*Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования*

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«Московский государственный технический университет  имени Н.Э. Баумана (национальный исследовательский институт)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**Отчёт**

**по лабораторной работе 1**

**Дисциплина: Операционные системы**

**Тема лабораторной работы работы:** Дизассемблирование INT 8h

Студентка гр. ИУ7-51б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сушина А.Д.**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рязанова Н.Ю.**

(Подпись, дата) (И.О. Фамилия)

Москва, 2019г

**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, который эмулирует реальный режим работы вычислительной системы на базе процессоров Intel.

**Задание:**

Используя sourser **( sr.exe**) получить дизассемблерный 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 коментариями;
* Графический алгоритм работы обработчика прерывания Int 8h, структурированный и выполненный в соответствии с ГОСТ19.701-90 ЕСПД – «Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения».

# Ход работы

**1. Получение дизассемблированного кода обработчика аппаратного прерывания от системного таймера int 8h.**

Для начала дизассемблирования необходимо определить начальный и конечный адреса кода. Получим начальный адрес из таблицы векторов прерываний по номеру прерывания. Вектор для 8ого прерывания имеет смещение 32( 8 \* 4 = 32 ). В шестнадцатеричном виде это 20h.

Используем отладчик Debug, чтобы получить необходимый адрес. Его вывод представлен на рисунке 1.

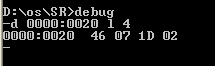
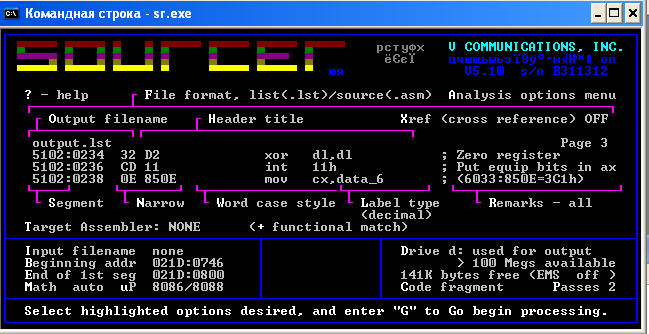


Рис 1. Использование отладчика Debug для получения адреса кода прерывания.

В результате получим адрес 021D : 0746.

Запустим программу sr.exe и получим необходимый файл. В поле begginnig adress введем полученный адрес, а в поле End of lst seg введем примерный конечный адрес кода. Затем выполним дизассемблирование.



Полученный файл (с комментариями) :

Sourcer v5.10 14-Sep-19 3:13 pm Page 1

021D:0746 E8 0070 ;\* call sub\_1 ; (07B9)

021D:0746 E8 70 00 db 0E8h, 70h, 00h

021D:0749 06 push es

021D:074A 1E push ds

021D:074B 50 push ax

021D:074C 52 push dx

021D:074D B8 0040 mov ax,40h

021D:0750 8E D8 mov ds,ax

021D:0752 33 C0 xor ax,ax ; Zero register

021D:0754 8E C0 mov es,ax

021D:0756 FF 06 006C inc word ptr ds:[6Ch] ; (0040:006C=3B3Bh)

021D:075A 75 04 jnz loc\_1 ; Jump if not zero

021D:075C FF 06 006E inc word ptr ds:[6Eh] ; (0040:006E=0Fh)

021D:0760 loc\_1:

021D:0760 83 3E 006E 18 cmp word ptr ds:[6Eh],18h ; (0040:006E=0Fh)

021D:0765 75 15 jne loc\_2 ; Jump if not equal

021D:0767 81 3E 006C 00B0 cmp word ptr ds:[6Ch],0B0h ; (0040:006C=3B3Bh)

021D:076D 75 0D jne loc\_2 ; Jump if not equal

021D:076F A3 006E mov word ptr ds:[6Eh],ax ; (0040:006E=0Fh)

021D:0772 A3 006C mov word ptr ds:[6Ch],ax ; (0040:006C=3B3Bh)

021D:0775 C6 06 0070 01 mov byte ptr ds:[70h],1 ; (0040:0070=0)

021D:077A 0C 08 or al,8

021D:077C loc\_2:

021D:077C 50 push ax

021D:077D FE 0E 0040 dec byte ptr ds:[40h] ; (0040:0040=36h)

021D:0781 75 0B jnz loc\_3 ; Jump if not zero

021D:0783 80 26 003F F0 and byte ptr ds:[3Fh],0F0h ; (0040:003F=0)

021D:0788 B0 0C mov al,0Ch

021D:078A BA 03F2 mov dx,3F2h

021D:078D EE out dx,al ; port 3F2h, dsk0 contrl output

021D:078E loc\_3:

021D:078E 58 pop ax

021D:078F F7 06 0314 0004 test word ptr ds:[314h],4 ; (0040:0314=3200h)

021D:0795 75 0C jnz loc\_4 ; Jump if not zero

021D:0797 9F lahf ; Load ah from flags

021D:0798 86 E0 xchg ah,al

021D:079A 50 push ax

021D:079B 26: FF 1E 0070 call dword ptr es:[70h] ; (0000:0070=6ADh)

021D:07A0 EB 03 jmp short loc\_5 ; (07A5)

021D:07A2 90 nop

021D:07A3 loc\_4:

021D:07A3 CD 1C int 1Ch ; Timer break (call each 18.2ms)

021D:07A5 loc\_5:

021D:07A5 E8 0011 call sub\_1 ; (07B9)

021D:07A8 B0 20 mov al,20h ; ' '

021D:07AA E6 20 out 20h,al ; port 20h, 8259-1 int command

; al = 20h, end of interrupt

021D:07AC 5A pop dx

021D:07AD 58 pop ax

021D:07AE 1F pop ds

021D:07AF 07 pop es

021D:07B0 E9 FE99 jmp $-164h

021D:07B3 C4 db 0C4h

;\* No entry point to code

021D:07B4 C4 0E 93E9 les cx,dword ptr ds:[93E9h] ; (0000:93E9=0) Load 32 bit ptr

021D:07B8 FE db 0FEh

Sourcer v5.10 14-Sep-19 3:13 pm Page 2

;ЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯ

; SUBROUTINE

;ЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬЬ

sub\_1 proc near

021D:07B9 1E push ds

021D:07BA 50 push ax

021D:07BB B8 0040 mov ax,40h

021D:07BE 8E D8 mov ds,ax

021D:07C0 9F lahf ; Load ah from flags

021D:07C1 F7 06 0314 2400 test word ptr ds:[314h],2400h

021D:07C7 75 0C jnz loc\_7 ; Jump if not zero

021D:07C9 F0> 81 26 0314 FDFF lock and word ptr ds:[314h],0FDFFh

021D:07D0 loc\_6:

021D:07D0 9E sahf ; Store ah into flags

021D:07D1 58 pop ax

021D:07D2 1F pop ds

021D:07D3 EB 03 jmp short loc\_8 ; (07D8)

021D:07D5 loc\_7:

021D:07D5 FA cli ; Disable interrupts

021D:07D6 EB F8 jmp short loc\_6 ; (07D0)

021D:07D8 loc\_8:

021D:07D8 C3 retn

sub\_1 endp