**Лекция №6. Резидентные программы И обработчиков прерываний**

***Резидентные программы.***

Программы, предназначенные для загрузки и оставленные в памяти (TSR - Termиnate and Stay Resident), обычно состоят из двух частей: инициализуючои и рабочей. В тексте рабочая часть размещается в начале, а инициализуюча - за ней. Запустить резидентную программу можно тремя способами:

а) Вызвать ее оператором CALL как подпрограмму.

б) Использовать механизм аппаратных прерываний.

в) С помощью программного прерывания.

Кроме того, специально для взаимодействия с резидентными программами в системе предусмотрено мультиплексного прерывания 2Fh.

Первое средство требует наличия в памяти текущей активной программы, которая должна образовывать с родительской программой межпрограммный интерфейс.

Например, чтобы упростить структуру и объем транзитных программ можно оформить в виде TSR-программы процедуры, общие для группы транзитных программ.

Второй способ предусматривает активизацию резидентной программы внешним прерыванием (от таймера, клавиатуры, периферийного устройства).

Третий способ иногда используется при отладки резидентных программ с помощью INT 3h.

пример:

Text segment "code"

Assume cs: text, ds: text

Org 100h

myproc proc far

Jmp init

* данные ... entry:

-----; текст резидентной секции iret

myproc endp

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ressize = | equ = $ | - | myproc | ; размер резидентной программы |
| init proc | ; секция | | инициализации | |
| ---- |  |  |  |  |
| mov dx, ( | ressize | + | 10Fh) / 16 | ; размер в пунктах |

mov ax, 3100h

int 21h

init endp

text ends

end myproc

При первом запуске программы с клавиатуры управление передается на начало процедуры myproc. Командой jmp осуществляется переход на секцию инициализации, в которой создаются условия для дальнейшей активизации программы в резидентном состоянии.

Содержательная часть резидентной программы начинается с отметки entry. Активизируется она с помощью аппаратного или программного прерывания или командой CALL.

Связь с резидентной программой.

Для обращения к резидентной программы можно использовать область мижзадачних связей.

По ходу выполнения секции инициализации будущая резидентная программа помещает данные в оговоренные ячейки области мижзадачних связей. Например, в слово 40h: F0h

- относительный адрес команды с пометкой Entry, а в слово 40h: F2h - содержимое сегментного регистра ES. Транзитная программа, желая передать управление резидентной, настраивает регистр ES на начало области данных BIOS и выполняет команду дальнего вызова: call dword ptr ES: 0F0h.

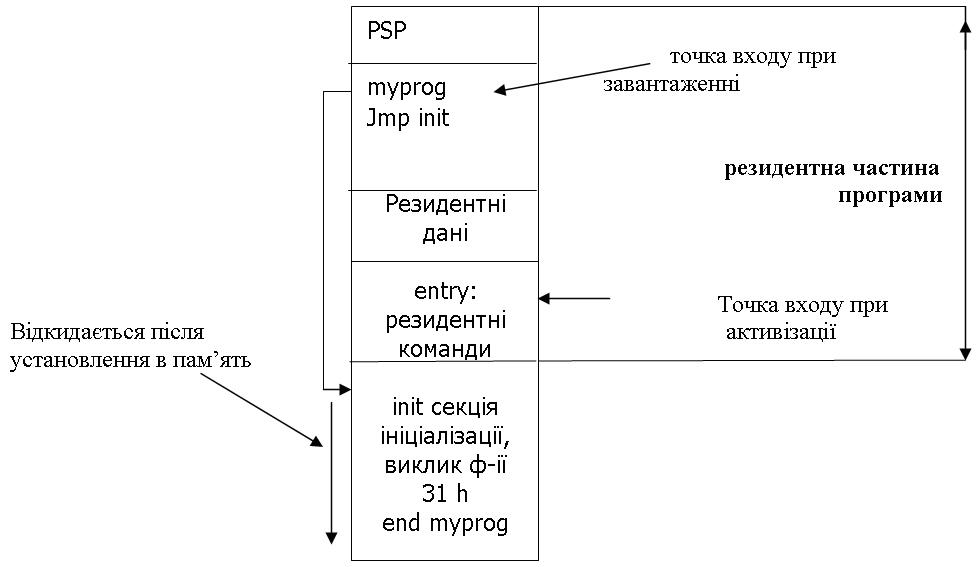


Рис. 1.11 Типовая структура резидентной программы

* + этом случае резидентная программа должна быть объявлена ​​как далека процедура (myproc proc far) и завершаться командой RET дальнего возврата.

Если резидентной программе нужно передать параметры, их начальный адрес заносится программой в другую ячейку области мижзадачних связей, например 40h: F4h. В этом случае резидентная программа переносит например, в регистр SI относительный адрес параметра с ячейки 40h: F4h, а в регистр DS - сегментный адрес параметра с ячейки 40h: F6h и после этого забирает сами параметры:

- mov ax, ds [SI];

- mov bx, ds: [SI + 2] и т.д.

Более изощренный метод передачи резидентной программе управления и параметров

* в использовании свободных векторов.
* процессе инициализации резидентная программа помещает свой адрес в свободный вектор, например 60h. Для ее активизации транзитной программе нужно выполнить команду INT 60h. В этом случае резидентная программа должна заканчиваться командой IRET. Адреса параметров нужно передать через другие свободные векторы.

Еще один способ взаимодействия с резидентными программами - прерывание 2Fh Функции (C0h ... FFh) этого прерывания зарезервированы для прикладных программ.

Это прерывание возвращает в регистр AL состояние резидентной программы: - AL = 0 - программа не установлена;

- DL = 1 - приложение установлено и ее нельзя установить;

- AL = FFh - программа уже установлена ​​и ее повторная установка не требуется. Для того, чтобы резидентная программа отзывалась на прерывание 2Fh, к ней

нужно включить прикладной обработчик одной или нескольких функций этого прерывания. Тогда вызов соответствующей функции этого прерывания в любой транзитной программе позволит организовать взаимодействие с загруженной резидентной программой.

***Проверка на повторное установки.***

Для защиты резидентной программы от повторной загрузки можно включить в загрузочный модуль программы некоторое произвольный код (сигнатуру) и

проверять его наличие в памяти для резидентной программы, запускаемой через аппаратный вектор прерывания (например 09h). Но если будет запущена другая резидентная программа, то в векторе прерывания будет находиться адрес другой программы, в которой данная сигнатура отсутствует.

Обычный метод защиты резидентных программ от повторной загрузки - использование прерывания 2Fh. В регистре АН задается номер функции (от 00h до FFh), а в регистре АL - номер подфункции (в том же диапазоне).

Чтобы резидентная программа могла отозваться на вызов прерывания INT 2Fh,

* ней должен быть обработчик этого прерывания. Он должен проверить номер функции в регистре АН. При обнаружении "своей" функции - проанализировать содержание регистра AL, выполнить вызваны действия, затем вернуть управления вызывая программе (IRET). Если функция в АН "чужая", нужно передать управление том обработчике, адрес которого был ранее в векторе 2Fh. Обработчик системы - самый последний в цепи. Обычно при проверке на повторное установление резидентная программа уже находится в памяти. Она возвращает в регистр AL значение FFh, что является признаком запрета повторной загрузки.

Можно вернуть какие-то обусловлены коды в регистрах. Например, имя программы в регистрах CX и DХ. (CX - "DU", DX - "MP"; DUMP). Возможно, что выбранная вами функция уже используется другой резидентной программой. Тогда в AL будет значение FFh, а в CX и DX другое значение. В этом случае нужно изменить функцию.

***Выгрузка резидентной программы из памяти.***

* системе нет средств выгрузки резидентных программ. Поэтому сама резидентная программа должна позаботиться о своем выгрузки из памяти. Обычно выгрузки резидентных программ осуществляется соответствующей командой, поданной с клавиатуры и воспринимаемого резидентной программой. Для этого резидентная программа должна перехватывать прерывания, поступающие с клавиатуры и "вылавливать" команды выгрузки. Другой способ заключается в запуске некоторой программы, передает резидентной программе команду выгрузки. Чаще всего в качестве программы которая

выгружается используют вторую копию резидентной программы, будучи запущена в определенном режиме, выгружает свою копию.

Пусть, например, резидентная программа имеет имя Dump.com. Команда dump загружает ее в память, оставляя резидентной, а команда dump off "дезактивирует" программу и выгружает ее из памяти. Это означает, что в PSP со смещением 80h будет записана следующая информация 4, 'off', 13. где:

* 4 длина цепочки;
* "Off" цепочку;
* 13 символ CR.
* этом случае программа должна анализировать наличие и содержание параметров ее запуска в PSP. Обнаружив там слово "off", вызвать функцию выгрузки (например, "свою" функцию 01h мультиплексного прерывания 2Fh).

Выгрузка можно осуществлять двумя способами:

- освобождением памяти, которую занимает программа и блоки ее окружения с помощью функции 49h;

- использовать в программе, выгружает, функцию 4Ch и заставить ее завершить не саму выгружая программу, а резидентную программу и затем вернуть управление программе, что выгружает.

* любом случае перед освобождением памяти нужно восстановить все векторы прерываний, которые были перехвачены резидентной программой. Но надежно можно выгрузить только последнюю из загруженных программ, так как вектор, перехваченный резидентной программой, мог быть затем перехваченным другой резидентной программой.

Рассмотрим случай выгрузки резидентной программы с помощью функции 4Ch. Функция 4Ch завершает выполнение программы. Но откуда система знает, какую именно программу нужно завершить и куда передать управление? Эта информация хранится в PSP выполняемой программы. В слове со смещением 16h от начала PSP сохраняется сегментный адрес PSP родительского процесса. В двосливний ячейке со сдвигом 0Ah - конкретный адрес возврата в него. А в области текущих данных - SDA (Swappable Data Area) в слове со смещением 10h сохраняется идентификатор текущего процесса. Адрес SDA можно получить с помощью недокументированные функции 5D06h.

Функция возвращает в регистры DS: SI адрес SDA, а в регистр CX - ее размер. Если компьютер не выполняет никакую программу, текущим является Command.com, и его ID записан в SDA.

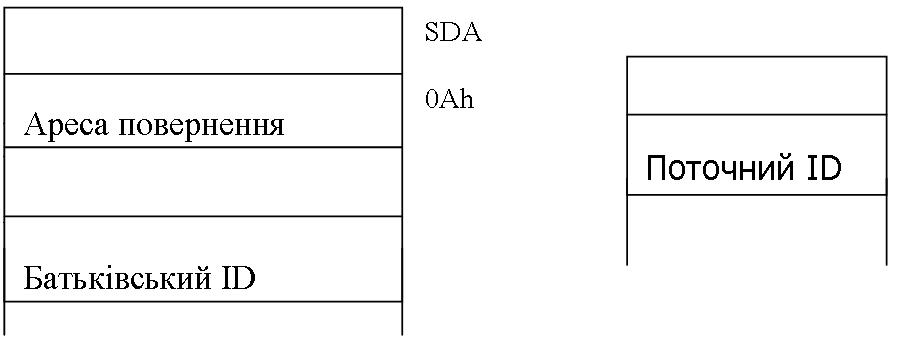


Рис. 1.12 PSP текущий процесс

При запуске с клавиатуры любой программы (в том числе и резидентной) система записывает в SDA ее ID, а в PSP запущенной программы - ID родительского процесса и адрес возврата в него.

Этим обеспечивается возможность завершения запущенной программы и возврата в систему. Для всех программ, запущенных с клавиатуры, родительским процессом будет Сommand.com.

Функция 4Ch выполняет действия по закрытию файлов, установке векторов 22h, 23h, 24h и освобождению памяти, которая занимается текущим процессом. Эта функция переносит в SDA содержание слова PSP со смещением 16h, назначая родительский процесс текущим и передает управление по адресу прерывания, записанному в PSP. Таким образом, чтобы с некоторой программы завершить другую (резидентную), нужно выполнить следующие действия:

* Занести в SDA ID заключительной программы, чтобы объявить ее текущей и чтобы 4Ch завершила именно ее.
* Занести в слово со смещением 16h в PSP программы которая завершается, ID программы, выгружает, чтобы после завершения резидентной программы текущей снова стала выгружая программа.
* Занести в двосливну ячейку со сдвигом 0Ah в PSP программы которая завершается необходимый адрес возврата в программу, выгружает.
* Восстановить кадр стека программы, выгружает, сегментные регистры и регистры общего назначения (функция 4Ch разрушает все регистры).

***Использование средств BIOS в обработчиках аппаратных прерываний.***

BIOS является нерентабельной программой. Но ее нерентабельности вызывается различными причинами. Прерывание BIOS нерентабельные только по отношению к самим себе. Таким образом, вызов функции BIOS возможен в обработчиках только тогда, когда прерывая программа не выполняет функции того прерывания, к которому мы хотим обратиться в обработчике.

Для этого в состав резидентной обработки включаются секции перехвата всех прерываний BIOS, которые предполагается использовать в обработчике. Входные адреса этих секций заносятся в соответствующие векторы прерываний. Сами программы перехватчиков строятся так, что часть своей работы они выполняют перед системными обработчиками данного прерывания, а часть - после. Перехватчик прерываний BIOS устанавливает флаг "занятости" данного прерывания, а после возвращения из BIOS он сбрасывается. Основная программа обработчика обращается к BIOS только в том случае, если флаг сброшен. Если BIOS оказывается занятой, обработчик должен установить флаг незавершенности и вернуть управление в прерванную программу. Обработчик нужно вызвать снова, пока он не обнаружит, что BIOS свободна. Тогда обработчик должен сбросить флаг незавершенности, выполнить нужную функцию BIOS и вернуть управление в прерванную программу. Для таких повторных вызовов обычно используются прерывания от таймера.

***Использование средств системы в обработчиках аппаратных прерываний***

Чтобы разобраться в причинах нерентабельности системы и в методах преодоления этого недостатка, рассмотрим часть внутренней организации системы. Среди системных областей есть структура, которая называется областью текущих данных (SDA - Swappable Data Area). Это область объемом 2К, адрес которой можно получить с помощью недокументированные функции 5D0h.

Функция возвращает в DS: SI адрес SDA, в CX-ее размер. Рассмотрим поля SDA, которые имеют отношение к разработке резидентных программ. Флаг критической ошибки ErrorMode устанавливает в 1, если зафиксировано состояние критической ошибки,

затем вызывается Int 24h. Этот флаг находится в SDA со смещением 00h. Его длина 1 байт. Другой флаг - флаг занятости системы - смещение 01h, длина 1 байт, наз. InDOS. Он устанавливается диспетчером системы сразу после анализа номера функции системы вызвавшей ее и сбрасывается перед возвращением в приложение.

Функции системы в приложении можно вызвать, когда флаг InDOS сброшен. Нерентабельности система и связана с тем, что при выполнении большинства функций используется один и тот же системный стек. Если выполнение любой системной функции системы будет прервано аппаратным прерыванием, и в обработчике прерываний будет вызвана функция из той же группы (то есть та, что использует тот же стек), то в SS: SP будут загружены те же значения,

* при первом вызове. В результате вложенный вызов будет затирать те данные, которые были в стеке. При вызове функции из другой группы ничего не случится. Наличие кадров в SDA не только прошлого, но и позапрошлого стеке, позволяет правильно обрабатывать один вложенный вызов.

Фактический адрес флага InDOS можно получить с помощью функции системы 34h. Она возвращает двосливну адрес флага InDOS в регистрах ES: BX. Получив и сохранив этот адрес на этапе инициализации обработчика прерываний, в самом обработчике перед выполнением любой функции системы нужно выполнить проверку флага занятости.

Но, когда возможно возникновение критической ошибки в обработчике аппаратного прерывания, перед вызовом любой функции системы, нужно проверить еще и состояние флага ErrorMode. Если хотя бы один из них не равен 0, вызвать систему нельзя. Определение адреса флага критической ошибки нужно выполнить на этапе инициализации обработчика.

Такая методика часто непригодна. Например, если текущая программа ожидает ввода с клавиатуры, она не выходит из соответствующей функции и флаг InDOS непрерывно установлен. В такой ситуации обработчик никогда не сможет вызвать нужную функцию.

* + системе включено специальное прерывание int 28h, которое вызывается функциями ввода с клавиатуры. Прикладная программа может поместить в вектор 28h адрес своей процедуры обработки этого прерывания. Поскольку прерывания 28h

возникает только при выполнении функций, использующих стек ввода / вывода, в обработчике прерываний доступен вызов функций 0dh-6Ch. Таким образом, прикладной обработчик аппаратных прерываний, в котором вызываются функции, должен иметь:

* активизаторы по прерыванием от таймера (вектор 08h)
* активизаторы по прерыванию 28h;

Пример обработки прерываний 28h:

|  |  |  |
| --- | --- | --- |
| in\_dos | dd 0 |  |
| crit\_err | dd 0 |  |
| task\_req | dd 0 |  |
| old\_28h | dd 0 |  |
| new\_28h: cmp cs: task\_req, 1 | | ; task требует запуска? |
| jne bx, cs: in\_dos | ; нет, завершить обработчик InDOS | |
| les bx, cs: in\_dos | ; так, получим адрес Прапа. InDOS | |
| lds si, cs: crit\_err | ; адрес флага ErrorMode | |
| cmp ds: [si] 0 | ; флаг ErrorMode свергнут? | |
| jne out\_28h | ; нет, ждем | |
| cmp es: [bx], 1 | ; флаг InDOS не более 1? | |
| ja out\_28h | ; больше, вложенный вызов. | |
| dec cs: task\_req | ; сброс флага требования запуска | |
| call tast | ; |  |
| out\_28h: jmp cs: old\_28h | | ; заканчиваем обработчик |

Прикладной обработчик прерываний 28h прежде всего проверяет, установлен ли флаг запроса запуска задачи. Если этот флаг сброшен, возвращаемся в прерванную функцию системы командой jmp cs: old\_28h - передачи управления на этот обработчик, адрес которого была изъята из вектора 28h.

Если флаг запроса запуска задачи установлен, сначала проверяем состояние флага критической ошибки и если он сброшен, то проверяем состояние флага занятости системы. Этот флаг должен быть равен 1, это значит, что уже вызвана функция системы и более вызывать систему нельзя. после всех

проверок вызывается процедура, которая выполняет вызовы системы. Затем управление передается предыдущем обработчике int 28h.

Вызов int 28h выполняется функцией системы на стеке в / в. Поэтому в прикладном обработчике прерывания 28h можно вызвать только функции дисковой группы. Кроме этого, недопустим вызов файловых функций с указанием стандартных дескрипторов клавиатуры и экрана (0-2). Чтобы выполнить в \ в через терминал можно открыть терминал как файл и получить от системы дескриптор использовать для операций в \ в. Другой способ - вызвать функции BIOS.

***Асинхронная активизация резидентных программ командами с клавиатуры.***

Для управления резидентной программой командами, подаваемыми с клавиатуры, в ее состав нужно включать обработчик прерываний от клавиатуры (09h). Выполнять резидентную программу, находясь "внутри" такого обработчика, можно только тогда, когда в программе не используются функции BIOS. Таким образом, в обработчике прерываний от клавиатуры только выполняется анализ введенной команды и устанавливается флаг запроса запуска задачи, если команда отвечает запроса на активизацию. Сама же активизация задачи осуществляется в обработчике Int 28h.

***Работа с файлами в резидентном обработчике аппаратных прерываний.***

Работа с файлами в резидентном обработчике аппаратных прерываний осложняется тем, что при переходе в обработчик текущей остается программа которая была прервана. Дескрипторы файлов которые открываются создаются системой в JFT ее PSP. Для того, чтобы дескрипторы файлов, открываемого в обработчике, полностью принадлежащих ему необходимо при входе в обработчик объявить его текущей программой для системы. Для манипуляции с идентификаторами программ в системе существуют функции 50h (установить PSP) и 51h (получить PSP). Обе функции не используют системные стеки (они являются рентабельными) и их можно использовать в обработчиках аппаратных прерываний.

При вызове в обработчике функций 4Eh и 4Fh (поиск файлов), которые используют область DTA, переключение ID (идентификаторов) программ

недостаточно. При входе в обработчик надо сохранить адрес текущей DTA и установить в качестве текущей DTA обработчика, а при выходе - восстановить DTA прерванной программы.

Для этого используются функции 2Fh и 1Ah. Они нерентабельны, поэтому перед их вызовом в резидентном обработчике прерываний надо использовать функцию 59h, которая получает из SDA расширенную информацию об ошибке.

Перед вызовом функции системы в обработчике надо получить с SDA расширенную информацию об ошибке, а перед выходом из обработчика восстановить ее с помощью функции 5D подфункции 0Ah.

Эта функция позволяет записать в поле SDA расширенную информацию об ошибке, полученную в поле функции 59h.

* SDA поступают значения, соответствующие регистрам ax, bx, di, es. Эти данные записываются в поля SDA со сдвигом в соответствии 04h, 06h, 08h, 0Ah.

***Свопинг области текущих данных.***

Наиболее радикальными методами предоставления системе свойств рентабельности является хранение всей SDA в прикладном обработчике перед вызовом любых системных функций и восстановления ее после завершения работы с системой. Такая процедура называется свопингом.

Перед выполнением подкачки целесообразно проверить состояние флага InDOS, так если флаг сброшен, в подкачки нет необходимости.