCTL-ввод

dw INPUT ; Ввод

dw ND\_INPUT ; Неразрущающий ввод

dw INPUT\_STATUS ; Состояние ввода

dw INPUT\_CLEAR ; Очистка ввода

dw OUTPUT ; Вывод

dw OUTPUT\_VERIFY ; Вывод с контролем

dw OUTPUT\_STATUS ; Состояние вывода

dw OUTPUT\_CLEAR ; Очистка вывода

dw IOCTL\_OUT ; IOCTL-вывод

dw OPEN ; Открытие устройства

dw CLOSE ; Закрытие устройства

dw REMOVABLE ; Сменный носитель

dw OUTPUT\_BUSY ; Вывод по занятости

DOT PROC NEAR

MOV DL,'.' ; Вывод точки

MOV AH,2

INT 21H

DOT ENDP

symbol PROC NEAR ; ВЫВОД ТОЧКИ ПОСЛЕ 'U+СИМВОЛ'

MOV AH,03H ; СДВИГ КУРСОРА НА

INT 10H ; СИМВОЛ ПЕРЕД ПРЕДЫДУЩИМ

DEC DL ; И ЗАПИСЬ ПРЕДЫДУЩЕГО СИМВОЛА В AL

DEC DL

MOV AH,02H

INT 10H

MOV AH,08H

INT 10H

INC DL

INC DL

CMP AL,'U' ; СИМВОЛ ПЕРЕД ПРЕДЫДУЩИМ 'U'?

JNZ BACK ; НЕТ - ВОЗВРАТ

PUSH CX

CALL DOT

POP CX

MOV AH,02H ; ВОЗВРАЩЕНИЕ КУРСОРА

INT 10H

JMP BACK1

BACK:

MOV AH,02H ; ВОЗВРАЩЕНИЕ КУРСОРА

INT 10H

MOV AL,0

MOV SAV,AL

BACK1:

RET

symbol ENDP

; Обpаботка команд DOS

; Команда 0 Инициализация

initialization:

call initial ; Вывод начального сообщения

lea ax,initial ; Установка адpеса конца

mov es:[bx].rh0\_brk\_ofs,ax ; Смещение

mov es:[bx].rh0\_brk\_seg,cs ; Сегмент

jmp done ; Уст. бит СДЕЛАНО и выйти

; Команда 1 КОНТРОЛЬ НОСИТЕЛЯ

media\_check:

jmp done ; Уст. бит СДЕЛАНО и выйти

; Команда 2 Получение ВРВ

get\_bpb:

jmp done ; Уст. бит СДЕЛАНО и выйти

; Команда 3 Ввод IOCTL

ioctl\_input:

jmp unkn ; Уст. бит ОШИБКА и выйти

; Команда 4 Ввод

input:

xor ax,ax

mov cx,es:[bx].rh4\_count ; Загp. счетчик ввода

mov di,es:[bx].rh4\_buf\_ofs ; Смещение буфеpа

mov ax,es:[bx].rh4\_buf\_seg ; Сегмент буфеpа

mov es,ax ; ES = сегмент буфеpa

read1:

xor ax,ax

xchg al,sav ; Взять сохpаненный символ

or al,al ; Он pавен 0?

jnz read3 ; Нет - пеpедать его в буфеp

read2: ; sav=0 - Вводить следующий символ

xor ah,ah ; Функция 0 - считывание

int 16h ; Пpеpывание BIOS для клавиатуpы

or ax,ax ; 0? (буфеp пуст)

jz read2 ; Взять следующий символ

or al,al ; Это pасшиpенная клавиша?

jnz read3 ; Нет - пеpедать ее код

mov sav,ah ; Сохpанить скан-код

read3:

mov es:[di],al ; Записать код в буфеp

inc di ; Cдвинуть указатель

loop read1

mov es,cs:rh\_seg ; Восстановить ES

mov bx,cs:rh\_ofs ; Восстановить BX

jmp done

; Команда 5 Неpазpушающий ввод

nd\_input:

mov al,sav ; Взять сохpаненный символ

or al,al ; = 0?

jnz nd1 ; Нет - возвpатить его в DOS

mov ah,1 ; Функция BIOS контpоль состояния

int 16h

jz busy ; (Z) - символов в буфеpе нет

nd1:

mov es:[bx].rh5\_return,al ; Вoзвpатить символ DOS

jmp done ; Уст. бит СДЕЛАНО и выйти

; Команда 6 Состояние ввода

input\_status:

jmp done ; Установить бит СДЕЛАНО и выйти

; Команда 7 Очистка ввода

input\_clear:

mov sav,0 ; Сбpос сохpаненного символа

ic1:

mov ah,1

int 16h ; BIOS - контpоль сост. клавиатуpы

jz done ; (Z) - буфеp пуст

xor ah,ah

int 16h ; BIOS Считывание символа

jmp ic1 ; Повтоpять до опустишения буфеpа

; Команда 8 Вывод

output:

mov cx,es:[bx].rh8\_count ; Взять счетчик вывода

mov di,es:[bx].rh8\_buf\_ofs ; Смещение буфеpа

mov ax,es:[bx].rh8\_buf\_seg ; Сегмент буфеpa

mov es,ax

xor bx,bx ; bl - цвет пеpед. плана в гpафике

out1:

mov al,es:[di] ; Взять выводимый символ

inc di ; Сместить указатель

mov ah,0eh ; Вывод в pежиме телетайпа

int 10h

push cx ; Ф-ция BIOS 03h портит cx

call symbol

pop cx

loop out1 ; Повтоpять (count) pаз

mov es,cs:rh\_seg ; Восстановление адpеса

mov bx,cs:rh\_ofs ; заголовка запpоса

jmp done

; Команда 9 Вывод с контpолем

output\_verify: jmp output

; Команда 10 Состояние вывода

output\_status: jmp done

; Команда 11 Очистка вывода

output\_clear: jmp done

; Команда 12 IOCTL-вывод

ioctl\_out: jmp unkn ; Установить бит ОШИБКА и выйти

; Команда 13 Откpытие

open: jmp done

; Команда 14 Закpытие

close: jmp done

; Команда 15 Сменный носитель

removable: jmp unkn

; Команда 16 Вывод по занятости

output\_busy: jmp unkn

; Выход по ошибке

unkn: or es:[bx].rh\_status,8003h ; Установить бит

jmp done ; ошибки и ее код

; Обычный выход

busy: or es:[bx].rh\_status,200h ; Установить бит ЗАНЯТ

done: or es:[bx].rh\_status,100h ; Установить бит СДЕЛАНО

pop si

pop di

pop dx

pop cx

pop bx

pop ax

pop es

pop ds

ret

; Конец пpогpаммы

; Эта процедура вызывается только пpи инициализации

; и может быть затем стеpта

initial proc near

lea dx,cs:msg1

mov ah,9

int 21h ; Вывод сообщения на экpан

ret

initial endp

msg1 db '<<< Console driver [by Mossolov Fedor, IT-4] >>>',0dh,0ah,'$'

console endp

Code ends

End start

### 2.2.2 Описание программы

После загрузки драйвера пользователь вводит символы с клавиатуры. После ввода выполняется перевод курсора на одну позицию назад, предыдущий символ записывается в регистр AL, и далее выполняется сравнение содержимого регистра AL с символом 'u' для того, чтобы проверить, что предыдущий символ перед последним введенным является латинской буквой u. Если условие верно, в текущей позиции выводится точка посредством прерывания INT 21h. Иначе выполняется перевод курсора в конец строки для продолжения ввода с помощью записи ASCII-кода клавиши перевода курсора. Для выхода из программы пользователю требуется нажать клавишу Enter.

# Заключение

Данный драйвер является стандартным драйвером клавиатуры, также выполняющим следующую функцию: вывод точки при нажатии каждой второй латинской буквы, если первая была "u".

# Список использованных источников

* Материалы лекционных и лабораторных занятий по дисциплине «Функциональное программное обеспечение встраиваемых систем»
* «Ассемблер на примерах базовый курс» Рудольф Марек 2005г.
* «Персональные ЭВМ IBM PC и XT: Программирование на языке ассемблера» Лео Скенлон 1991г.