ЛЕКЦІЯ №6. РЕЗИДЕНТНІ ПРОГРАМИ ТА ОБРОБНИКИ ПЕРЕРИВАНЬ

Резидентні програми.

Програми, призначені для завантаження і залишені в пам'яті (TSR – Terminate and Stay Resident), зазвичай складаються з двох частин: ініціалізуючої і робочої. У тексті робоча частина розміщується на початку, а ініціалізуюча - за нею. Запустити резидентну програму можна трьома засобами:

а) Викликати її оператором CALL як підпрограму.

б) Використовувати механізм апаратних переривань.

в) За допомогою програмного переривання.

Крім того, спеціально для взаємодії з резидентними програмами, у системі передбачено мультиплексне переривання 2Fh.

Перший засіб потребує наявності у пам'яті поточної активної програми, що повинна утворювати з батьківською програмою міжпрограмний інтерфейс.

Наприклад, щоб спростити структуру та обсяг транзитних програм можна оформити у вигляді TSR–програми процедури, загальні для групи транзитних програм.

Другий засіб передбачує активізацію резидентної програми зовнішнім перериванням (від таймера, клавіатури, периферійного пристрою).

Третій засіб іноді використовується при відлагодженні резидентних програм за допомогою INT 3h.

Приклад:

```
Text segment "code"
Assume cs:text , ds:text
Org 100h
myproc proc far
Jmp init
; дані …
entry:
----- ; текст резидентної секції
iret
myproc endp
ressize = equ = $ - myproc   ; розмір резидентної програми
init proc  ; секція ініціалізації
----
mov dx ,( ressize + 10Fh )/16  ; розмір у параграфах
```

```
mov ax , 3100h
int 21h
init endp
text ends
end myproc
```

При першому запуску програми з клавіатури керування передається на початок процедури myproc. Командою jmp здійснюється перехід на секцію ініціалізації, у якій готуються умови для подальшої активізації програми в резидентному стані.

Змістовна частина резидентної програми починається з помітки entry. Активізується вона за допомогою апаратного або програмного переривання або командою CALL.

Зв'язок з резидентною програмою.

Для звернення до резидентної програми можна використовувати область міжзадачних зв'язків.

По ходу виконання секції ініціалізації майбутня резидентна програма поміщає дані в обумовлені чарунки області міжзадачних зв'язків. Наприклад, у слово 40h:F0h – відносну адресу команди з поміткою Entry, а в слово 40h:F2h – вміст сегментного регістру ES. Транзитна програма, бажаючи передати керування резидентній, налаштовує регістр ES на початок області даних BIOS і виконує команду дальнього виклику: call dword ptr ES:0F0h.

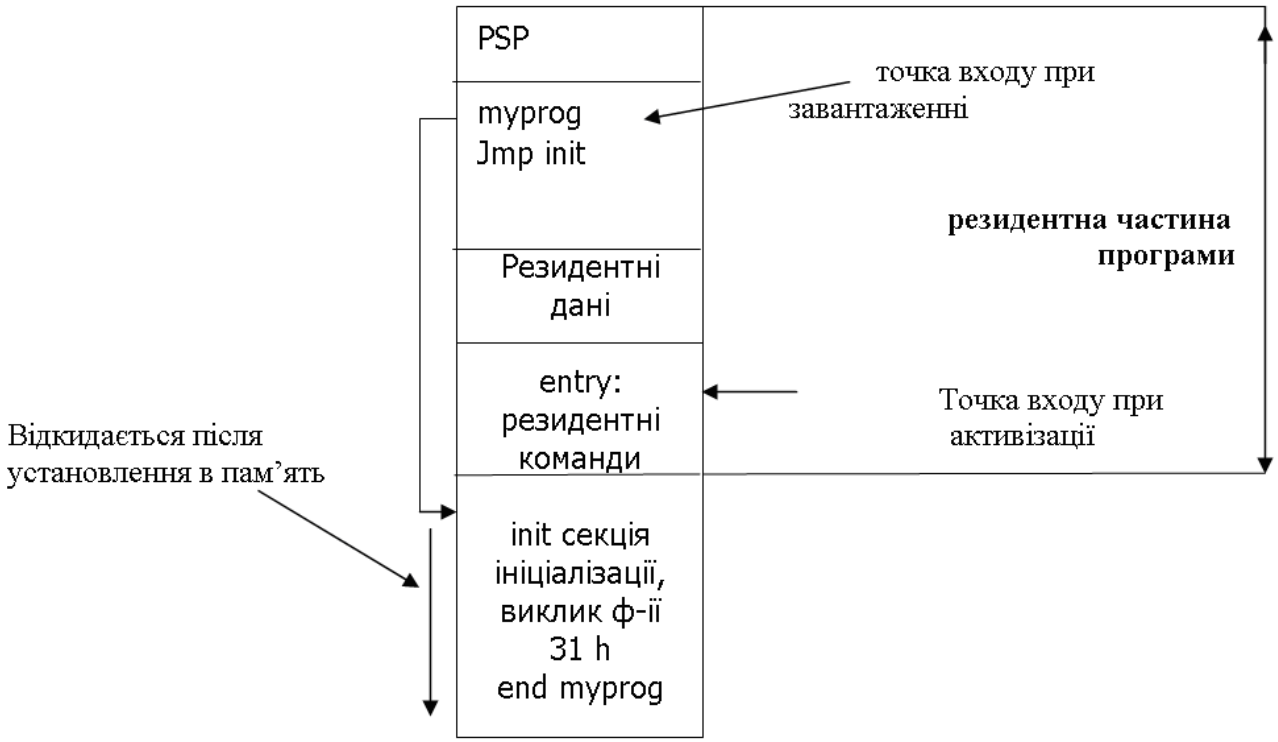

Рис. 1.11 Типова структура резидентної програми

В цьому випадку резидентна програма повинна бути оголошена як далека процедура (myproc proc far) і завершуватися командою RET дальнього повернення.

Якщо резидентній програмі потрібно передати параметри, їхня початкова адреса заноситься програмою в іншу чарунку області міжзадачних зв'язків, наприклад 40h:F4h. У цьому випадку резидентна програма переносить наприклад, у регістр SI відносну адресу параметру з чарунки 40h:F4h, а в регістр DS – сегментну адресу параметру з чарунки 40h:F6h і після цього забирає самі параметри:

- mov ax,ds [SI];

- mov bx,ds:[SI + 2] і т.д.

Більш витончений метод передачі резидентній програмі керування і параметрів - у використанні вільних векторів.

У процесі ініціалізації резидентна програма поміщає свою адресу в вільний вектор, наприклад 60h. Для її активізації транзитній програмі потрібно виконати команду INT 60h. У цьому випадку резидентна програма повинна закінчуватися командою IRET. Адреси параметрів потрібно передати через інші вільні вектори.

Ще один засіб взаємодії з резидентними програмами - переривання 2Fh

Функції (C0h…FFh) цього переривання зарезервовані для прикладних програм. Це переривання повертає в регістр AL стан резидентної програми:

- AL=0 - програма не встановлена;

- DL=1 – програма не встановлена і її не можна встановити;

- AL=FFh – програма уже встановлена й її повторне встановлення не потрібне.

Для того, щоб резидентна програма відгукувалась на переривання 2Fh, до неї потрібно включити прикладний обробник однієї або декількох функцій цього переривання. Тоді виклик відповідної функції цього переривання в будь-якій транзитній програмі дозволить організувати взаємодію з завантаженою резидентною програмою.

Перевірка на повторне встановлення.

Для захисту резидентної програми від повторного завантаження можна включити в завантажувальний модуль програми деякий довільний код (сигнатуру) і

перевіряти його наявність у пам’яті для резидентної програми, що запускається через апаратний вектор переривання (наприклад 09h). Але якщо буде запущена інша резидентна програма, то у векторі переривання буде знаходитися адреса іншої програми, у якої дана сигнатура відсутня .

Звичайний метод захисту резидентних програм від повторного завантаження - використання переривання 2Fh. В регістрі AH задається номер функції (від 00h до FFh), а в регістрі AL – номер підфункції (у тому ж діапазоні).

Щоб резидентна програма могла відгукнутися на виклик переривання INT 2Fh, у ній повинен бути обробник цього переривання. Він повинен перевірити номер функції в регістрі AH. При виявленні “своєї” функції – проаналізувати вміст регістра AL, виконати викликані дії, потім повернути керування викликаючій програмі (IRET). Якщо функція в AH “чужа”, потрібно передати керування тому обробнику, адреса якого була раніше у векторі 2Fh. Обробник системи - найостанніший у ланцюгу. Звичайно при перевірці на повторне встановлення резидентна програма вже знаходиться в пам’яті. Вона повертає в регістр AL значення FFh, що є ознакою заборони повторного завантаження.

Можна повернути якісь обумовленні коди в інших регістрах. Наприклад, ім’я програми в регістрах CX і DX. (CX – “DU”, DX – “MP”; DUMP). Можливо, що обрана вами функція вже використовується іншою резидентною програмою. Тоді в AL буде значення FFh , а в CX і DX інше значення. У цьому випадку потрібно змінити функцію.

Вивантаження резидентної програми з пам’яті.

У системі немає засобів вивантаження резидентних програм. Тому сама резидентна програма повинна подбати про своє вивантаження з пам’яті. Зазвичай вивантаження резидентних програм здійснюється відповідною командою, поданою з клавіатури і сприймаємою резидентною програмою. Для цього резидентна програма повинна перехоплювати переривання, які надходять з клавіатури і “виловлювати” команди вивантаження. Інший засіб полягає в запуску деякої програми, що передає резидентній програмі команду вивантаження. Найчастіше в якості програми яка

вивантажується використовують другу копію резидентної програми, яка будучи запущена в визначеному режимі, вивантажує свою копію.

Нехай, наприклад, резидентна програма має ім'я Dump.com. Команда dump завантажує її у пам'ять, залишаючи резидентною, а команда dump off "дезактивує" програму і вивантажує її з пам'яті. Це означає, що в PSP зі зсувом 80h буде записана наступна інформація 4,'off ',13. Де:

- 4 довжина ланцюжка;
- 'off 'ланцюжок;
- 13 символ CR.

У цьому випадку програма повинна аналізувати наявність і вміст параметрів її запуску в PSP. Виявивши там слово 'off', викликати функцію вивантаження (наприклад, "свою" функцію 01h мультиплексного переривання 2Fh) .

Вивантаження можна здійснювати двома засобами:

– звільненням пам'яті, яку займає програма і блоки її оточення за допомогою функції 49h;

– використати в програмі, що вивантажує, функцію 4Ch та змусити її завершити не саму вивантажуючу програму, а резидентну програму і потім повернути керування програмі, що вивантажує.

У будь-якому випадку перед звільненням пам'яті потрібно відновити усі вектори переривань, що були перехоплені резидентною програмою. Але надійно можна вивантажити тільки останню з завантажених програм, тому що вектор, перехоплений резидентною програмою, міг бути потім перехопленим іншою резидентною програмою.

Розглянемо випадок вивантаження резидентної програми за допомогою функції 4Ch. Функція 4Ch завершує виконання програми. Але звідки система знає, яку саме програму потрібно завершити і куди передати керування? Ця інформація зберігається в PSP виконуваної програми. У слові зі зсувом 16h від початку PSP зберігається сегментна адреса PSP батьківського процесу. У двослівній чарунці з зсувом 0Ah – конкретна адреса повернення в нього. А в області поточних даних – SDA (Swappable Data Area) у слові з зсувом 10h зберігається ідентифікатор поточного процесу. Адресу SDA можна одержати за допомогою недокументованої функції 5D06h.

Функція повертає в регістри DS:SI адресу SDA, а в регістр CX – її розмір. Якщо комп'ютер не виконує ніяку програму, поточним є Command.com, і його ID записаний у SDA.

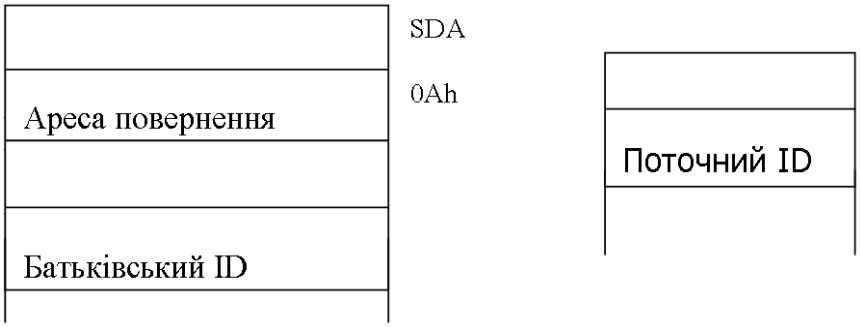

Рис. 1.12 PSP поточний процес

При запуску з клавіатури будь-якої програми (у тому числі і резидентної) система записує в SDA її ID, а в PSP запущеної програми - ID батьківського процесу й адресу повернення в нього.

Цим забезпечується можливість завершення запущеної програми і повернення в систему. Для всіх програм, запущених з клавіатури, батьківським процесом буде Command.com.

Функція 4Ch виконує дії по закриттю файлів, встановленню векторів 22h, 23h, 24h і звільненню пам'яті, яка займається поточним процесом. Ця функція переносить у SDA вміст слова PSP зі зсувом 16h, призначаючи батьківський процес поточним і передає керування за адресою переривання, записаному у PSP. Таким чином, щоб із деякої програми завершити іншу (резидентну), потрібно виконати такі дії:

- Занести в SDA ID завершальної програми, щоб оголосити її поточною і щоб 4Ch завершила саме її.

- Занести в слово із зсувом 16h у PSP програми яка завершується, ID програми, яка вивантажує, щоб після завершення резидентної програми поточною знову стала вивантажуюча програма.

- Занести в двослівну чарунку з зсувом 0Ah у PSP програми яка завершується необхідну адресу повернення в програму, що вивантажує.

- Відновити кадр стеку програми, що вивантажує, сегментні регістри і регістри загального призначення (функція 4Ch руйнує всі регістри).

Використання засобів BIOS в обробниках апаратних переривань.

BIOS є нерентабельною програмою. Але її нерентабельність викликається різними причинами. Переривання BIOS нерентабельні тільки по відношенню до самих себе. Таким чином, виклик функції BIOS можливий в обробниках тільки тоді, коли перериваюча програма не виконує функції того переривання, до якого ми хочемо звернутися в обробнику.

Для цього до складу резидентної обробки включаються секції перехоплення всіх переривань BIOS, які передбачається використовувати в обробнику. Вхідні адреси цих секцій заносяться у відповідні вектори переривань. Самі програми перехоплювачів будуються так, що частину своєї роботи вони виконують перед системними обробниками даного переривання, а частину – після. Перехоплювач переривань BIOS встановлює прапор “зайнятості” даного переривання, а після повернення з BIOS він скидається. Основна програма обробника звертається до BIOS лише в тому випадку, якщо прапор скинутий. Якщо BIOS виявляється зайнятою, обробник повинен встановити прапор незавершеності і повернути управління в перервану програму. Обробник потрібно викликати знову доти, доки він не виявить, що BIOS вільна. Тоді обробник повинен скинути прапор незавершеності, виконати потрібну функцію BIOS і повернути управління в перервану програму. Для таких повторних викликів зазвичай використовуються переривання від таймера.

Використання засобів системи в обробниках апаратних переривань

Щоб розібратися в причинах нерентабельності системи і в методах подолання цього недоліку, розглянемо частину внутрішньої організації системи. Серед системних областей є структура, яка називається областю поточних даних (SDA – Swappable Data Area). Це область об’ємом 2К, адресу якої можна одержати за допомогою недокументованої функції 5D0h.

Функція повертає в DS:SI адресу SDA, в CX-її розмір. Розглянемо поля SDA, які мають відношення до розробки резидентних програм. Прапор критичної помилки ErrorMode встановлює в 1, якщо зафіксовано стан критичної помилки,

потім викликається Int 24h. Цей прапор знаходиться в SDA зі зсувом 00h. Його довжина 1 байт. Інший прапор – прапор зайнятості системи – зсув 01h, довжина 1 байт, наз. InDOS. Він встановлюється диспетчером системи відразу після аналізу номеру функції системи що викликала її і скидається перед поверненням в прикладну програму.

Функції системи в прикладній програмі можна викликати, коли прапор InDOS скинутий. Нерентабельність системаи пов'язана з тим, що при виконанні більшості функцій використовується один і той самий системний стек. Якщо виконання будь-якої системної функції системи буде перервано апаратним перериванням, і в обробнику переривань буде викликана функція з тієї ж групи (тобто та, що використовує той самий стек), то в SS:SP будуть завантажені ті ж самі значення, що і при першому виклику. В результаті вкладений виклик буде затирати ті дані, які були в стеку. При виклику функції з іншої групи нічого не станеться. Наявність кадрів в SDA не тільки минулого, але й позаминулого стеку, дозволяє правильно обробляти один вкладений виклик.

Фактичну адресу прапору InDOS можна одержати за допомогою функції системи 34h. Вона повертає двослівну адресу прапору InDOS в регістрах ES:BX. Одержавши і зберігши цю адресу на етапі ініціалізації обробника переривань, в самому обробнику перед виконанням будь-якої функції системи потрібно виконати перевірку прапору зайнятості.

Але, коли можливе виникнення критичної помилки в обробнику апаратного переривання, перед викликом будь-якої функції системи, потрібно перевірити ще й стан прапору ErrorMode. Якщо хоча б один з них не дорівнює 0, викликати систему не можна. Визначення адреси прапору критичної помилки потрібно виконати на етапі ініціалізації обробника.

Така методика часто непридатна. Наприклад, якщо поточна програма чекає введення з клавіатури, вона не виходить з відповідної функції і прапор InDOS неперервно встановлений. В такій ситуації обробник ніколи не зможе викликати потрібну функцію.

В системі включене спеціальне переривання int 28h, яке викликається функціями введення з клавіатури. Прикладна програма може помістити в вектор 28h адресу своєї процедури обробки цього переривання. Оскільки переривання 28h

виникає тільки при виконанні функцій, що використовують стек введення/виведення, в обробнику переривань доступний виклик функцій 0dh-6Ch. Таким чином, прикладний обробник апаратних переривань, в якому викликаються функції, повинен мати:

- активізатор по перериванням від таймера (вектор 08h);
- активізатор по перериванню 28h;

Приклад обробки переривань 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 потребує запуску?
jne bx,cs:in_dos        ; ні, завершити обробник InDOS
les bx,cs:in_dos        ; так, отримаємо адресу прап. InDOS
lds si,cs:crit_err      ;  адресу прапору ErrorMode
cmp ds:[si],0           ;  прапор ErrorMode скинутий?
jne out_28h             ; ні, чекаємо
cmp es:[bx],1           ; прапор InDOS не більше 1?
ja out_28h              ; більше, вкладений виклик.
dec cs:task_req         ; скид прапору вимоги запуска
call tast               ;
out_28h:  jmp cs:old_28h        ;  закінчуємо обробник
```

Прикладний обробник переривань 28h перш за все перевіряє, чи встановлений прапор запиту запуску задачі. Якщо цей прапор скинутий, повертаємося в перервану функцію системи командою jmp cs:old_28h – передачі управління на цей обробник, адреса якого була вилучена з вектора 28h.

Якщо прапор запиту запуску задачі встановлений, спочатку перевіряємо стан прапору критичної помилки і якщо він скинутий, то перевіряємо стан прапору зайнятості системи. Цей прапор повинен дорівнювати 1, це значить, що вже викликана функція системи і більше викликати систему не можна. Після всіх

перевірок викликається процедура, яка виконує виклики системи. Потім управління передається попередньому обробнику int 28h.

Виклик int 28h виконується функцією системи на стеку в/в. Тому в прикладному обробнику переривання 28h можна викликати тільки функції дискової групи. Крім цього, недопустимий виклик файлових функцій з вказанням стандартних дескрипторів клавіатури та екрану (0-2). Щоб виконати в\в через термінал можна відкрити термінал як файл і одержати від системи дескриптор використовувати для операцій в\в. Інший спосіб - викликати функції BIOS.

Асинхронна активізація резидентних програм командами з клавіатури.

Для управління резидентною програмою командами, які подаються з клавіатури, до її складу потрібно включати обробник переривань від клавіатури (09h). Виконувати резидентну програму, знаходячись "всередині" такого обробника, можна тільки тоді, коли в програмі не використовуються функції BIOS. Таким чином, в обробнику переривань від клавіатури лише виконується аналіз введеної команди і встановлюється прапор запиту запуску задачі, якщо команда відповідає запиту на активізацію. Сама ж активізація задачі здійснюється в обробнику Int 28h.

Робота з файлами в резидентному обробнику апаратних переривань.

Робота з файлами в резидентному обробнику апаратних переривань ускладнюється тим, що при переході в обробник поточною лишається програма яка була перервана. Дескриптори файлів які відкриваються створюються системою в JFT її PSP. Для того, щоб дескриптори файлів, відкриваємих в обробнику, повністю належали йому, необхідно при вході в обробник оголосити його поточною програмою для системи. Для маніпуляції з ідентифікаторами програм в системі існують функції 50h (встановити PSP) і 51h (одержати PSP). Обидві функції не використовують системні стеки (вони є рентабельними) і їх можна використовувати в обробниках апаратних переривань.

При виклику в обробнику функцій 4Eh і 4Fh (пошук файлів), котрі використовують область DTA, перемикання ID (ідентифікаторів) програм

недостатньо. При вході в обробник треба зберегти адресу поточної DTA і встановити в якості поточної DTA обробника, а при виході – відновити DTA перерваної програми.

Для цього використовуються функції 2Fh і 1Ah. Вони нерентабельні, тому перед їх викликом в резидентному обробнику переривань треба використовувати функцію 59h, котра отримує з SDA розширену інформацію про помилку.

Перед викликом функції системи в обробнику треба отримати з SDA розширену інформацію про помилку, а перед виходом з обробника відновити її за допомогою функції 5D підфункції 0Ah.

Ця функція дозволяє записати в поле SDA розширену інформацію про помилку, отриману в полі функції 59h.

В SDA надходять значення, відповідні регістрам ax, bx, di, es. Ці дані записуються в поля SDA зі зсувом відповідно 04h, 06h, 08h, 0Ah.

Свопінг області поточних даних.

Найбільш радикальними методами надання системі властивостей рентабельності є зберігання всієї SDA в прикладному обробнику перед викликом будь-яких системних функцій та відновлення її після завершення роботи з системою. Така процедура називається свопінгом.

Перед виконанням свопінгу доцільно перевірити стан прапору InDOS, так якщо прапор скинутий, в свопінгу немає необхідності.