|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 1 |

**Название:**

Дизассемблирование INT 8h

**Дисциплина:** Операционные системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Н.А. Гарасев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н.Ю. Рязанова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

Оглавление

[Введение 3](#_Toc51234700)

[Листинг 4](#_Toc51234703)

[Схема – алгоритма 7](#_Toc51234724)

[Заключение: 10](#_Toc51234725)

# Введение

Цель лабораторной работы: знакомство со средством дизассемблирования – sourcer и с получением дизассемблерного кода ядра операционной системы Windows на примере обработчика прерывания Int 8h в virtual mode – специальном режиме защищенного режима (32-разрядный режим работы), который эмулирует реальный режим работы вычислительной системы на базе процессоров Intel.

## Задание:

Используя sourser (sr.exe) получить дизассемблированный код обработчика аппаратного прерывания от системного таймера Int 8h.

На основе полученного кода составить алгоритм работы обработчика Int 8h.

По данной лабораторной работе составляется отчет в письменном виде.

• Отчет должен содержать: полученный ассемблерный код с адресами команд и комментариями;

• Графический алгоритм работы обработчика прерывания Int 8h, структурированный и выполненный в соответствии с ГОСТ 19.701-90 ЕСПД – «Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения».

# **Листинг**

Листинг 1. прерывания INT 08h

020A:0746 E8 0070 ;\* call sub\_1 ; (07B9)

020A:0746 E8 70 00 db 0E8h, 70h, 00h

020A:0749 06 push es

020A:074A 1E push ds

020A:074B 50 push ax

020A:074C 52 push dx

020A:074D B8 0040 mov ax,40h

020A:0750 8E D8 mov ds,ax

020A:0752 33 C0 xor ax,ax ; Zero register

020A:0754 8E C0 mov es,ax

020A:0756 FF 06 006C inc word ptr ds:[6Ch] ; (0040:006C=0AB71h)

020A:075A 75 04 jnz loc\_1 ; Jump if not zero

020A:075C FF 06 006E inc word ptr ds:[6Eh] ; (0040:006E=0Dh)

020A:0760 loc\_1:

020A:0760 83 3E 006E 18 cmp word ptr ds:[6Eh],18h ; (0040:006E=0Dh)

020A:0765 75 15 jne loc\_2 ; Jump if not equal

020A:0767 81 3E 006C 00B0 cmp word ptr ds:[6Ch],0B0h ; (0040:006C=0AB71h)

020A:076D 75 0D jne loc\_2 ; Jump if not equal

020A:076F A3 006E mov word ptr ds:[6Eh],ax ; (0040:006E=0Dh)

020A:0772 A3 006C mov word ptr ds:[6Ch],ax ; (0040:006C=0AB71h)

020A:0775 C6 06 0070 01 mov byte ptr ds:[70h],1 ; (0040:0070=0)

020A:077A 0C 08 or al,8

020A:077C loc\_2:

020A:077C 50 push ax

020A:077D FE 0E 0040 dec byte ptr ds:[40h] ; (0040:0040=0FFh)

020A:0781 75 0B jnz loc\_3 ; Jump if not zero

020A:0783 80 26 003F F0 and byte ptr ds:[3Fh],0F0h ; (0040:003F=0)

020A:0788 B0 0C mov al,0Ch

020A:078A BA 03F2 mov dx,3F2h

020A:078D EE out dx,al ; port 3F2h, dsk0 contrl output

020A:078E loc\_3:

020A:078E 58 pop ax

020A:078F F7 06 0314 0004 test word ptr ds:[314h],4 ; (0040:0314=3200h)

020A:0795 75 0C jnz loc\_4 ; Jump if not zero

020A:0797 9F lahf ; Load ah from flags

020A:0798 86 E0 xchg ah,al

020A:079A 50 push ax

020A:079B 26: FF 1E 0070 call dword ptr es:[70h] ; (0000:0070=6ADh)

020A:07A0 EB 03 jmp short loc\_5 ; (07A5)

020A:07A2 90 nop

020A:07A3 loc\_4:

020A:07A3 CD 1C int 1Ch ; Timer break (call each 18.2ms)

020A:07A5 loc\_5:

020A:07A5 E8 0011 call sub\_1 ; (07B9)

020A:07A8 B0 20 mov al,20h ; ' '

020A:07AA E6 20 out 20h,al ; port 20h, 8259-1 int command

; al = 20h, end of interrupt

020A:07AC 5A pop dx

020A:07AD 58 pop ax

020A:07AE 1F pop ds

020A:07AF 07 pop es

020A:064B 5E ;\* pop si

020A:064B 5E db 5Eh

020A:064C loc\_1:

020A:064C 1E push ds

020A:064D 50 push ax

020A:064E B8 0040 mov ax,40h

020A:0651 8E D8 mov ds,ax

020A:0653 F7 06 0314 2400 test word ptr ds:[314h],2400h ; (0040:0314=3200h)

020A:0659 75 4F jnz loc\_9 ; Jump if not zero

020A:065B 55 push bp

020A:065C 8B EC mov bp,sp

020A:065E 8B 46 0A mov ax,[bp+0Ah]

020A:0661 5D pop bp

020A:0662 A9 0100 test ax,100h

020A:0665 75 43 jnz loc\_9 ; Jump if not zero

020A:0667 A9 0200 test ax,200h

020A:066A 74 22 jz loc\_5 ; Jump if zero

020A:066C F0> 81 0E 0314 0200 lock or word ptr ds:[314h],200h ; (0040:0314=3200h)

020A:0673 F7 06 0314 0003 test word ptr ds:[314h],3 ; (0040:0314=3200h)

020A:0679 75 2F jnz loc\_9 ; Jump if not zero

020A:067B loc\_2:

020A:067B 86 E0 xchg ah,al

020A:067D FC cld ; Clear direction

020A:067E A8 04 test al,4

020A:0680 75 25 jnz loc\_8 ; Jump if not zero

020A:0682 loc\_3:

020A:0682 A8 08 test al,8

020A:0684 75 11 jnz loc\_6 ; Jump if not zero

020A:0686 70 19 jo loc\_7 ; Jump if overflow=1

020A:0688 loc\_4:

020A:0688 9E sahf ; Store ah into flags

020A:0689 58 pop ax

020A:068A 1F pop ds

020A:068B CA 0002 retf 2

020A:068E loc\_5:

020A:068E F0> 81 26 0314 FDFF lock and word ptr ds:[314h],0FDFFh ; (020A:0314=3231h)

020A:0695 EB E4 jmp short loc\_2 ; (067B)

020A:0697 loc\_6:

020A:0697 70 EF jo loc\_4 ; Jump if overflow=1

020A:0699 50 push ax

020A:069A B0 7F mov al,7Fh

020A:069C 04 02 add al,2

020A:069E 58 pop ax

020A:069F EB E7 jmp short loc\_4 ; (0688)

020A:06A1 loc\_7:

020A:06A1 50 push ax

020A:06A2 32 C0 xor al,al ; Zero register

020A:06A4 58 pop ax

020A:06A5 EB E1 jmp short loc\_4 ; (0688)

020A:06A7 loc\_8:

020A:06A7 FD std ; Set direction flag

020A:06A8 EB D8 jmp short loc\_3 ; (0682)

020A:06AA loc\_9:

020A:06AA 58 pop ax

020A:06AB 1F pop ds

020A:06AC CF iret ; Interrupt return

;\* No entry point to code

Листинг 2. Sub\_1 SUBROUTINE

020A:07B9 1E push ds

020A:07BA 50 push ax

020A:07BB B8 0040 mov ax,40h

020A:07BE 8E D8 mov ds,ax

020A:07C0 9F lahf ; Load ah from flags

020A:07C1 F7 06 0314 2400 test word ptr ds:[314h],2400h ; (0040:0314=3200h)

020A:07C7 75 0C jnz loc\_7 ; Jump if not zero

020A:07C9 F0> 81 26 0314 FDFF lock and word ptr ds:[314h],0FDFFh ; (0040:0314=3200h)

020A:07D0 loc\_6:

020A:07D0 9E sahf ; Store ah into flags

020A:07D1 58 pop ax

020A:07D2 1F pop ds

020A:07D3 EB 03 jmp short loc\_8 ; (07D8)

020A:07D5 loc\_7:

020A:07D5 FA cli ; Disable interrupts

020A:07D6 EB F8 jmp short loc\_6 ; (07D0)

020A:07D8 loc\_8:

020A:07D8 C3 retn

sub\_1 endp

# Схема – алгоритма

На рис. 1 – 3 представлены схемы алгоритма int 8h и подпрограммы sub\_1

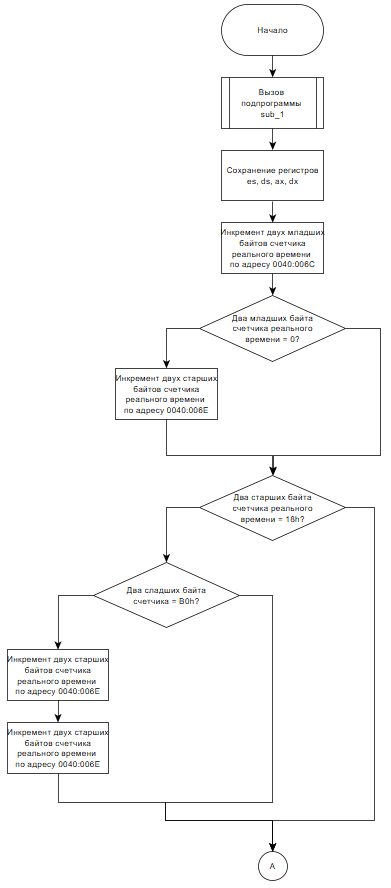


Рисунок 1. Int 8h

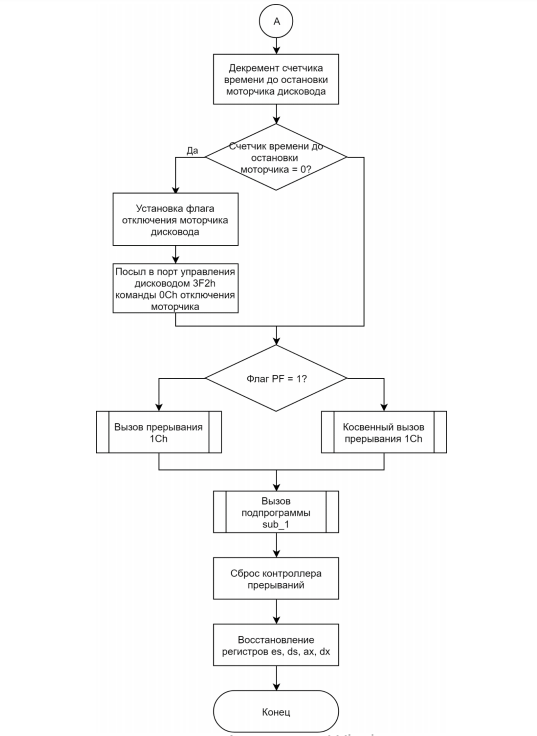


Рисунок 2. Int 8h

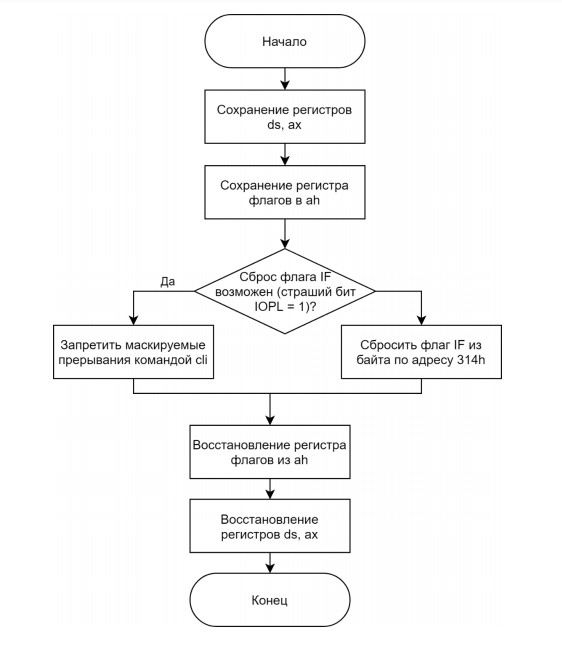


Рисунок 3. Подпрограмма sub\_1

**Функции обработчика прерывания int 8h:**

* Инкремент счетчика времени
* Вызов пользовательского прерывания 1Ch
* Декремент счетчика времени до отключения моторчика дисковода

# Заключение:

В ходе данной лабораторной работы были приобретены навыки получения адреса начала прерывания и листинг прерывания с помощью дизассемблера. Были изучены алгоритмы работы прерывания int 8h. Данное прерывание отвечает за изменение счетчика системного времени, управление контроллером дисковода, а также является способом вызова пользовательского прерывания.