Лабораторная работа № 8

Создание резидентных программ на ассемблере

Цель настоящей работы - изучение принципов построения резидентных программ.

1. Общие принципы организации резидентных программ

Резидентной программой (или TSR-программой, от слов: Terminate but Stay Resident) называется программа, которая остается в памяти ЭВМ в то время, как операционная система выполняет свои обычные функции: исполняет команды, запускает другие (фоновые) программы и т.п. Резидентными являются многие системные программы. В системе MS-DOS пользователь имеет возможность запускать и собственные резидентные программы.

Чтобы программа была резидентной, нужно, чтобы операционная система считала занятой ту область памяти, в которой размещается эта программа, и не отдавала эту память для других целей. При этом единственным способом для резидентной программы проявить каким-то образом себя на фоне обычной работы операционной системы является реакция на аппаратные или программные прерывания. Таким образом, резидентная программа - это всегда программа обработки прерываний.

Наиболее часто резидентные программы используют прерывания по работе с клавиатурой (аппаратные или программные). Такая программа активизируется при нажатии специфичной для нее комбинации клавиш. Используются также прерывания от таймера, других устройств, программные прерывания BIOS и функции DOS.

2. Структура резидентных программ и их запуск

Резидентная программа состоит, как правило, из нерезидентной и резидентной частей. Задача нерезидентной части - выполнить действия по начальной установке (загрузке) TSR-программы. Главное здесь - запоминание прежних значений перехватываемых векторов прерываний и установка новых значений, т.е. перехват прерываний. Кроме того, здесь может выполняться инициализация переменных, используемых затем при резидентной работе, чтение адреса флага DOS, выделение динамической памяти и другие начальные действия.

Резидентная часть программы представляет собой одну или несколько процедур обработки перехваченных прерываний. Желательно, чтобы при любых условиях такая процедура вызывала также и прежнюю подпрограмму, на которую указывал вектор до перехвата. Даже если программе не нужна стандартная обработка прерываний, может случиться, что вектор был ранее перехвачен другими TSR-программами, тогда по одному прерыванию могут успешно вызываться несколько программ.

Если старый вектор сохранен в памяти (скажем, в двойном слове OldVector), то его вызов из TSR-программы выполняется либо командами

```
pushf
call    OldVector
```

(при этом возврат происходит в TSR-программу), либо командой

```
jmp     OldVector
```

(с возвратом в прямо в фоновую программу).

Как и всякая подпрограмма обработки прерываний, TSR-программа должна сохранять и затем восстанавливать значения всех регистров, за исключением тех регистров, через которые соответствующее программное прерывание возвращает результаты. Надо также иметь в виду, что многие прерывания и функции DOS изменяют значения некоторых флагов, поэтому перехватывающие их TSR-программы должны обеспечивать правильные значения регистра флагов.

Важный момент при запуске TSR-программы - предотвратить повторную загрузку программы, если она уже была ранее загружена. Повторная загрузка, как минимум, без пользы уменьшает размер памяти, доступной DOS, а часто приводит и к неверной работе TSR-программы (необходимые действия выполняются два раза). Чтобы предотвратить это, программа еще до перехвата векторов должна проверить, не были ли они уже перехвачены при предыдущем запуске той же программы. Для этого обычно проверяются ключевые значения, хранящиеся на определенном смещении в сегменте из перехватываемого вектора. Если этот ключ совпадает с заданным (например, это может быть определенное число или даже название самой программы), то вектор, очевидно, уже указывает на данную программу, т.е. он перехвачен. Такой способ не дает стопроцентной гарантии, поскольку возможно, что векторы были второй раз перехвачены другой программой, сквозь которую трудно разглядеть первый перехват. Иногда применяют более радикальное решение: ищут ключ в ОЗУ, перебирая подряд все значения адреса сегмента, меньшие, чем текущий адрес PSP.

В конце работы нерезидентной части она должна выполнить функцию DOS, которая завершает выполнение программы, оставляя заданную часть резидентной.

Функция 31h: Завершить, оставив резидентной.
---- Аргументы:
AL - код завершения.
DX - размер резидентной части в параграфах.
---- Результаты: нет.

Код завершения принимается равным 0, если программа проработала нормально.

Размер резидентной части отсчитывается от начала PSP (а не от начала кодового сегмента!). Для уменьшения занимаемой памяти принято размещать резидентную часть в начале программы, а нерезидентную - в конце. Чтобы подсчитать требуемый размер, обычно используют метку, которую ставят сразу после конца резидентной части. Смещение этой метки относительно CS делят на 16, получая число параграфов, которое для EXE-программы следует еще увеличить на разность между значением в CS и адресом PSP. Напомним, что в момент запуска EXE-программы адрес PSP содержится в регистрах DS и ES (а для COM-программ еще и в CS и SS).

Иногда предусматривается возможность выгрузки TSR-программы с освобождением занимаемой ею памяти. Здесь, однако, есть серьезные проблемы в том случае, если после данной TSR-программы была загружена другая TSR-программа, тем более перехватывающая те же векторы. Поэтому чаще TSR-программа остается в памяти вплоть до перезагрузки DOS. Гораздо проще предусмотреть возможность деактивации TSR-программы без ее выгрузки. Для этого надо по определенной комбинации клавиш устанавливать в программе флаг, заставляющий ее при последующих вызовах просто вызывать старую подпрограмму обработки прерывания без вмешательства в ее работу.

3. Работа с данными и стеком в резидентных программах

В тот момент, когда TSR-программа получает управление по прерыванию, значения в регистрах DS, SS, и ES принадлежат фоновой программе и могут быть произвольными. Это требует осторожности при использовании стека и данных в памяти.

Один из вариантов работы с данными - вообще не использовать в резидентной программе сегмент данных и адресовать переменные в памяти относительно CS, задав в программе директиву:

```
ASSUME  cs:code,ds:NOTHING
```

и описав все переменные в сегменте кода.

Другой вариант - использовать сегмент данных как обычно, но в резидентной части при каждом вызове записывать в DS адрес своего сегмента данных (разумеется, после сохранения прежнего значения DS).

Резидентная программа может использовать для своих нужд стек фоновой программы, работавшей в момент вызова, при этом TSR не изменяет значение регистра SS. При аккуратном обращении со стеком это может вызвать ошибку только в случае переполнения стека фоновой программы. Если программист достаточно осторожен, чтобы этого опасаться, он может зарезервировать место для стека в резидентной части программы, в виде отдельного сегмента или в сегменте кода. При вызове TSR следует тогда сохранить прежние значения SS и SP, но только не в старом стеке!

Впрочем, осторожных программистов немного.

4. Использование функций DOS в резидентных программах

Общее правило гласит: использовать функции DOS в TSR-программах нельзя. Более точно запрет формулируется так: нельзя вызывать функцию DOS в момент, когда еще выполняется предыдущий вызов какой-либо функции DOS. Как принято говорить, прерывание 21h нереентерабельно, т.е. его нельзя вызывать второй раз, пока не закончена обработка первого вызова. А поскольку при вызове TSR трудно гарантировать, что система не обрабатывала в момент вызова какую-либо функцию (особенно если TSR обрабатывает аппаратное прерывание), то из этого и вытекает запрет на использование функций DOS в TSR-программах.

Еще более точная формулировка: при выполнении функций 01h - 0ch нельзя вызывать функции из этого же диапазона номеров, но можно вызывать остальные функции, и наоборот: при выполнении остальных функций можно использовать только функции 01h - 0ch.

Причина нереентерабельности функций DOS в том, что при входе в прерывание 21h система устанавливает сегмент стека SS на собственный стек DOS, сохраняя прежнее (пользовательское) значение SS в отдельной переменной (а не в стеке, что кажется более естественным, но не так-то просто осуществить!). В результате при повторном вызове в эту переменную будет записано значение SS перед вторым вызовом, и оно затрет значение пользовательского SS, делая невозможным возврат в вызвавшую программу.

Несмотря на досадный запрет, резидентные программы нередко все же используют функции DOS, соблюдая при этом определенные предосторожности.

При входе в прерывание 21h система записывает значение 1 в определенный байт ОЗУ, называемый флагом DOS. Перед выходом из прерывания 21h флаг DOS обнуляется. Учитывая это обстоятельство, резидентная программа может использовать следующую стратегию. При вызове (например, по нажатию комбинации клавиш) TSR-программа не

пытается выполнить функцию DOS, а только устанавливает свой собственный флаг, означающий наличие запроса на выполнение. Другой же блок TSR-программы, работающий по прерываниям от таймера, постоянно проверяет наличие запроса и флаг DOS. Как только флаг DOS становится равным 0, запрос выполняется и флаг запроса сбрасывается.

Адрес флага DOS может быть получен еще до запуска TSR с помощью функции DOS 34h.

Функция 34h: Адрес флага DOS.
---- Аргументы: нет.
---- Результаты:
ES:BX - адрес флага DOS.

К сожалению, эта функция не упомянута в официальной документации по MS-DOS, а это значит, что ее правильная работа не гарантируется, тем более в будущих версиях MS-DOS. Тем не менее, она довольно широко используется. Если программист все же не хочет использовать недокументированную функцию, то он без труда сам может отслеживать состояние DOS, перехватывая прерывание 21h.

К еще большему сожалению, проверка флага DOS еще не решает проблему использования функций DOS в TSR-программах. Беда в том, что некоторые функции DOS, а именно функции, работающие с клавиатурой, могут выполняться неограниченно долго. Это относится прежде всего к функции ввода строки 0ah, которая может "висеть" часами, пока пользователь не нажмет "Enter". В этих условиях TSR-программа будет вынуждена ждать окончания ввода, что не всегда приемлемо.

Многие резидентные программы используют еще одну недокументированную возможность - перехват прерывания 28h. Это прерывание вызывается не программами пользователя, а самой DOS в те моменты, когда функции 01h - 0ch ожидают нажатия клавиши. Фактически при этом выполняется цикл, состоящий из проверки наличия нажатой клавиши и, если ее нет, вызова прерывания 28h. Таким образом, если программа пользователя перехватила прерывание 28h, то можно быть уверенным, что в этот момент DOS находится в состоянии выполнения одной из функций 01h - 0ch, и поэтому можно безбоязненно вызывать остальные функции DOS.

Напишите программу, которая бы выводила каждую минуту сообщение «Привет!».