**Лекция №5. РОДИТЕЛЬСКИЕ И дочерний процесс**

***Организация дочерних процессов.***

Функция 4Вh. Загрузка и выполнение одной программой другой программы. Эта функция предоставляет такие возможности:

1. Программа может загружать и выполнять другую программу и затем вновь получить управления;
2. Программа может выполнить внутреннюю команду DOS;
3. Может загрузить программу без построения для нее PSP и не передавая ей управление. Эта возможность позволяет реализовать программы по оверлейный структурой.

Во всех трех случаях входные и выходные регистры для функции загружаются одинаково.

входные данные: - AH - 4Bh;

- DS: DX файловая спецификация;

- ES: BX указатель в область с параметрами загрузки;

- ALномер подфункции (00-загрузки и выполнения программы; 03-

загрузки оверлейной программы).

Исходные данные:

* АХ код ошибки при CF = 1; при CF = 0 - не имеет значения. Возможные коды ошибок:
* 1 нет такой функции;
* 2 файл не обнаружено;
* 3 нет такого пути;
* 5 отказано в доступе;
* 8 не хватает памяти;
* 10 недопустимо окружения;
* 11 недопустим формат.

Загрузка и выполнение программы. Регистр AL установлен в 0. Пара ES: BX содержит адрес блока параметров.

Структура блока параметров:

- Поле 1 (одно слово). Сегментный адрес нового окружения;

* Поле 2 (два слова). Указатель к командной строке, который нужно записать в PSP + 80h;
* Поле 3 (два слова). Указатель к FCB, который нужно записать в PSP + 5Ch;
* Поле 4 (два слова). Указатель к FCB, который нужно записать в PSP + 6Ch.

В целом программа, указанная с помощью DS: DX выполняется так же, как и при самостоятельном выполнении. Есть и некоторые различия:

1. Перед тем, как выполнить EXEC, родительская программа должна обеспечить свободный блок достаточного размера для выполняемой программы. В общем случае вся рабочая память после резидентной части COMMAND описана в одном блоке, который присвоен родительской программе. Для того, чтобы освободить место, она должна уменьшить этот блок с помощью функции 4Аh для смены блока памяти. Затем можно выполнить функцию 48h с BX = FFFFh, с получением при этом в BX размера блока памяти, уволился.

2. При выполнении EXEC DOC не осуществляет сохранение и восстановление регистров - это должна делать родительская программа. Для этого она может использовать свой стек, но так как SS и SP тоже меняются, их нужно сохранить в переменных и восстановить сразу после того, как программа вновь получит управление.

3. Открытые методом ФМ файлы наследуются программой которая выполняется с помощью EXEC, если только они не открыты с режимом наследования 1. (функция 3Dh). Это означает, что программа выполняемой может использовать файловые манипуляторы родительской программы. Указатель позиции в каждом открытом файле (FP) сохраняется таким, каким он был в родительской программе. Для того, чтобы использовать файловые манипуляторы, они должны быть известны программе выполняемой работы. Можно, например, передать их номера как параметры в командной строке (указанном в блоке параметров).

4. Когда программа выполняется командным процессором COMMAND, она может найти параметры командной строки в PSP. Параметры может передавать и родительская программа. С этой целью поле 2 блока параметров заполняется адресу файловой спецификации, содержащий параметры. Есть и дополнительная особенность. При выполнении командным процессором COMMAND параметры командной строки

располагаются со смещением 81h в PSP и ограниченные символом CR (0Dh). Общая длина цепочке параметров записана со смещением 80h. При выполнении из другого приложения цепочку с параметрами записывается в PSP выполняемой программы со сдвигом 80h без поля длины перед ним и без символа СR конце. Это может привести к ошибкам, если программа выполняется самостоятельно. Поэтому цепочка параметров должен начинаться байтом длины цепи и завершаться символом CR. Если нет параметров, цепочка параметров должен содержать только символы 00h i 0Dh. Все это имеет смысл только для программ, использующих параметры командной строки.

1. Родительская программа может передавать программе выполняемой копию своего окружения или совершенно новое окружение, которое она построила в своей рабочей области. Для того, чтобы передать копию своего окружения, она записывает 0 поле 1 блока параметров. Для передачи новому окружению она должна записать в поле его сегментный адрес. Новое окружение должно быть действительным, то есть содержать ASCIIZ цепочки в формате имя = переменная и завершаться нулевым байтом. Например, в окружении, как и для программ выполняемых командным процессором COMMAND, располагается файловая спецификация программы (то есть ASIIZ-цепочку, чей адрес находится в DS: DX).
2. Когда программа выполняется командным процессором COMMAND, система использует первые два параметра программного строке как имена файлов и конструирует на их основе два неоткрытых FCB в PSP (со сдвигом 5Ch и 6CH). Когда программа выполняется другой программой, в поле 3 и 4 блока параметров можно записать адреса двух FCB, которые будут построены в PSP при выполнении EXEC. Если не будут передаваться файлы таким методом, поля 3 и 4 должны содержать 0.
3. Адрес завершения в PSP выполняемой программы (со сдвигом 0Ah) указывает адрес после инструкции INT 21h, с помощью которой выполняется EXEC.

***Выполнение внутренней команды***

Это - конкретный случай, когда одна программа выполняет другую. Пользовательская программа вызывает с помощью функции EXEC командный интерпретатор Command.com и передает ему в качестве параметра внутреннюю команду. Таким образом, в

память загружается вторая копия Command (вторичный командный процессор), который выполняет команду, после чего освобождает память и возвращает управление пользовательской программе. Можно скачать и вторичный командный интерпретатор, остается активным до выполнения команды EXIT. В этом случае командный процессор, получив управления, ожидает ввода команды системы с клавиатуры.

Процедура для выполнения внутренней команды состоит из трех шагов:

1. Обеспечивается свободный блок достаточного размера, в который будет загружен Command.com. Для этого обычно используется функция 4Ah, с помощью которой уменьшается блок текущей программы.
2. Строится ASСII-цепочка из параметров для Command такой структуры:

* 1 байт длина значимой части цепочке;
* ХХ байт команда для COMMAND (формат \ С)
* 1 байт 0Dh (CR - символ конца строки). Пример - db 12 / C Dir \* .ASM, 13.

1. Строится блок параметров длиной 7 слов. Адрес цепочке записывается в поле 2 блока параметров. В других полях блока записывается 0.
2. В DS: DX записывается адрес ASCIIZ-цепочке, содержащий файловую спецификацию командного интерпретатора. В ES: BX записывается адрес блока параметров. В AX записывается 4B00h. После этого выполняется прерывание INT 21h.

Загрузка программы как оверлейного модуля.

Регистр AL имеет значение 3. Пара ES: BX содержит адрес блока параметров с такой структурой:

Поле 1. Длина первого слова адреса сегмента для загрузки программы; Поле 2. Длина первого слова - фактор смещения для загружаемой программы

(Только для ЕХЕ файлов).

Программа (оверлейный модуль) загружается без построения PSP для нее и без передачи управления. Обращение к ней осуществляется как в далекую процедуры, а она возвращает управление инструкции RET. Программа не порождает новый процесс.

Перед выполнением ЕХЕС нужно обеспечить достаточно памяти для загрузки программы. Это может быть рабочая область в блоке главной программы или отдельный блок памяти, заявленный главной программе с помощью функции 48h. Перед этим нужно выполнить функцию 4Ah, чтобы уменьшить блок главной

программы. В первом случае главная программа должна вычислить сегментный адрес области для оверлейного модуля, а во втором она получает его при вызове функции 48h. Этот адрес нужно записать в поле первого блока. Система настраивает оверлейный модуль к адресу загрузки в соответствии с фактору сдвига в поле 2 блока параметров.

* общем случае фактор сдвига должен равняться сегментной адресу загрузки модуля, то есть поле 2 должно равняться полю 1.

***Нестандартные приемы передачи параметров в дочерней программы.***

Все изложенные приемы передачи параметров в дочернюю программу удобны тем, что они входят в число стандартных средств. Но в такое средство можно передать только ограниченный объем информации, причем преимущественно символьной. При разработке сложных программных комплексов возникает необходимость в более тесном взаимодействии родительских и дочерних процессов. Например, желательна возможность передачи адресов массивов для того, чтобы открыть доступ дочерней программе в родительских данных. В системе таких средств нет, сами иерархические программы не имеют никакой информации друг о друге.

Для того, чтобы передать дочерней программе адреса своих областей данных, родительская программа может воспользоваться двумя областями памяти, доступными всем программам: свободными векторами прерываний и областью мижзадачних связей.

* таблицы векторов прерываний свободные участки. В частности векторы 60h ... 66h отведенные для прерываний пользователя. Это означает, что программы DOS и BIOS не обращаются к этим векторов, поэтому приложения могут использовать их по своему усмотрению. Но для того, чтобы дочерние и родительские программы могли обмениваться информацией через свободные векторы, для них должен быть обсужден межпрограммный интерфейс - обе программы должны знать, в каких векторах и именно находится.

Поскольку каждый вектор - это 4-х байтовая ячейка, в нее можно записать полный адрес любого данного и тем самым обеспечить доступ дочерней программы к полям данных родительской программы. В родительской программе для реализации описанного

приема нужно использовать функцию 25h (заполнение вектора прерывания), а в дочерней - функцию 35h (получение вектора прерывания).

Кроме свободных векторов для передачи информации между программами можно пользоваться областью мижзадачних связей. Она располагается в конце области данных BIOS по адресам 40h: F0h - 40h: FFh (всего четыре двойных слова). Эта область не используется системой и к ней можно записывать свои данные.

**Обработка прерываний в реальном режиме**

***Обработка прерываний. Взаимодействие прикладных и системных обработчиков прерываний.***

Структура программ обработки прерываний и их взаимодействие с другими программами определяется рядом факторов:

* + прерывания, которое инициализирует обработчик может быть аппаратным (от периферийных устройств) и программным (команда INT)
  + Программа может быть резидентной или транзитной;
  + Вектор обрабатываемого прерывания может быть свободным или использоваться системой;
  + Если вектор уже используется системой, то новый обработчик может полностью заменять его или зчеплятися с ним,
  + В случае сцепления с системным обработчиком новый обработчик может выполнять свои функции к системному или после него.

Чтобы прикладной обработчик получал управление в результате прерывания, его адрес нужно поместить в соответствующий вектор прерывания.

Для этого предназначена функция 25h прерывания INT 21h. В регистр AL помещается номер вектора, модифицируется, а в DS: DX - адрес нового обработчика. Старый содержимое вектора нужно сохранить в выделенных для этого ячейках

* восстановить перед завершением программы (с помощью этой функции). Для получения исходного содержания предусмотрена функция 35h.

В случае, когда нужно внести незначительные изменения или дополнить системный алгоритм программа пользователя "сцепляется" с системной программой обработки

прерываний. При инициализации обработчика, что "сцепляется" с системным, надо сохранить адрес системного обработчика и поместить в вектор прерывания адрес прикладного обработчика.

Если прикладная обработка должна выполняться после системной, то в стеке прерванного процесса оказывается три слова: слово флагов и двосливна адрес возврата в прерванную программу.

Такая структура данных должна быть в верхушке стека, чтобы команда IRET могла вернуть управления к прерванному процессу:

Формат стека:

* IP2 относительный адрес точки возврата в прикладной обработчик;
* CS2 сегментный адрес прикладного обработчика;

флаги:

* IP1 относительный адрес точки возврата к прерванному процессу;
* CS1 сегментный адрес прерванного процесса.

Системный обработчик, закончив обработку прерывания, завершается командой IRET. Эта команда забирает из стека три верхних слова и осуществляет переход по адресу CS1: IP1.

Если прикладная обработка выполняется в системной, то в ней завершающая команда - это команда перехода - передачи управления к системному обработчика. Эта команда не касается стеке.

***Обработчик прерываний от таймера.***

Для того, чтобы приложения могли использовать сигналы таймера, не нарушая при этом работу системных часов, в программу BIOS, которая обслуживает аппаратные прерывания от таймера, включен вызов 1Сh. Прерывание 1Сh ​​передает управление на программу-заглушку BIOS, которая содержит единую команду iret. Пользователь может записать в вектор 1Сh адрес прикладного обработчика сигналов таймера и использовать в своей программе средства реального времени. Перед завершением программы надо восстановить старое значение вектора 1Сh. Текст обработчика можно разместить в любом месте программы, обеспечив невозможность случайного перехода на его строки не в результате прерывания, а по

хода выполнения основной программы. Обработчик прерывания может быть процедурой,

* может начинаться просто с пометки. Важно только, чтобы последней командой обработчика была команда iret.

*Алгоритм обработчика прерывания 1Сh.*

Пример вывода в угол экрана символа с периодичностью 4 раза в 1 сек., Меняя каждый раз его атрибут.

* 1. Сохранить системный вектор (функция 35h возвращает его содержание в регистрах ES: BX).
  2. Записать в вектор 1Сh адрес нового обработчика прерывания (25h). Для этого нужно настроить регистр DS на сегмент, в котором содержится процедура обработчика или на сегмент команд. (DS сохранить в стеке и через стек записать в него содержимое регистра CS).
  3. Начиная с этого момента, прерывания от таймера будут активизировать 18,2 раза в секунду программу нового обработчика int 1Ch. Для задержки всей программы в памяти она должна выполнять какие-то действия, кроме вызова процедуры обработчика. Например, выводить в цикле на экран текст.
  4. Выполнить процедуру обработчика прерывания 1Сh:

4.1 Сохранить в стеке регистры, которые будут использоваться в обработчике. Это важно, потому что переход на программу обработчика осуществляется по команде int 1Ch с системной программы обработки прерываний от таймера. При выполнении процедуры прерывания процессор настраивает должным образом только регистры CS и IP. Содержание всех других регистров отображает состояние системной программы. Если это состояние будет изменен, то после возвращения из нашего обработчика в системную программу, она перестанет функционировать.

4.2 Регистр ES настроить в адрес видеобуферу. В AX - код ASCII символа, который выводится на экран. Надо сделать замену регистра DS на CS. DS указывает не на данные, а на сегмент программы BIOS. В этом случае и данные надо размещать в сегменте кода.

4.3 Выводить символ в одно и то же место экрана с периодической сменой атрибутов.

4.4 Восстановить сохранены в стеке регистры, завершить программу обработчика командой iret.

***Обработка прерываний по <Ctrl> / C <Ctrl> / <BREAK>***

* многих вычислительных системах комбинация клавиш Ctrl / C зарезервирована для принудительного завершения активной программы и передачи управления системе. Но для этого нужно, чтобы система постоянно анализировала коды, поступающих при обработке прерывания от клавиатуры. И система проверяет наличие Ctrl-С во входном потоке на более высоком уровне при выполнении программных запросов. Различные функции системы по-разному реагируют на ввод с клавиатуры Ctrl- С.

Все функции системы делятся на две группы функций: ввода / вывода по номерам 01h-0Ch и другие. Функции с номерами, превышающих 6Ch используют расширители, сетевые программы и другие вокруг системные программы.

Расхождение этих двух групп функций заключается в том, что при вызове функции в / в системы переходит на внутренний стек ввода / вывода, а при вызове всех других функций - на дисковый стек. Наличие в система и двух внутренних стеков обеспечивает ее частичную рентабельность. То есть, при ошибке в процессе исполнения "дисковой" функции системы можно вызвать функцию в / в. Большая часть функции в / в проверяет наличие в буфере клавиатуры кода 03h (Ctrl / C). При обнаружении этого кода выполняется команда INT 23h (завершение текущего процесса). Исключение составляют функции 06h i 07h, нечувствительны к Ctrl / C. Функции 02h и 09h анализируют буфер клавиатуры на наличие Ctrl / C один раз в 64 вызовы.

"Дисковые" функции проверяют нажатия Ctrl / C только когда установлен флаг Break, то есть когда была выполнена команда Break ON.

Нажатием Ctrl / C нельзя завершить чисто вычислительную задачу, а только такую, в которой вызываются системные функции.

При завершении задачи с помощью Ctrl / C могут оставаться не установленными изменены векторы прерывания или неправильно закрыться открытые файлы. Поэтому большинство приложений не используют системный обработчик Ctrl / C, а заменяют его своим (вектор 23h). После завершения задачи желательно восстановить исходное содержание вектора, так как возможно нарушение работы системы при выполнении других задач.

* обработчике вектора 23h можно использовать любые функции системы. При завершении обработчика командой IRET управления вернется в программу в той же точке, где она была прервана. Но в программе обработчика можно предусмотреть переход в любое место программы без выполнения IRET. Это позволяет организовать нажатием Ctrl / C изменения хода выполнения программы, в частности ее корректное завершение.

Есть еще одна возможность вмешательства в ход выполнения программы - нажатие клавиш <Ctrl-Break>.

Системный обработчик прерываний от клавиатуры, входящей в состав BIOS, при обнаружении комбинации клавиш Ctrl-Break передаст управление программе, адрес которой находится в векторе 1Bh. Эта программа также входит в состав BIOS и состоит из единственной команды IRET. Но в процессе начальной загрузки OC, система изменяет содержимое вектора 1Bh, записывая в него адрес своего обработчика. Этот обработчик выполняет следующие действия:

1. Включает 0000h в буфер клавиатуры на место главного символа;

2. Модифицирует указатели буфера так, что они кажутся системе очищенными;

3. Запись флаг Ctrl / Break в ячейку области данных BIOS по адресу 40h: 71h;

4. Запись код Ctrl / C (03h) в буфер драйвера консоли CON, так что драйвер считает, что выполнена комбинация <Ctrl / C>.

В результате система передает управление на вектор 23h, вроде была нажата комбинация клавиш Ctrl / C. Итак, системная обработка Ctrl-C i Ctrl-Вreak происходят почти одинаково.

Особенность обработки Ctrl / С заключается в том, что если перед нажатием Ctrl / С были нажаты какие-либо клавиши и их коды при этом остались в кольцевом буфере клавиатуры, они будут маскировать Ctrl / C.

При обработке Сtrl / Break программа системы, которая обрабатывает прерывания 1Bh, отправляет код не в кольцевой буфер клавиатуры, а непосредственно в драйвер CON.

Таким образом комбинацию Ctrl / Break нельзя замаскировать и кроме того, программа обработки Ctrl / Break очищает кольцевой буфер клавиатуры.

Прикладная программа может заменить содержимое вектора 1Bh адресу собственного обработчика. В этом случае при нажатии Ctrl / Break происходит немедленный

переход на программу обработчика, который может взять на себя управление в любой точке программы в том числе при зацикливания.

Первоначальный смысл вектора 1Bh не восстанавливается системой автоматически при завершении программы. Поэтому в прикладной программе, которая перехватывает вектор 1Bh следует предусмотреть перед ее завершением восстановления первоначального содержания вектора.

Но программа может завершиться аварийно с помощью Ctrl / C. Это значит, что нужно предусмотреть собственный обработчик прерываний по Ctrl / C, в котором перед завершением программы восстанавливается 1Вh.

***Пользовательская процедура для обработки прерывания 24h.***

Стандартную обработку критических ошибок можно заменить пользовательской процедурой путем изменения вектора 24h. Любая такая процедура должна отвечать следующим требованиям:

* 1. Все регистры должны сохранять свои значения.
  2. Нельзя выполнять системные функции за исключением функций от 01h до 12h

и 59h.

* 1. Нельзя менять блок заголовке драйвера.
  2. Процедура должна заканчиваться инструкцией IRET, с помощью которой управление возвращается в систему. При этом AL должен содержать необходимые указания (0 - ignore, 2 - abort, 3 - fial). Управление можно вернуть и непосредственно пользовательской программе, сразу после INT 24h. Для этого необходимо убрать последние три слова с вершины стека (адрес возврата в систему), восстановить регистры пользователя из стека, после чего выполнить инструкцию IRET.

В некоторых случаях удобно построить пользовательскую процедуру для обработки прерывания 24h как надстройку стандартной:

а) Перед тем, как поставить в вектор 24h адрес своей процедуры обработки критических ошибок пользовательская программа сохраняет его содержимое.

б) Получив управление, пользовательская процедура выполняет инструкцию PUSHF

* затем CALL FAR с адресом хранения старого в города вектора 24h в качестве операнда. Таким средство управления передается стандартной процедуре. Она выводит необходимые сообщения и ждет ответа пользователя.

в) Ответ кодируется в AL и управление возвращается пользовательской программе. г) Пользовательская процедура выполняет необходимые действия и возвращает управление. Старый содержание 24h сохраняется в PSP пользовательской программы (сдвиг 12h) и

восстанавливается автоматически по окончании программы.

Изменением вектора 24h активизируется пользовательская процедура для обработки критических ошибок.

* 1. При получении управления процедура сохраняет в полях пользовательской программы полную информацию об ошибке (регистры AH, DI, BP: SI). Можно выполнить и функцию 59h, чтобы получить более полную информацию об ошибке. Эта информация тоже сохраняется для использования позже. После этого процедура записывает 3 в AL (fail) и возвращает управление с помощью IRET. Если ответ fail Не допускается, то система прерывает выполнение программы прерыванием 23h. Стандартную обработку прерывания 23h можно заменить пользовательской программой. Если ответ fail предполагается, система прекращает текущую системную функцию и возвращает управление пользовательской программе с кодом ошибки 83.
  2. Любое обращение к системной функции обрабатывается в зависимости от вида
* результата системной функции. Если функция выполняется в качестве индикации ошибки AL = FFh или CF = 1 и действительно завершилась ошибкой, выполняется функция 59h.
  1. Если код ошибки равен 83, то это критическая ошибка и она обрабатывается в зависимости от информации, сохраненной на этапе 1. Если код отличный от 83, то используется и другая информация об ошибке, возвращена функцией 59h (тип, и возможные действия). Нужно иметь в виду, что процесс может передавать код завершения процесса, его выполнил. Чтобы использовать эту возможность необходимо поместить желаемый код завершения в AL и выполнить функцию 4Сh прерывания 21h для завершения программы. Когда управления будет возвращено родительском процесса, то он выполнит функцию 4Dh прерывания 21h (без входных регистров) и в AL будет получен код завершения, который затем может быть проанализирован. Кроме того, AH будет содержать информацию о том, как завершился викликаемий процесс:

- 0 для нормального завершения; - 1 по Ctrl / Вreak;

* 2 по критической ошибке устройства;
* 3 с помощью функции 31h, что оставляет программу резидентной.

Если программа завершилась с помощью этой функции, то система получает код выхода и он может быть включен в обработчик командным файлом с помощью подкоманды IF.

Код выхода рассматривается как номер ERRORLEVEL. С помощью этой возможности командные файлы могут прекращать обработку и выводить сообщение о возникновении ошибки в одной из запущенных программ.