ЛЕКЦІЯ №5. БАТЬКІВСЬКІ ТА ДОЧІРНІ ПРОЦЕСИ

Організація дочірніх процесів.

Функція 4Bh. Завантаження і виконання однією програмою іншої програми. Ця функція надає такі можливості:

1. Програма може завантажувати і виконувати іншу програму і потім знов одержати керування;

2. Програма може виконати внутрішню команду DOS;

3. Може завантажити програму без побудови для неї PSP і не передаючи їй керування. Ця можливість дозволяє реалізувати програми з оверлейною структурою.

В усіх трьох випадках вхідні і вихідні регістри для функції завантажуються однаково.

Вхідні дані:

- AH – 4Bh;

- DS:DX файлова специфікація;

- ES:BX вказівник до області з параметрами завантаження;

- AL номер підфункції (00–завантаження і виконання програми; 03–завантаження оверлейної програми).

Вихідні дані:

- AX код помилки при CF = 1; при CF = 0 - не має значення.

Можливі коди помилок:

- 1 немає такої функції;

- 2 файл не виявлено;

- 3 немає такого шляху;

- 5 відмовлено в доступі;

- 8 не вистачає пам’яті;

- 10 неприпустиме оточення;

- 11 неприпустимий формат.

Завантаження і виконання програми. Регістр AL встановлений в 0. Пара ES:BX містить адресу блоку параметрів.

Структура блоку параметрів:

- Поле 1 (одне слово). Сегментна адреса нового оточення;

- Поле 2 (два слова). Покажчик до командного рядку, який потрібно записати в PSP + 80h;

- Поле 3 (два слова). Покажчик до FCB, який потрібно записати в PSP + 5Ch;

- Поле 4 (два слова). Покажчик до FCB, який потрібно записати в PSP + 6Ch.

У цілому програма, вказана за допомогою DS:DX виконується так само, як і при самостійному виконанні. Є і деякі розходження :

1. Перед тим, як виконати EXEC, батьківська програма повинна забезпечити вільний блок достатнього розміру для виконуваної програми. У загальному випадку вся робоча пам’ять після резидентної частини COMMAND описана в одному блоці, який привласнений батьківській програмі. Для того, щоб звільнити місце, вона повинна зменшити цей блок за допомогою функції 4Ah для зміни блоку пам’яті. Потім можна виконати функцію 48h з BX=FFFFh, з одержанням при цьому в BX розміру блоку пам’яті, що звільнився .

2. При виконанні EXEC DOC не здійснює збереження і відновлення регістрів – це повинна робити батьківська програма. Для цього вона може використовувати свій стек, але так як SS і SP теж змінюються, їх потрібно зберегти в змінних і відновити відразу після того, як програма знов одержить керування.

3. Відкриті за методом ФМ файли успадковуються програмою яка виконується за допомогою EXEC, якщо тільки вони не відкриті з режимом успадкування 1. (функція 3Dh). Це означає, що програма яка виконується може використовувати файлові маніпулятори батьківської програми. Вказівник позиції в кожному відкритому файлі (FP) зберігається таким, яким він був у батьківській програмі. Для того, щоб використовувати файлові маніпулятори, вони повинні бути відомі програмі що виконується. Можна, наприклад, передати їхні номери як параметри в командному рядку(зазначеному в блоці параметрів).

4. Коли програма виконується командним процесором COMMAND, вона може знайти параметри командного рядку в PSP. Параметри може передавати і батьківська програма. З цією метою поле 2 блоку параметрів заповнюється адресою файлової специфікації, що містить параметри. Є й додаткова особливість. При виконанні командним процесором COMMAND параметри командного рядка

розташовуються зі зсувом 81h у PSP і обмежені символом CR(0Dh). Загальна довжина ланцюжку параметрів записана зі зсувом 80h. При виконанні з іншої програми ланцюжок із параметрами записується в PSP виконуваної програми зі зсувом 80h без поля довжини перед ним і без символу CR наприкінці. Це може призвести до помилок, якщо програма виконується самостійно. Тому ланцюжок параметрів повинен починатися байтом довжини ланцюжку і завершуватися символом CR. Якщо немає параметрів, ланцюжок параметрів повинен містити тільки символи 00h і 0Dh. Все це має сенс тільки для програм, що використовують параметри командного рядку.

5. Батьківська програма може передавати програмі що виконується копію свого оточення або зовсім нове оточення, яке вона побудувала у своїй робочій області. Для того, щоб передати копію свого оточення, вона записує 0 у поле 1 блоку параметрів. Для передачі нового оточення вона повинна записати в поле його сегментну адресу. Нове оточення повинно бути дійсним, тобто містити ASCIIZ ланцюжки у форматі ім'я = змінна і завершуватися нульовим байтом. Наприклад, в оточенні, як і для програм що виконуються командним процесором COMMAND, розташовується файлова специфікація програми (тобто ASIIZ-ланцюжок, чия адреса знаходиться в DS:DX).

6. Коли програма виконується командним процесором COMMAND, система використовує перші два параметри програмного рядку як імена файлів і конструює на їхній основі два невідкритих FCB у PSP (зі зсувом 5Ch і 6CH). Коли програма виконується іншою програмою, у поле 3 і 4 блоку параметрів можна записати адреси двох FCB, що будуть побудовані у PSP при виконанні EXEC. Якщо не будуть передаватися файли таким методом, поля 3 і 4 повинні містити 0.

7. Адреса завершення в PSP виконуваної програми (зі зсувом 0Ah) вказує адресу після інструкції INT 21h, за допомогою якої виконується EXEC .

Виконання внутрішньої команди

Це - конкретний випадок, коли одна програма виконує іншу. Користувацька програма викликає за допомогою функції EXEC командний інтерпретатор Command.com і передає йому як параметр внутрішню команду. Таким чином, у

пам`ять завантажується друга копія Command (вторинний командний процесор), що виконує команду, після чого звільняє пам`ять і повертає керування користувацькій програмі. Можна завантажити і вторинний командний інтерпретатор, що залишається активним до виконання команди EXIT. У цьому випадку командний процесор, отримавши керування, очікує введення команди системи з клавіатури.

Процедура для виконання внутрішньої команди складається з трьох кроків:

1. Забезпечується вільний блок достатнього розміру, в який буде завантажений Command.com. Для цього зазвичай використовується функція 4Ah, за допомогою якої зменшується блок поточної програми.

2. Будується ASCII-ланцюжок із параметрів для Command такої структури:

- 1 байт довжина значущої частини ланцюжку;

- XX байт команда для COMMAND (формат \C);

- 1 байт 0Dh (CR – символ кінця рядку). Приклад - db 12 ,/C Dir *.ASM,13.

3. Будується блок параметрів довжиною 7 слів. Адреса ланцюжку записується в поле 2 блоку параметрів. В інших полях блоку записується 0.

4. У DS:DX записується адреса ASCIIZ-ланцюжку, що містить файлову специфікацію командного інтерпретатора. У ES:BX записується адреса блоку параметрів. У AX записується 4B00h. Після цього виконується переривання INT 21h.

Завантаження програми як оверлейного модулю.

Регістр AL має значення 3. Пара ES:BX містить адресу блоку параметрів з такою структурою:

Поле 1. Довжина першого слова адреси сегменту для завантаження програми;

Поле 2. Довжина першого слова – фактор зсуву для завантажуваної програми (тільки для EXE файлів).

Програма (оверлейний модуль) завантажується без побудови PSP для неї і без передачі керування. Звернення до неї здійснюється як до далекої процедури, а вона повертає керування інструкцією RET. Програма не породжує новий процес.

Перед виконанням EXEC потрібно забезпечити достатньо пам’яті для завантаження програми. Це може бути робоча область у блоці головної програми або окремий блок пам’яті, заявлений головною програмою за допомогою функції 48h. Перед цим потрібно виконати функцію 4Ah, щоб зменшити блок головної

програми. У першому випадку головна програма повинна обчислити сегментну адресу області для оверлейного модуля, а в другому вона одержує його при виконанні функції 48h. Цю адресу потрібно записати в поле першого блоку. Система налаштовує оверлейний модуль до адреси завантаження відповідно до чиннику зсуву в полі 2 блоку параметрів.

У загальному випадку фактор зсуву повинен дорівнювати сегментній адресі завантаження модуля, тобто поле 2 повинно дорівнювати полю 1.

Нестандартні прийоми передачі параметрів до дочірньої програми.

Всі викладені прийоми передачі параметрів у дочірню програму зручні тим, що вони входять до числа стандартних засобів. Але в такий засіб можна передати лише обмежений обсяг інформації, причому переважно символьної. При розробці складних програмних комплексів виникає необхідність у більш тісній взаємодії батьківських та дочірніх процесів. Наприклад, бажана можливість передачі адрес масивів для того, щоб відкрити доступ дочірній програмі до батьківських даних. У системи таких засобів не має, самі ж ієрархічні програми не мають ніякої інформації одна про одну.

Для того, щоб передати дочірній програмі адреси своїх областей даних, батьківська програма може скористатися двома областями пам’яті, доступними усім програмам: вільними векторами переривань та областю міжзадачних зв’язків.

У таблиці векторів переривань є вільні ділянки. Зокрема вектори 60h…66h відведені для переривань користувача. Це означає, що програми DOS і BIOS не звертаються до цих векторів, тому прикладні програми можуть використовувати їх за своїм розсудом. Але для того, щоб дочірні та батьківські програми могли обмінюватися інформацією через вільні вектори, для них повинен бути обговорений міжпрограмний інтерфейс – обидві програми повинні знати, у яких векторах і що саме знаходиться.

Оскільки кожний вектор – це 4-х байтова чарунка, в неї можна записати повну адресу будь-якого даного і тим самим забезпечити доступ дочірньої програми до полів даних батьківської програми. У батьківській програмі для реалізації описаного

прийому потрібно використовувати функцію 25h (заповнення вектору переривання), а в дочірній – функцію 35h (одержання вектору переривання).

Крім вільних векторів для передачі інформації між програмами можна користуватися областю міжзадачних зв'язків. Вона розташовується наприкінці області даних BIOS за адресами 40h:F0h – 40h:FFh (усього чотири подвійних слова). Ця область не використовується системою й до неї можна записувати свої дані.

Обробка переривань в реальному режимі

Обробка переривань. Взаємодія прикладних і системних обробників переривань.

Структура програм обробки переривань та їх взаємодія з іншими програмами визначається низкою чинників:

- переривання, яке ініціалізує обробник, може бути апаратним (від периферійних пристроїв) і програмним (команда INT);
- Програма може бути резидентною або транзитною;
- Вектор оброблюваного переривання може бути вільним або використовуватися системою;
- Якщо вектор уже використовується системою, то новий обробник може цілком замінювати його або зчеплятися з ним;
- У випадку зчеплення з системним обробником новий обробник може виконувати свої функції до системного або після нього.

Щоб прикладний обробник одержував керування в результаті переривання, його адресу потрібно помістити у відповідний вектор переривання.

Для цього призначена функція 25h переривання INT 21h. У регістр AL поміщається номер вектора, що модифікується, а в DS:DX – адреса нового обробника. Старий вміст вектора потрібно зберегти у виділених для цього чарунках і відновити перед завершенням програми (за допомогою цієї функції). Для одержання вихідного вмісту передбачена функція 35h.

У випадку, коли потрібно внести незначні зміни або ж доповнити системний алгоритм програма користувача "зчеплюється" з системною програмою обробки

переривань. При ініціалізації обробника, що “зчеплюється” з системним, треба зберегти адресу системного обробника і помістити у вектор переривання адресу прикладного обробника.

Якщо прикладна обробка повинна виконуватися після системної, то в стеку перерваного процесу опиняється три слова: слово прапорів і двослівна адреса повернення в перервану програму.

Така структура даних повинна бути в верхівці стеку, щоб команда IRET могла повернути керування до перерваного процесу:

Формат стеку:

- IP2 відносна адреса точки повернення в прикладний обробник;
- CS2 сегментна адреса прикладного обробника;

Прапори:

- IP1 відносна адреса точки повернення до перерваного процесу;
- CS1 сегментна адреса перерваного процесу.

Системний обробник, закінчивши обробку переривання, завершується командою IRET. Ця команда забирає зі стеку три верхніх слова і здійснює перехід за адресою CS1:IP1.

Якщо прикладна обробка виконується до системної, то в ній завершальна команда - це команда переходу - передачі керування до системного обробника. Ця команда не торкається стеку.

Обробник переривань від таймера.

Для того, щоб прикладні програми могли використовувати сигнали таймера, не порушуючи при цьому роботу системного годинника, до програми BIOS, яка обслуговує апаратні переривання від таймера, включений виклик 1Ch. Переривання 1Ch передає керування на програму-заглушку BIOS, яка містить єдину команду iret. Користувач може записати в вектор 1Ch адресу прикладного обробника сигналів таймера та використовувати в своїй програмі засоби реального часу. Перед завершенням програми треба відновити старе значення вектору 1Ch. Текст обробника можна розмістити в будь-якому місці програми, забезпечивши неможливість випадкового переходу на його рядки не в результаті переривання, а по

ходу виконання основної програми. Обробник переривання може бути процедурою, а може починатися просто з помітки. Важливо тільки, щоб останньою командою обробника була команда iret.

Алгоритм обробника переривання 1Ch.

Приклад виведення в кут екрану символу з періодичністю 4 рази на 1 сек., змінюючи кожного разу його атрибут.

1. Зберегти системний вектор (функція 35h повертає його вміст у регістрах ES:BX).

2. Записати у вектор 1Ch адресу нового обробника переривання (25h). Для цього потрібно налаштувати регістр DS на сегмент, в якому міститься процедура обробника або на сегмент команд. (DS зберегти у стеку і через стек записати в нього вміст регістру CS).

3. Починаючи з цього моменту, переривання від таймера будуть активізувати 18,2 рази на секунду програму нового обробника int 1Ch. Для затримки всієї програми в пам'яті вона повинна виконувати якісь дії, крім виклику процедури обробника. Наприклад, виводити в циклі на екран текст.

4. Виконати процедуру обробника переривання 1Ch:

4.1 Зберегти в стеку регістри, які будуть використовуватись в обробнику. Це важливо, тому що перехід на програму обробника здійснюється по команді int 1Ch з системної програми обробки переривань від таймера. При виконанні процедури переривання процесор налаштовує належним чином тільки регістри CS та IP. Вміст всіх інших регістрів відображує стан системної програми. Якщо цей стан буде змінений, то після повернення з нашого обробника в системну програму, вона перестане функціонувати.

4.2 Регістр ES налаштувати на адресу відеобуферу. В AX – код ASCII символу, який виводиться на екран. Треба зробити заміну регістру DS на CS. DS вказує не на дані, а на сегмент програми BIOS. В цьому випадку і дані треба розміщувати в сегменті коду.

4.3 Виводити символ в одне й теж саме місце екрану з періодичною зміною атрибутів.

4.4 Відновити збережені в стеку регістри, завершити програму обробника командою iret.

Обробка переривань по <Ctrl>/C <Ctrl>/<BREAK>

У багатьох обчислювальних системах комбінація клавіш Ctrl/C зарезервована для примусового завершення активної програми і передачі керування системі. Але для цього потрібно, щоб система постійно аналізувала коди, що надходять при обробці переривання від клавіатури. Та система перевіряє наявність Ctrl-C у вхідному потоці на більш високому рівні при виконанні програмних запитів. Різні функції системи по-різному реагують на введення з клавіатури Ctrl– C.

Всі функції системи діляться на дві групи функцій: введення/виведення за номерами 01h–0Ch та інші. Функції з номерами, що перевищують 6Ch використовують розширювачі, мережні програми та інші навколо системні програми.

Розбіжність цих двох груп функцій полягає в тому, що при виклику функцій в/в системи переходить на внутрішній стек введення/виведення, а при виклику всіх інших функцій - на дисковий стек. Наявність у системаи двох внутрішніх стеків забезпечує її часткову рентабельність. Тобто, при помилці в процесі виконання “дискової“ функції системи можна викликати функцію в/в. Велика частина функції в/в перевіряє наявність у буфері клавіатури коду 03h (Ctrl/C). При виявленні цього коду виконується команда INT 23h (завершення поточного процесу). Виняток складають функції 06h і 07h, нечутливі до Ctrl/C. Функції 02h і 09h аналізують буфер клавіатури на наявність Ctrl/C один раз на 64 виклики.

“Дискові” функції перевіряють натискання Ctrl/C тільки коли встановлений прапор Break, тобто коли була виконана команда Break ON.

Натисканням Ctrl/C не можна завершити чисто обчислювальну задачу, а тільки таку, у якій викликаються системні функції.

При завершенні задачі за допомогою Ctrl/C можуть залишатися не встановленими змінені вектори переривання або неправильно закритися відкриті файли. Тому більшість прикладних програм не використовують системний обробник Ctrl/C, а замінюють його своїм (вектор 23h). Після завершення задачі бажано відновити початковий вміст вектора, тому що можливе порушення роботи системи при виконанні інших задач.

В обробнику вектора 23h можна використовувати будь-які функції системи. При завершенні обробника командою IRET керування повернеться в програму в тій самій точці, де вона була перервана. Але в програмі обробника можна передбачити перехід у будь-яке місце програми без виконання IRET. Це дозволяє організувати натисканням Ctrl/C зміни ходу виконання програми, зокрема її коректне завершення.

Є ще одна можливість втручання в хід виконання програми – натискання клавіш <Ctrl–Break>.

Системний обробник переривань від клавіатури, що входить до складу BIOS, при виявленні комбінації клавіш Ctrl–Break передасть керування програмі, адреса якої знаходиться у векторі 1Bh. Ця програма також входить до складу BIOS і складається з єдиної команди IRET. Але в процесі початкового завантаження ОС, система змінює вміст вектора 1Bh, записуючи в нього адресу свого обробника. Цей обробник виконує такі дії:

1. Включає 0000h у буфер клавіатури на місце головного символу;

2. Модифікує покажчики буфера так, що вони здаються системі очищеними;

3. Записує прапор Ctrl/Break в чарунку області даних BIOS за адресою 40h:71h;

4. Записує код Ctrl/C (03h) у буфер драйвера консолі CON, так що драйвер вважає, що виконана комбінація <Ctrl/C >.

У результаті система передає керування на вектор 23h, начебто була натиснута комбінація клавіш Ctrl/C. Отже, системна обробка Ctrl–C і Ctrl-Break відбуваються майже однаково.

Особливість обробки Ctrl/C полягає у тому, що якщо перед натисканням Ctrl/C були натиснуті які-небудь клавіші і їхні коди при цьому залишилися в кільцевому буфері клавіатури, вони будуть маскувати Ctrl/C.

При обробці Ctrl/Break програма системи, яка обробляє переривання 1Bh, відправляє код не в кільцевий буфер клавіатури, а безпосередньо в драйвер CON.

У такий спосіб комбінацію Ctrl/Break не можна замаскувати і крім того, програма обробки Ctrl/Break очищує кільцевий буфер клавіатури.

Прикладна програма може замінити вміст вектора 1Bh адресою власного обробника. У цьому випадку при натисканні Ctrl/Break відбувається негайний

перехід на програму обробника, який може взяти на себе керування в будь-якій точці програми в тому числі при зациклюванні.

Початковий зміст вектора 1Bh не відновлюється системою автоматично при завершенні програми. Тому в прикладній програмі, яка перехоплює вектор 1Bh варто передбачити перед її завершенням відновлення початкового змісту вектора.

Але програма може завершитися аварійно за допомогою Ctrl/C. Це означає, що потрібно передбачити власний обробник переривань по Ctrl/C, в якому перед завершенням програми відновлюється 1Bh .

Користувацька процедура для обробки переривання 24h.

Стандартну обробку критичних помилок можна замінити користувацькою процедурою шляхом зміни вектора 24h. Будь-яка така процедура повинна відповідати таким вимогам:

1. Всі регістри повинні зберігати свої значення.

2. Не можна виконувати системні функції за винятком функцій від 01h до 12h та 59h.

3. Не можна змінювати блок заголовку драйвера.

4. Процедура повинна закінчуватися інструкцією IRET, за допомогою якої керування повертається в систему. При цьому AL повинен містити необхідні вказівки (0 – ignore, 2 – abort, 3 – fial). Керування можна повернути і безпосередньо користувацькій програмі, відразу після INT 24h. Для цього необхідно прибрати останні три слова з вершини стеку (адреса повернення в систему), відновити регістри користувача зі стеку, після чого виконати інструкцію IRET.

У деяких випадках зручно побудувати користувацьку процедуру для обробки переривання 24h як надбудову стандартної:

а) Перед тим, як поставити у вектор 24h адресу своєї процедури обробки критичних помилок користувацька програма зберігає його вміст.

б) Отримавши керування, користувацька процедура виконує інструкцію PUSHF і потім CALL FAR з адресою зберігання старого вміста вектора 24h у якості операнду. У такий засіб керування передається стандартній процедурі. Вона виводить необхідні повідомлення і чекає відповіді користувача.

в) Відповідь кодується в AL і керування повертається користувацькій програмі.

г) Користувацька процедура виконує необхідні дії і повертає керування.

Старий вміст 24h зберігається в PSP користувацької програми (зсув 12h) і відновлюється автоматично по закінченню програми.

Зміною вектора 24h активізується користувацька процедура для обробки критичних помилок.

1. При одержанні керування процедура зберігає в полях користувацької програми повну інформацію про помилку (регістри AH,DI,BP:SI). Можна виконати і функцію 59h, щоб одержати більш повну інформацію про помилку. Ця інформація теж зберігається для використовування пізніше. Після цього процедура записує 3 у AL (fail) і повертає керування за допомогою IRET. Якщо відповідь fail не припускається, то система перериває виконання програми перериванням 23h. Стандартну обробку переривання 23h можна замінити користувацькою програмою. Якщо відповідь fail припускається, система припиняє поточну системну функцію і повертає керування користувацькій програмі з кодом помилки 83.

2. Будь-яке звернення до системної функції обробляється в залежності від виду і результату системної функції. Якщо функція виконується в якості індикації помилки AL=FFh або CF=1 і дійсно завершилася помилкою, виконується функція 59h.

3. Якщо код помилки дорівнює 83, то це критична помилка і вона обробляється в залежності від інформації, збереженої на кроці 1. Якщо код відмінний від 83, то використовується й інша інформація про помилку, повернута функцією 59h (тип, місце і можливі дії). Потрібно мати на увазі, що процес може передавати код завершення процесу, що його виконав. Щоб використовувати цю можливість необхідно помістити бажаний код завершення в AL і виконати функцію 4Ch переривання 21h для завершення програми. Коли керування буде повернуто батьківському процесу, то він виконає функцію 4Dh переривання 21h (без вхідних регістрів) і в AL буде отриманий код завершення, який потім може бути проаналізований. Крім того, AH міститиме інформацію про те, як завершився викликаємий процес:

- 0 для нормального завершення;

- 1 по Ctrl/Break;

- 2 по критичній помилці пристрою;

- 3 за допомогою функції 31h, що залишає програму резидентною.

Якщо програма завершилась за допомогою цієї функції, то система одержує код виходу і він може бути ввімкнений в обробник командним файлом за допомогою підкоманди IF.

Код виходу розглядається як номер ERRORLEVEL. За допомогою цієї можливості командні файли можуть припиняти обробку і виводити повідомлення про виникнення помилки в одній із запущених програм.