**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# ОТЧЁТ

**по лабораторной работе №4**

**по дисциплине «Операционные системы»**

**Тема: Обработка стандартных прерываний**

Студентка гр. 8381 _____ Звегинцева Е.Н.
Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

**Цель работы**

Исследовать механизм обработки прерываний и технику построения резидентных программ на примере программы резидентного обработчика прерывания таймера.

**Основные теоретические положения.**

Резидентные обработчики прерываний - это программные модули, которые вызываются при возникновении прерываний определенного типа (сигнал таймера, нажатие клавиши и т.д.), которым соответствуют определенные вектора прерывания. Когда вызывается прерывание, процессор переключается на выполнение кода обработчика, а затем возвращается на выполнение прерванной программы. Адрес возврата в прерванную программу (CS:IP) запоминается в стеке вместе с регистром флагов. Затем в CS:IP загружается адрес точки входа программы обработки прерывания и начинает выполняться его код. Обработчик прерывания должен заканчиваться инструкцией IRET (возврат из прерывания).

Вектор прерывания имеет длину 4 байта. В первом хранится значение IP, во втором - CS. Младшие 1024 байта памяти содержат 256 векторов. Вектор для прерывания 0 начинается с ячейки 0000:0000, для прерывания 1 - с ячейки 0000:0004 и т.д.

Обработчик прерывания - это отдельная процедура, имеющая следующую структуру:

```
ROUT PROC FAR
PUSH AX ; сохранение изменяемых регистров
        .....
<действия по обработке прерывания>
POP AX ; восстановление регистров
MOV AL, 20H
OUT 20H, AL
IRET
ROUT ENDP
```

Две последние строки необходимы для разрешения обработки

прерываний с более низкими уровнями, чем только что обработанное. Для установки написанного прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес.

```
PUSH DS
MOV DX, OFFSET ROUT       ; смещение для процедуры в
DX MOV AX, SEG ROUT       ; сегмент процедуры
MOV DS, AX                ; помещаем в DS
MOV AH, 25H               ; функция установки вектора
MOV AL, 1CH               ; номер вектора
INT 21H                   ; меняем прерывание
POP DS
```

Программа, выгружающая обработчик прерываний должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H позволяет восстановить значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. Программа должна содержать следующие инструкции:

```
; -- хранится в обработчике прерываний
KEEP_CS DW 0              ; для хранения
сегмента KEEP_IP DW 0     ; и смещения
прерывания
; -- в программе при загрузке обработчика
прерывания MOV AH, 35H    ; функция получения
вектора MOV AL, 1CH   ; номер вектора ШТЕ 21Р
MOV KEEP_IP, BX       ; запоминание смещения
MOV KEEP_CS, ES       ; и сегмента
; -- в программе при выгрузке обработчика прерываний CLI

PUSH DS
MOV             DX,
KEEP_IP      MOV
AX,       KEEP_CS
MOV DS, AX
MOV      AH,
25H     MOV
AL, 1CH
INT 21H      ; восстанавливаем вектор
POP DS
STI
```

Для того, чтобы оставить процедуру прерывания резидентной в памяти,

следует воспользоваться функцией DOS 31h прерывания 21h. Эта функция оставляет память, размер которой указывается в качестве параметра, занятой, а остальную память освобождает и осуществляет выход в DOS.

Функция 31h int 21h использует следующие параметры:

AH - номер функции 31h;
AL - код завершения программы;
DX - размер памяти в параграфах, требуемый резидентной программе.
Пример обращения к функции:

```
MOV DX, OFFSET LAST_BYTE ; размер в байтах от начала сегмента
MOV CL,4 ; перевод в параграфы
SHR DX,CL
INC DX       ; размер в параграфах
MOV AH,31h
INT 21h
```

**Выполнение работы**

Программа разделена на 4 сегмента – код и данные резидентной части (`RESIDENT`), стек (`STACK`), строковые данные (`STRINGS`) и код нерезидентной части (`TEXT`). Также после загрузки в память, программа будет содержать PSP.

Резидентная часть будет включать в себя сегмент `RESIDENT`, а также сохранит PSP процесса. Обработчик прерывания будет выполняться на отдельном стеке (он предоставляется DOS), поэтому стек, использующийся нерезидентной программой сохранять не нужно. В `RESIDENT` содержится:

1. «Сигнатура» – несколько байт в начале раздела, по которым можно с достаточной точностью определить, что установленные обработчик прерывания является данной программой. Сигнатура расположена в самом начале сегмента (т.е. `RESIDENT:0000`), поэтому для проверки нужно взять сегментный адрес обработчика прерывания и проверить первые байты сегмента с этим адресом.
2. Адрес (`CS:IP`) «старого» обработчика прерывания, т.е. обработчика, который мы заменяем при запуске программы.
3. Счётчик.
4. Функцию `WRD2DEC_RJUST` для записи 16-битного числа в строку. Эта

функция пишет ровно 5 знаков; число получается выровненным по правому краю.

5. Функция DISPLAY_STRING, которая отображает строку при помощи int 10h, AH=13h.
6. Функция ISR1C, служащая обработчиком прерывания 1Ch.

При запуске программы происходит проверка сигнатуры текущего обработчика прерывания 1Ch, а также проверка хвоста командной строки. При этом получается номер команды, которая затем вызывается по таблице. В ES:BX ко- манде передаётся адрес текущего обработчика прерывания, а в DS – адрес сегмента STRINGS (это удобно, т.к. 3 из 5 команд это обработчики ошибок, печатающие сообщения).

- Проверка сигнатуры происходит в функции CHECK_SIGNATURE. Она сравнивает то, что расположено по адресу сигнатуры в текущем обработчике, с настоящей сигнатурой в сегменте RESIDENT.
- Проверкой командной занимается функция CHECK_CMDLINE.
- Таблица команд – ACTIONS_TAB.

Установка обработчика прерывания происходит в функции INSTALL_ISR.

1. Она освобождает память, выделенную загрузчиком под среду процесса. Среда не понадобится резиденту.
2. Сохраняется старый обработчик. Его адрес записывается в память резидента, поэтому его можно будет вернуть при выгрузке резидента.
3. Устанавливается свой обработчик прерывания. В качестве адреса его сегмента кода берётся адрес сегмента RESIDENT, а в качестве его IP – адрес ISR1C относительно этого сегмента.
4. Вычисляется размер резидента в параграфах. Резидент состоит из PSP и сегмента RESIDENT, поэтому для этого из адреса сегмента, который следует

за ними (это сегмент стека), вычитается адрес сегмента PSP.

5. Выполняется выход с сохранением резидента (`int 21h`, `AH=31h`).

6. Если программа продолжает выполняться дальше, сообщается об ошибке.

Выгрузка резидента происходит в функции `UNINSTALL_ISR`:

1. Из памяти резидента извлекается адрес старого обработчика. Он восстанавливается.

2. Вычисляется адрес начала блока памяти, занимаемого резидентом, т.е. адрес его PSP.

3. Освобождается память резидента.

Для освобождения памяти в обоих случаях (и среда, и резидент) используется функция `FREE`. Она просто использует `int 21h`, `AH=49h` и обрабатывает ошибки. При выходе из `FREE` флаг `CF`, обозначающий ошибку, сохраняется; благодаря этому можно использовать инструкции `JC` и `JNC` для проверки ошибок, как и при использовании функции DOS напрямую.

Для установки обработчика прерывания написана функция `SET_ISR1C`, которая вызывает `int 21h`, `AH=25h` с `AL=1Ch`.

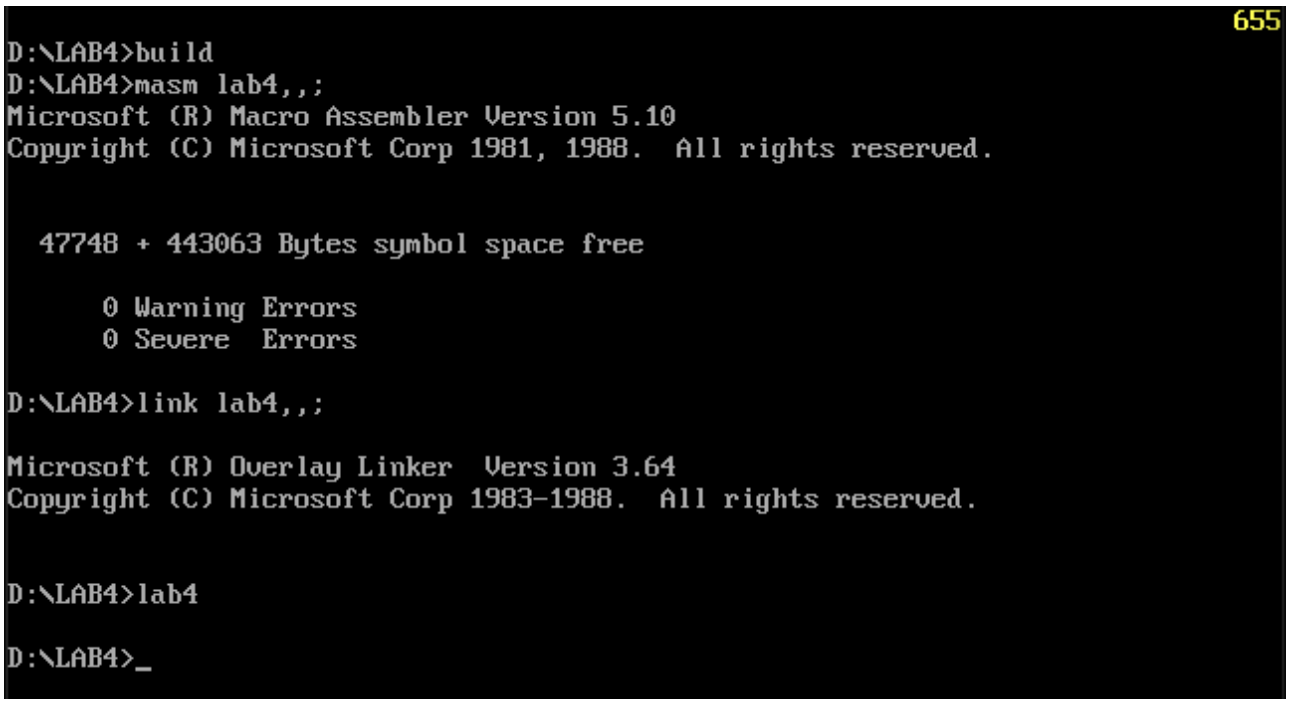


Рисунок 1- Запущенная программа

Обработчик прерывания выводит текст жёлтого цвета при помощи передачи значения `0Eh` функции `int 10h`, `AH=13h` в регистре `BX` (атрибут). Верхний октет атрибута обозначает цвет фона ($0$ → чёрный), а нижний – цвет текста ($E_{16}$ → жёлтый).

Текст выводится в колонке $4B_{16} = 75$. Таким образом, строка длиной 5 знаков оказывается на правом крае терминала шириной в 80 колонок.

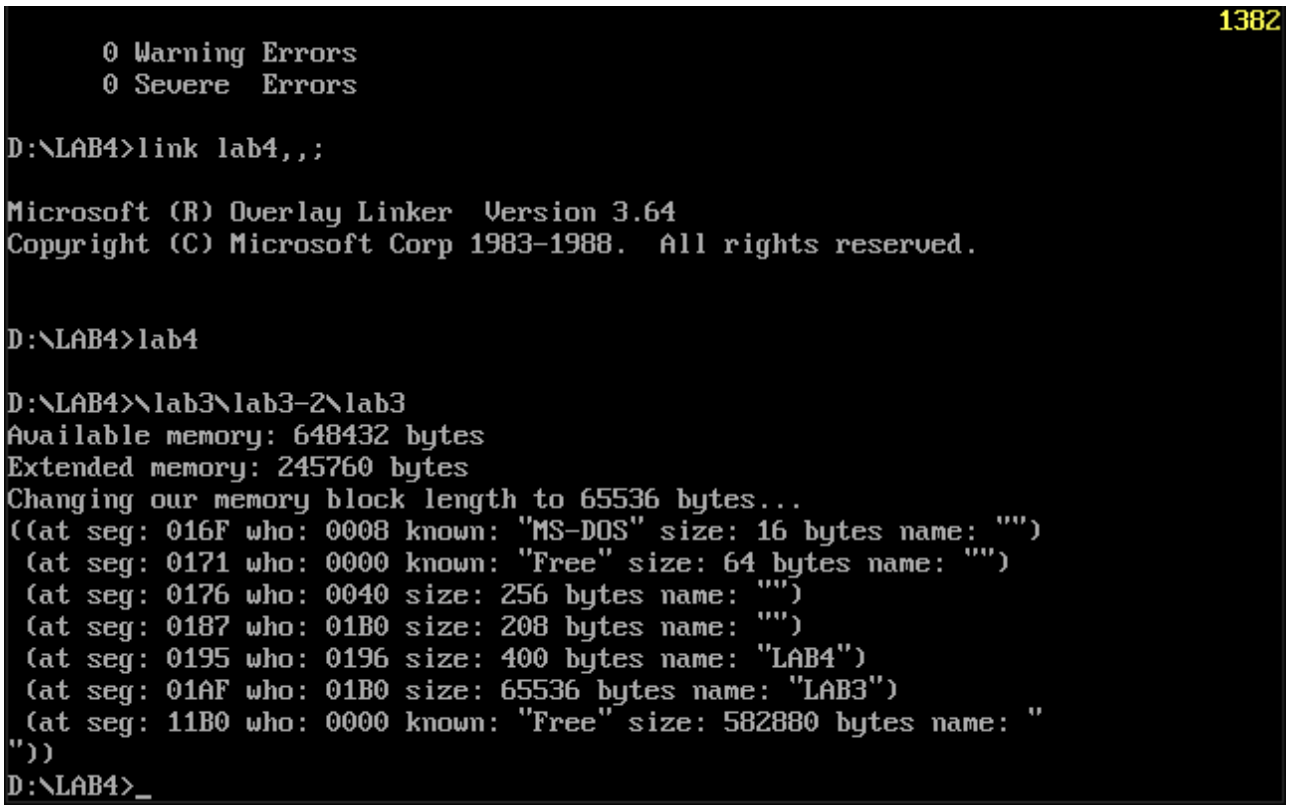


Рисунок 2- Запуск программы из ЛР №3 при загруженном резиденте

Здесь видно, что только один блок памяти принадлежит резиденту (адрес PSP: `0196h`). Перед ним расположен блок, содержащий среду программы LAB3 (`01B0h`). Его размер равен размеру блока среды LAB4, поэтому он попал на место, появившееся после освобождения памяти, которую занимала среда LAB4.

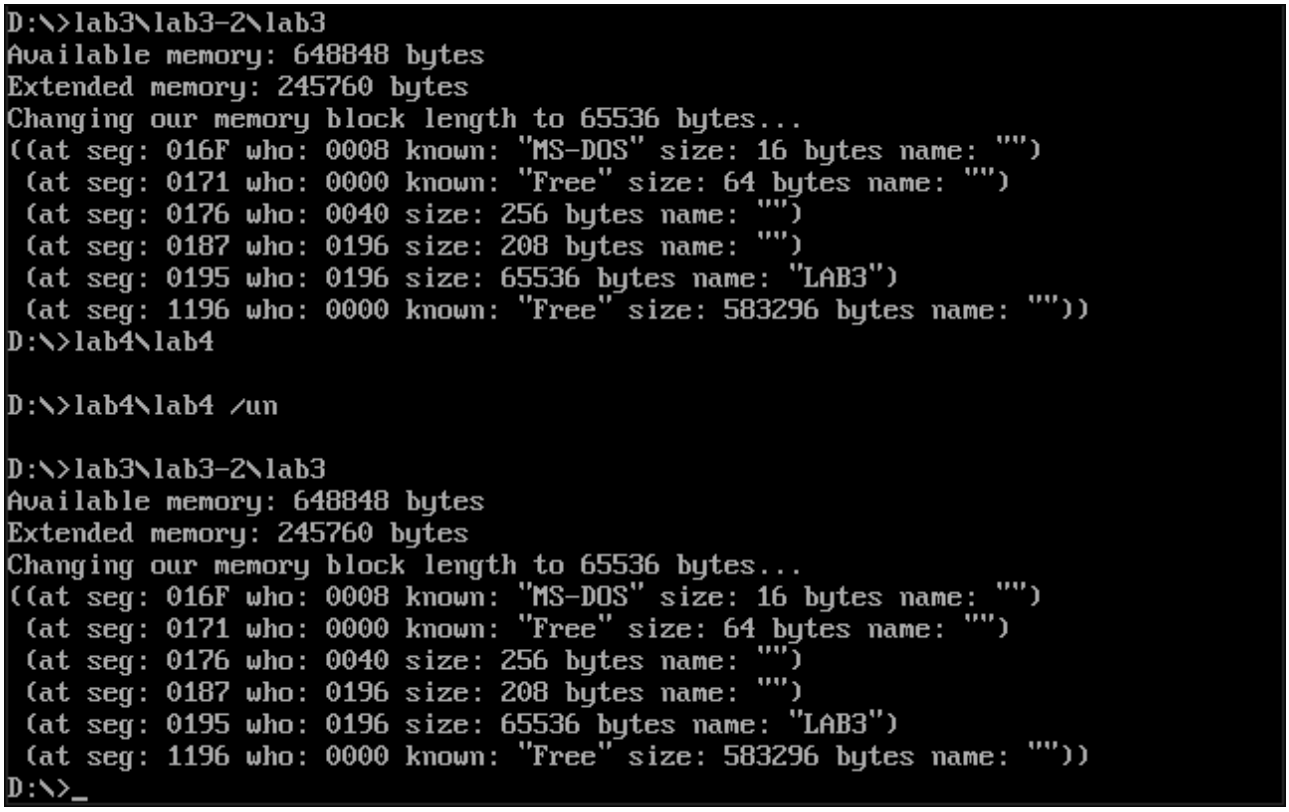


Рисунок 3- Запуск программы из ЛР №3 до загрузки и после выгрузки резидента

Здесь видно, что после выгрузки резидента вся занимаемая им память освобождается (для сравнения, вверху то, что было до его загрузки).

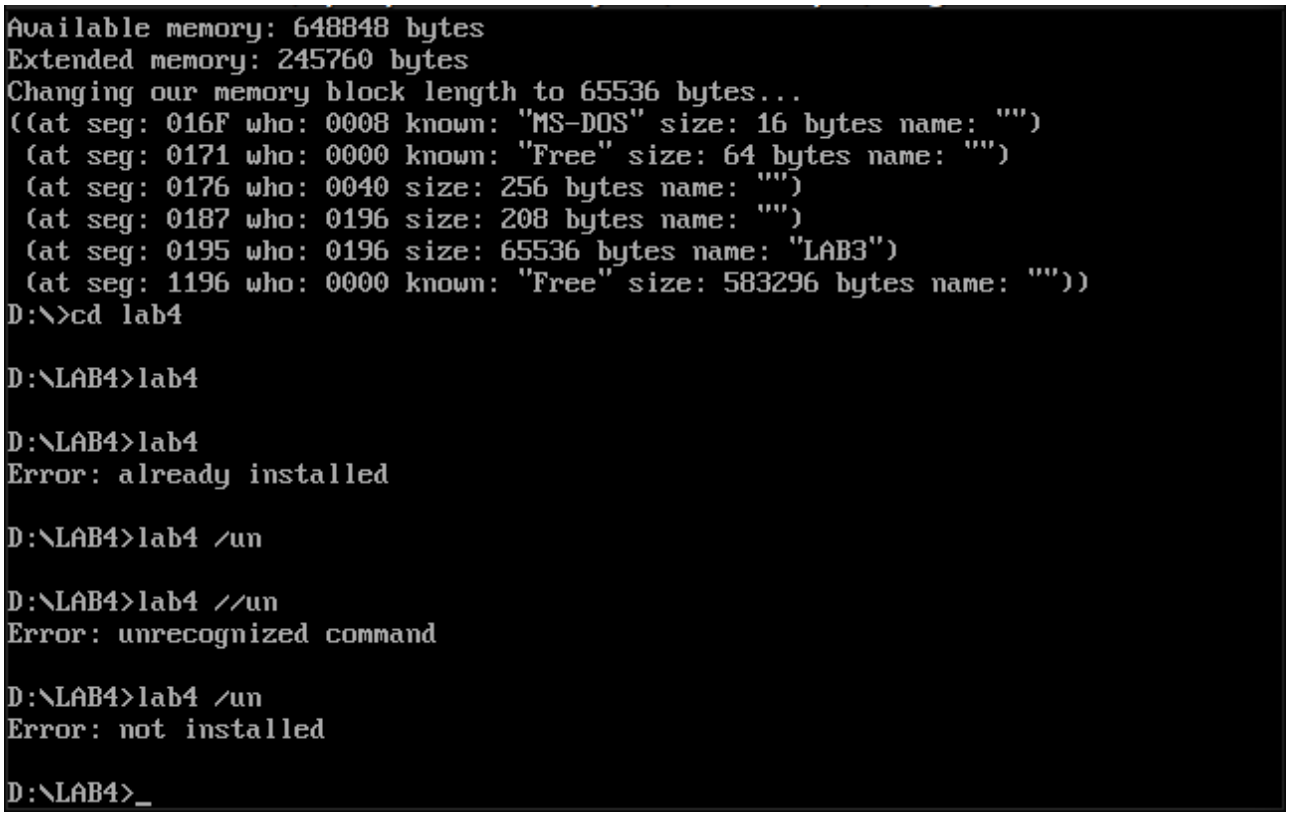


Рисунок 4- Обнаружение резидента и обработка ошибок

Здесь видно, что программа корректно обрабатывает ситуацию, когда запрошена загрузка резидента, но он уже загружен, или его выгрузка, но он не загружен (или загружен, но перекрыт; в этом случае его обнаружить не удастся).

**Контрольные вопросы**

1. Как реализован механизм прерывания от часов?

• Через контроллер прерываний процессору поступает аппаратное прерывание IRQ 0 от системного таймера.

• Процессор получает прерывание от контроллера (оно отображается в прерывание `08h` процессора) и начинает его обрабатывать. Сохраняются значения регистра флагов, `CS` и `IP`. Запрещаются прерывания (сбрасывается флаг `IF` в регистре флагов). Вызывается установленный DOS обработчик.

• DOS сохраняет контекст выполняемой в данный момент задачи и вызывает прерывание `1Ch`. Запускается его обработчик.

• Обработчик прерывания делает всё, что ему нужно, а потом

посылает контроллеру (его шина команд на порту `20h`) команду `20h` (*End of Interrupt*) для разрешения прерываний того же или более низкого приоритета (чем выше номер IRQ, тем ниже приоритет). Обработчик прерывания возвращает управление DOS.

• DOS восстанавливает контекст задачи и возвращает ей управление.

• Процессор восстанавливает старые значения регистра флагов, `CS` и `IP`, которые были сохранены на стеке.

2. Какого типа прерывания использовались в работе?

Аппаратное прерывание 0 (IRQ 0) – прерывание от таймера, поступающее контроллеру; пользовательское прерывание `1Ch`, которое вызывает DOS; программное прерывание `21h`, обрабатываемое DOS; прерывание `10h`, обрабатываемое BIOS.

**Вывод**

В ходе выполнения лабораторной работы была исследована работа с прерываниями, а также была написана резидентная программа для DOS.