ЛЕКЦІЯ №4. СИСТЕМНІ ТАБЛИЦІ ТА ОБЛАСТІ ДАНИХ, МЕТОДИ РОБОТИ З ФАЙЛАМИ.

Методи роботи з файлами. Метод дескрипторів, DM.

Метод дескрипторів припускає використання дескрипторів (файлових індексів, описувачів, маніпуляторів) які можемо розглядати як номери відкритих файлів.

Процедура звертання до файлу в загальному випадку розпадається на наступні операції:

1. Створення файлу з заданим ім'ям в зазначеному каталозі або відкриття файлу, якщо був створений раніше.
2. Запис до файлу або читання з файлу всього збереженого або будь-якої його частини.
3. Закриття файлу.

Відкриваючи файл, призначає йому вільний елемент (блок опису файлу). Цей елемент міститься в спеціальній системній таблиці – таблиці відкритих файлів – SFT (SYSTEM FILE TABLE), яка розташовується в оперативній пам'яті серед системних областей даних. Обсяг цієї таблиці визначається на етапі конфігурування директивою Files в Config .sys.

Знайшовши в системі каталогів диску запис про відкриваємий файл, система записує у виділеній йому SFT основні характеристики файлу: ім'я, довжина, атрибути, дату та час створення, стартовий кластер та інші. Частина інформації переписується в елемент SFT з запису каталогу, частина (вказівник на BPB) ОС поставляє сама.

В SFT є двослівна комірка, в якій зберігається вказівник файлу - № байту відносно початку файлу, з якого починається чергова операція читання чи запису. Це дозволяє організувати прямий доступ до файлу. Посилка на № блоку опису файлу в SFT повертається до програми у вигляді дескриптору. Звертання до відкритого файлу виконується по привласненому йому дескриптору. Невідкритий файл дескрипторів не має. Зміст в SFT модифікується під час роботи з файлом, так би мовити завжди відображує поточний стан файлу. Після закінчення роботи з файлом його потрібно закрити. В процесі закриття виконується скид буферів на

диск, модифікація запису каталогу та звільнення блоку опису файлу в SFT разом із закріпленим за ним дескриптором. Таким чином, система може працювати з необмеженим числом файлів, але кількість одночасно відкритих файлів визначається об’ємом системної таблиці файлів.

Для роботи зі стандартними пристроями система надає 5 визначених дескрипторів:

- 0 стандартне введення (CON);
- 1 стандартне виведення (CON);
- 2 стандартна помилка (CON);
- 3 стандартний допоміжний порт (AUX);
- 4 стандартний принтер (PRN).

Різниця між дескриптором 1 та 2 полягає в тому що стандартне виведення (1) можна перенаправити, а (2) завжди виводиться на дисплей.

Система використовує в своїй роботі низку системних таблиць, які в документації відображені лише частково.

Основною таблицею є список списків (List of lists).

Адресу списку списків можна отримати за допомогою функції 52h переривання 21h. Функція повертає двослівну адресу списку списків в регістрах ES:BX.

Табл. 1.13 Вміст деяких полів списку списків

Зсув	Розмір	Опис
-06h	4	Вказівник на поточний диск буферу
-02h	2	Сегментна адреса першого блоку MCB
+00h	4	Вказівник на перший блок параметру диску
+04h	4	Вказівник на першу системну таблицю файлів (SFT)
+08h	4	Вказівник на заголовок активного пристрою CLOCK$
+0Ch	4	Вказівник на заголовок активного пристрою CON
+10h	2	Максимальний розмір сектора в байтах у блочному пристрої
+12h	4	Вказівник на інформаційний запис дискового буферу
+16h	4	Вказівник на масив структур поточних каталогів
+20h	1	Число встановлених блочних пристроїв
+21h	1	Число елементів в масиві структур поточних каталогів
+22h	18	Заголовок драйвера пристрою
+3Fh	2	Значення x в BUFFERS=x,y
+41h	2	Значення y в BUFFERS=x,y
+43h	1	Дисковод завантаження (1=A:)
+44h	2	Розмір розширеної пам’яті в кілобайтах

В байтах 4-7 списку списків зберігається адреса SFT. За замовчуванням при завантаженні ОС формується одна SFT, до якої входить 5 блоків опису файлів.

При включенні до файлу Config.sys директиви FILES створюється нова таблиця, пов'язана з першою. Загальна кількість блоків в двох таблицях дорівнює параметру директиви FILES.

Табл. 1.14 Блок опису файлу системи

Зсув	Розмір	Опис
00h	2	Кількість дескрипторів, закріплених за даним файлом або 0, якщо блок вільний
02h	2	Режим доступу до файлу, заданий при його відкритті
04h	1	Атрибути файлу
05h	2	Інформаційне слово пристрою
07h	4	Вказівник на заголовок драйверу символьного пристрою або вказівник на блок параметрів диску
0Bh	2	Номер першого кластеру файлу
0Fh	2	Дата останньої модифікації файлу
0Eh	4	Розмір файлу в байтах
15h	4	Поточне положення вказівника файлу
19h	2	Відносний номер останнього кластеру файлу прочитаного або записаного
1Bh	4	Номер сектору з записом каталогу про даний файл
1Fh	1	Номер запису каталогу всередині сектору
20h	11	Ім'я та розширення файлу (без шляху)
2Eh	2	Сегментна адреса PSP програми, яка відкрила файл
35h	2	Абсолютний номер останнього записаного або прочитаного кластеру

Блок параметрів диску (Drive Parameter Block, DPB) містить інформацію, частково повторюючи дані в секторі завантаження. Адреса цього блоку розміщується в блоці опису файлу.

Кожна SFT починається з 6-ти байтового заголовку:

1. Зсув - 0, Розмір - 4, Опис - Адреса наступної таблиці або FFFFh в першому слові, якщо ця таблиця остання.

2. Зсув - 4, Розмір - 2, Опис - Кількість блоків опису файлів в даній таблиці.

Далі йдуть блоки опису файлів.

Табл. 1.15 Блок параметрів диску DPB

Зсув	Розмір	Опис
00h	1	Номер диску 00H-A,01H-B …
01h	1	№ пристрою всередині драйверу пристрою
02h	2	Розмір сектору в байтах
04h	1	Найбільший номер сектору в кластері (розмір кластеру -1)
05h	1	Число зсувів, щоб перетворити кластери на сектори
06h	2	Число зарезервованих секторів з початку диску
08h	1	Число FAT
09h	2	Число елементів в кореневому каталозі
0Bh	2	Номер сектору, з якого починаються дані
0Dh	2	Найбільший номер кластеру (число кластеру даних+1)
0Fh	2	Число секторів в одній FAT
11h	2	Номер першого сектору каталогів
13h	4	Адреса заголовку драйверу пристрою
18h	1	00, якщо до диску було звертання, FF в іншому випадку
19h	4	Вказівник на наступний DPB
1Dh	2	Номер кластеру, з якого потрібно починати пошук вільного простору на диску
1Fh	2	Число вільних кластерів на диску, FFFFh-якщо невідомо

Для отримання адреси блоку параметрів диску можемо користуватись функцією 32h. Вона має наступний формат виклику :

Вхід: AH-32h

DL-номер дисководу ,0- поточний .

Вихід: AL-0, якщо був заданий вірно номер дисководу, 0FFh- невірно .

DS: BX-адреса блоку параметрів диску.

Для отримання адреси блоку параметрів поточного диску можемо користуватись функцією 1Fh.

Тому що порядок опису відкритих файлів в SFT невідомий наперед, для знаходження конкретного блоку опису файлу потрібно:

1. Відкрити файл з функцією 3Dh.

2. Отримати від системи його дескриптор.

3. По дескриптору знайти відповідний йому блок опису файлу в SFT.

В префіксі програмного сегменту (PSP) є таблиця файлів завдань (Job File Table, JFT) рисунок 1.10. За замовчанням ця таблиця починається з байту 18h PSP і має розмір 20 байтів. В біти JFT по мірі відкриття файлів записує номер відповідних блоків опису файлів в SFT. Вільні байти JFT мають коди FFh.

Відносні номери байтів JFT від її початку і є дескриптори відкритих файлів. Перші 5 дескрипторів (від 0 до 4) система закріплює за стандартними пристроями CON, AUX, PRN. Цим пристроям відповідають 3 блоки опису файлів в першій SFT.

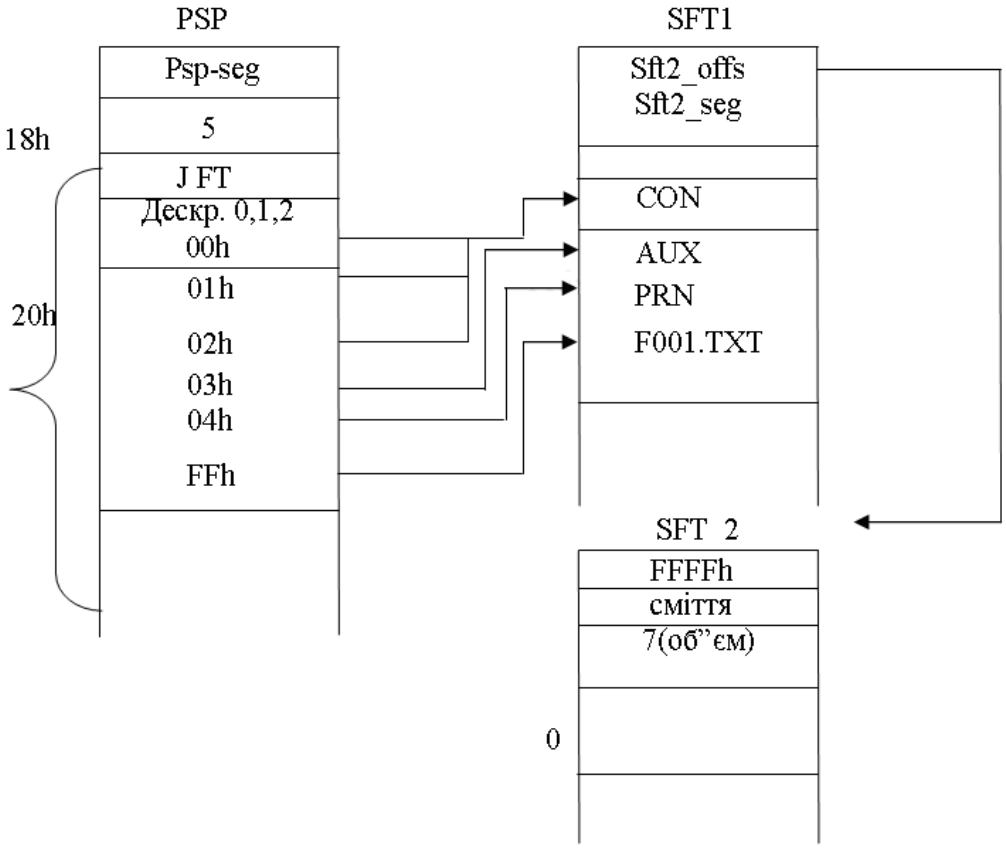

Рис. 1.10 Порядок опису відкритих файлів

Таким чином перший вільний дескриптор повинен мати значення 5, а відповідна йому інформація про файл знаходиться в блоці опису файлу з номером 3. Нехай в CONFIG.SYS FILES=12, відкриті файли F.001 та F.002.

В PSP програми крім таблиці файлів завдань є ще поля, зв'язані з цією таблицею. В слові, що має зсув 32h від початку PSP зберігається розмір JFT (за замовчанням 20=14h). У двослівній чарунці зі зсувом 36h - сегментна адреса PSP, яка при завантаженні програми надходить до регістрів DS та ES.

Оскільки розмір JFT за замовчанням складає 20 байтів при чому 5 з них зайняті конкретними значеннями, то програма не може відкрити більш ніж 15 файлів, щоб відкрити більшу кількість файлів потрібно за допомогою 67h створити нову JFT. Система, виділивши для неї пам'ять, запише до відповідних PSP нову адресу та розмір цієї таблиці, зкопіює зміст старої JFT в нову таблицю і тепер буде відкривати файли за допомогою нової JFT.

Системні функції для роботи з файлами за методом дескрипторів.

Всі функції зручно розкласти на смислові групи.

1. Створення, відкриття та закриття файлів:

- 3Ch створити файл;
- 5Ah створити тимчасовий файл;
- 5Bh створити новий файл;
- 3Dh відкрити файл;
- 3Eh закрити файл;
- 68h скинути файл на диск.

2. Запис та читання даних:

- 42h встановити вказівник;
- 40h записати в файл або в пристрій;
- 3Fh читати з файлу або з пристрою.

3. Зміна характеристик файлу:

- 43h отримати або встановити атрибути файлу;
- 56h перейменувати файл;
- 57h одержати або перевстановити дату та час створення файлу.

4. Пошук файлу:

- 1Ah встановити адресу області передачі даних (DTA);
- 2Fh одержати адресу області передачі даних;
- 4Eh знайти перший файл;
- 4Fh знайти наступний файл.

5. Операція над каталогами:

- 39h створити каталог;
- 3Ah виділити каталог;
- 3Bh змінити поточний каталог;
- 47h отримати поточний каталог.

6. Операції над дисками:

- 19h отримати поточний диск;
- 0Eh змінити поточний диск;
- 36h отримати інформацію про диск.

Функція 3Ch та 5Bh дозволяють створити файл з заданою специфікацією. Специфікація файлу вказується у вигляді символьного рядку, який завершується двійковим нулем. Адреса цього рядку заноситься до регістрів DS:DX.

В регістрі CX задається код атрибутів файлу що створюється.

В регістр AX повертається дескриптор створеного файлу, який потім можемо використовувати для читання файлу або для запису до файлу.

Розбіжність між функціями 3Ch та 5Bh проявляється в тому випадку, коли файл який створюється вже існує.

Функція 3Ch знищує файл, який вже маємо, а функція 5Bh завершується з CF=1.

Функція 5Ah використовується для створення тимчасового файлу, ім'я якому дає система. В регістрах DS:DX вказується адреса шляху до файлу (але не ім'я файлу) у вигляді символів рядку, в кінці якого мають бути передбачені 13 порожніх байтів, куди помістить "/" та ім'я файлу що створюється, яке закінчується двійковим нулем. Можемо надати будь-які атрибути, крім атрибутів позначки диску. При завершенні програми автоматично такий файл не видаляється. Турбота про це лежить на програмісті. Для запису в створений тимчасовий файл потрібно використовувати дескриптор, повертаємий функцією 5Ah в регістрі AX.

Функція 3Dh дозволяє відкрити вже існуючі файли. В регістрах DS:DX задається специфікація файлу у вигляді рядку ASCIIZ, в регістр AL - режим доступу (0-читання, 1- запис, 2- читання та запис). В подальшому запис в файл та читання з нього здійснюється за допомогою дескриптору, повертаємого функцією в регістр AX.

Для кожного відкритого файлу створюється і підтримується вказівник, який являє собою відносний номер байту в файлі, починаючи з якого буде здійснюватися запис або читання даних. Вказівник тільки що відкритого або створеного файлу встановлюється системою на початок файлу. Функції читання або запису зміщують його на число прочитаних або записаних байтів.

Для організації прямого доступу до довільного місця файлу передбачена функція 42h. Вона дозволяє задати положення вказівника відносно початку файлу (AL=0), кінця файлу (AL=2) або поточного положення вказівника (AL=1).

Саме значення положення вказівника (зі знаком) заноситься до регістру CX (старша половина) і DX (молодша).

Функція 3Fh використовується для читання з файлу або пристрою, 40h-для запису. Перед викликом функції в регістр BX поміщається дескриптор, в регістр CX-число байтів що читаються або записуються, а в регістри DS:DX - адреса буферу в програмі користувача.

Іноді виникає необхідність знайти в каталозі всі файли, які задовільняють умовам шаблону (наприклад, *. txt). Такий пошук здійснюється функціями 4Eh та 4Fh (знайти наступний файл). Для їх використання необхідно за допомогою функції 1Ah організувати в програмі область передачі даних (DTA, Disk Transfer Area), розміром не менше 64б або за допомогою функції 2Fh отримати адресу DTA. В якості такої DTA Система використовує PSP від байту 80h до кінця. Система поміщає в DTA інформацію про знайдений файл (атрибут, розмір, час, дата створення і т. д.) В байтах 1Eh…2Fh в DTA утримується ім’я та розширення файлу у вигляді рядка ASCIIZ.

При пошуку файлів при заданому шаблоні спочатку активізується функція 4Eh. В регістрах DS:DX міститься адреса рядка ASCIIZ зі шляхом розглядаємого каталогу. В регістрі CX- код комбінації повертає в CF=0, а ім’я та розширення файлу у вигляді рядка ASCIIZ поміщається в DTA. Отримавши ім’я файлу, можна відкрити його за допомогою функції 3Dh та виконати операції над ним.

Функція 4Fh використовується так само, як і 4EH.

Метод FCB для роботи з файлами.

FCB – блок керування файлом. Використовується традиційними функціями. Розширені функції використовують метод ФМ – файловий маніпулятор.

Основне в цьому методі полягає в тому, що користувацька програма повинна побудувати блок керування файлом (FCB) для кожного файлу, з яким вона працює. FCB представляє собою область пам’яті яка використовується з довжиною 37 або 44 байти, яка має стандартний формат. У FCB підтримується необхідна для роботи з файлом інформація: пристрій, ім’я, розмір, дата, розмір запису, поточна позиція у файлі і т. д.

Відзначимо, що не передбачене місце шляху до файлу – доступні тільки файли поточного каталогу.

Частина інформації (пристрій, ім'я) заповнюється програмою ще до відкриття файлу. При відкритті файлу заповнює інші поля FCB (розмір, дата і т. д.). Користувацька програма задає параметри для системних функцій у полях FCB, а вони зі свого боку повертають інформацію через FCB. У цьому процесі часто припускаються помилки, а створення і підтримка FCB для кожного файлу потребує додаткових зусиль. Одна з типових помилок – використання одного FCB для роботи з декількома файлами, без виконання CLOSE при кожній зміні імені файлу. Помилкою вважається також усунення або перейменування відкритого файлу.

При цьому методі обмін даними між програмою і файлами здійснюється завжди через буфер або так звану область обміну з диском (DTA – Disk Transfer Area). Адреса буферу підтримується у покажчику до DTA. При запуску програми, покажчик до DTA вказує на область довжиною 128б у PSP. В подальшому програма може змінити (за допомогою спеціальних системних функцій) покажчик до DTA.

Необхідність у цьому може виникнути, якщо наприклад 128б недостатньо. Звісно, за новою адресою потрібно мати вільне місце відповідного розміру для потрібної DTA.

Опис FCB.

При розгляді системних функцій потрібно знати поля FCB. FCB ділиться на дві основні частини: головна частина розміром 37 байтів і префікс розміром 7 байтів. У головній частині утримується основна інформація про файл - пристрій, ім'я, розмір запису, її номер та ін.

Префікс використовується при необхідності зазначити спеціальні атрибути файлу (схований, системний і т.д.). Всі системні функції при методі FCB одержують як параметр адресу даного FCB. Якщо зі зсувом 0 від цієї адреси знаходиться байт із вмістом FFh, це означає, що FCB має префікс, і зсув для таких полів FCB зростає відповідно на 7 байт. Якщо байт із зсувом 0 має значення, відмінне від FFh, система вважає, що FCB не має префіксу та приймає це значення як номер пристрою. Номер пристрою ніколи не може бути FFh.

Поля FCB.

1. Зсув (0). Індикатор активного префікса FCB. Значення FFh означає активний префікс - наявність спеціальних атрибутів. Записується до виконання OPEN.

2. Зсув (1) .Резервовано . Повинно містити 00h.

3. Зсув (6). Задає атрибути файлу: прихований, системний, позначка диску або каталог. Для звичайних файлів і файлів тільки для читання не потрібний. Заповнюється до виконання OPEN.

4. Зсув (7). Номер пристрою. Перед відкриттям файлу користувач записує в дане поле номер пристрою, з яким хоче працювати. Використовується нумерація, при якій 0 означає поточний пристрій, 1- пристрій A і т. д. Якщо використано значення 0, система відкриває файл на поточному пристрою і записує в поле його дійсний номер. Заповнюється до виконання OPEN.

5. Зсув (8) і (16). Ім’я та розширення файлу. Поля вирівняні ліворуч і доповнені праворуч проміжками. Можна використовувати малі і великі літери. Резервовані імена (CON, AUX, COM1, COM2, PRN, LPT1, LPT2, NUL) інтерпретуються як пристрої, а не як імена файлів. Заповнюються до виконання OPEN.

6. Зсув (19) і (39). Номер поточного блоку і номер запису в блоці. Нумерація починається з 0. Кожний блок складається з 128 записів, чий розмір визначається полем зі зсувом (21). Ці два поля використовуються при послідовному доступі до файлу. При відкритті файлу поля не ініціалізуються. Користувач ініціалізує їх перед першою операцією (послідовне читання або запис 0, а система автоматично збільшує їх після кожної операції. У даному випадку послідовний доступ реалізований як варіант прямого. У будь-який момент користувач може позиціонуватися у довільному місці файлу і почати з цього місця послідовні операції.

7. Зсув (21). Розмір логічного запису файлу. Використовується як при послідовному так і при прямому доступі. У самому файлі, записаному на диску, не зберігається ніякої інформації про розмір логічного запису файл є просто послідовністю байтів. Розмір логічного запису в FCB відбиває, як розглядається логічна структура файлу в конкретній програмі. Очевидно, той самий файл можна обробляти, задавши різний розмір запису. У багатьох випадках, коли файл не складається з записів фіксованої довжини (наприклад, текстовий файл), зручно працювати з довжиною запису 1 байт. При відкритті файлу система заповнює

довжину запису 128б. Якщо потрібна інша довжина, її слід записати в поля 23, 27, 29, 40 FCB після відкриття файлу.

Виконання програм. Структура і завантаження EXE і COM файлів.

EXE файли.

EXE файли створюються редактором зв'язків LINK і складаються з двох частин: префікс і модуль виконання. Префікс складається з керуючої інформації і таблиці символів які переміщуються. Модуль виконання містить інструкції і дані програми і знаходиться безпосередньо за префіксом. У модулі виконання можна мати адресні константи, які потрібно налаштувати для конкретної адреси завантаження. Це адреси сегментів, значення яких в модулі обчислені відносно його початку. Під час виконання адресні константи повинні містити абсолютні адреси. Перетворення відносних адрес на абсолютні виконуються системною програмою завантаження, що використовує інформацію з таблиці символів які переміщуються – там описане місце розташування адресних констант у модулі виконання.

Керуюча інформація в префіксі.

Поле 1. Зсув 00h. 4Dh, 5Ah - ідентифікатор EXE файлу.

Поле 2. Зсув 02h. Довжина файлу по модулю 512 (залишок ділення довжини файлу на 512)

Поле 3. Зсув 04h. Довжина файлу в блоках по 512б.

Поле 4. Зсув 06h. Кількість елементів у таблиці символів які переміщуються.

Поле 5. Зсув 08h. Розмір префіксу у параграфах.

Поле 6. Зсув 0Ah. Мінімальна кількість параграфів, необхідних після завантаженої програми (зазвичай 0).

Поле 7. Зсув 0Ch. Максимальна кількість параграфів, необхідних після завантаження програми (FFFFh для програм, що завантажуються в молодші адреси пам'яті, 0000h- у старші адреси пам'яті)

Поле 8. Зсув 0Eh. Зсув SS у модулі виконання(у параграфах).

Поле 9. Зсув 10h. Значення SP при одержанні керування у модулі виконання.

Поле 10. Зсув 12h. сума (від'ємна сума всіх слів у файлі з ігноруванням перевантаження).

Поле 11. Зсув 14h. Значення IP, коли модуль виконання одержує керування.

Поле 12. Зсув 16h. Зсув CS у модулі виконання (у параграфах).

Поле 13. Зсув 18h. Початок таблиці символів які переміщуються (зсув з початку файлу в байтах)

Поле 14. Зсув 1Ah. Номер перекриття (0-для резидентної частини програми)

Адреса таблиці символів, що переміщуються, знаходиться у полі 13, а кількість її елементів – у полі 4. Кожний елемент таблиці дає зсув адресної константи у модулі виконання, який потрібно налаштувати щодо адреси завантаження, перед тим, як передати керування модулю. Елементи мають довжину 2 слова і формат сегмент:зсув (зсув у першому слові).

Завантаження програми здійснюється у такій послідовності:

1. Керуюча інформація з префіксу зчитується в робочу область системи.

2. Визначається розмір модуля виконання в залежності від значень полів 2,3,4,5.

3. Знаходиться перший вільний блок пам’яті, розмір якого достатній для модуля виконання і PSP. Блок привласнюється програмі. Якщо програма завантажується у старшій області пам’яті (поле 7), то шукається останній, а не перший вільний блок пам’яті необхідного розміру. Програма завантажується наприкінці, а не на початку блоку, частина блоку може залишитися вільною.

4. Будується PSP для програми. Сегмент після PSP називається старшим сегментом. Модуль виконання завантажується з початку цього сегменту.

5. З таблиці символів які переміщуються зчитується в робочу область. До кожного елементу додається значення старшого елементу. Отримана адреса у форматі сегмент:зсув вказує слово модуля виконання (у пам'яті), до вмісту якого додається значення старшого сегменту. Після закінчення цього процесу адресні константи в модулі виконання завантажені відповідно до адреси завантаження (стартового сегменту).

6. Регістрам ES і DS привласнюється значення сегменту програми (сегментна адреса PSP). SS і SP привласнюються значення полів 8 та 9 префіксу і додається стартовий сегмент. Значення CS утворюється як сума поля 12 у префіксі стартового сегменту. IP утворюється з поля 11.

7. Тепер вже модуль виконання налаштований (переміщений) відповідно до адреси завантаження. Керування передається за адресою CS:IP.

COM файли.

COM файли не мають префіксів. Вони побудовані так, щоб не містили адресних констант, залежних від адреси завантаження програми в пам'яті (Все в програмі подано, як зсув з початку сегменту для коду, включаючи дані і стек. Отже, розмір програми не може перевищувати 64К). COM представляє собою точну копію програми в двійковому вигляді, в якому її потрібно завантажити в пам'ять.

Тому завантаження COM-файлів зводиться до визначення великого вільного блоку пам'яті, побудові PSP і зчитування усього файлу в область після PSP.

Регістрам CS,DS,SS,ES привласнюється значення сегменту програми (сегментна адреса PSP). Покажчик інструкції IP одержує значення 100h. SP одержує значення FFFEh, тобто для стеку використовується кінець адресного простору з 64К. Байт за адресою FFFEh у стеку містить 00h, так що після кінця команди RET, наприкінці програми, в IP отримується значення 00h, тобто виконується інструкція INT 20h на початку PSP.

Перетворення EXE файлів на COM файли.

Перетворення здійснюється допоміжною командою EXE2BIN.EXE. Командний рядок має формат EXE2BIN <вхідний файл> [< вихідний файл >].

Вхідний файл повинен бути у форматі EXE. Він має розширення за замовчанням EXE. Загальна довжина інструкцій і даних не може перевищувати 64Кб. Не припускається сегмент типу STACK. Вихідні файли мають формат COM (копія програми в двійковому вигляді) і за замовчанням одержують розширення BIN.

Можливі два засоби перетворення в залежності від значення CS:IP у префіксі EXE файлу (поля 12 і 11):

1. Значення CS:IP-0:0. Це означає, що сегмент коду є першим у програмі і перша інструкція - ORG 0 (або взагалі немає інструкції ORG). Якщо в програмі є адресні константи, які потрібно налаштувати відповідно до конкретної адреси завантаження, це здійснюється під час перетворення. EXE2BIN поставить питання про абсолютну адресу завантаження програми й виконає налаштування відповідно до отриманої відповіді. Отже, для правильного виконання програма повинна бути

завантажена саме з цієї адреси. Неможливо використовувати систему для завантаження таких програм - їх повинні завантажувати (зчитувати) конкретні користувацькі програми. В цьому випадку PSP не будується.

2. Значення CS:IP дає зсув 100h з початку виконання модуля. Це означає, що програма буде виконувати як нормальний COM файл з побудовою PSP і інструкціями безпосередньо після нього. У цьому випадку програма не повинна містити адресних констант, які потрібно налаштувати, тобто таблиць символів які переміщуються у префіксі EXE файлу повинна бути порожньою (поле 4 = 0). Одержаний COM файл можна завантажувати в будь-яке місце пам'яті.

Якщо EXE файл не відповідає перерахованим обмеженням, EXE2BIN видає повідомлення "File cannot be converted " (файл неможливо перетворити) і закінчує роботу. Другий засіб кращий, тому що створюються головні COM файли.

Правила створення таких файлів:

1. Програма повинна починатися директивою ORG 100h і вхідна точка повинна бути на початку програми.

```
ORG 100h
START:
……..
END START
```

Після її завантаження CS містить сегментну адресу PSP, а IP встановлений в значеня 100h. Якщо потрібно почати виконання з іншого місця, першою інструкцією після ORG 100h повинна бути JMP.

1. Не припускається визначити окремі сегменти для даних і для стеку. Визначається тільки сегмент для коду. Використовується директива ASSUME, щоб залишити всі сегментні регістри сегменту для коду.

2. Дані можна розташувати у довільному місці програми, аби не плутати їх з кодом.

3. Не припускаються інструкції, що мають у якості операндів сегментні адреси. Зокрема, на початку програми відсутні визначення окремого сегменту даних, що є типом для EXE файлів.

4. При завантаженні програми SS, як і інші сегментні регістри, має значення сегментної адреси PSP. SP вказує останню адресу 64К байтового адресного

простору (наприклад, програма, що залишається резидентною). Можна змінити SP ще до використання стеку.

ВМІСТ PSP.

Поле 1. Довжина 2, зсув 0. Містить код інструкції INT 20h. Переривання 20h — один із засобів завершення виконання програми. Отже, програма може завершуватися інструкцією для переходу на початок PSP. CS повинен містити сегментну адресу PSP. Ця можливість виглядає цілком безглуздою при наявності системної функції для завершення програми - 4Ch. Не говорячи про те, що програма сама може виконати переривання 20h. Насправді INT 20h на початку PSP має інше призначення - підвищує надійність при виконанні програми з невирішеними зовнішніми зв'язками. Якщо програма виконує перехід до адреси, яка не визначена при редагуванні, то це призведе до її завершення. Редактор зв'язків дає значення 0 усім таким адресам, тобто буде здійснений перехід до початку PSP.

Поле 2. Довжина 2, зсув 2. Містить розмір наявної оперативної пам'яті в блоках по 16 байт (параграфах). Точніше, поле містить сегментну адресу останнього параграфу пам'яті. Це значення використовує програма CHKDSK у своєму звіті. Потрібно знати, що в деяких випадках це значення може не збігатися з дійсною кількістю пам'яті. Наприклад, деякі віртуальні диски використовують старші адреси пам'яті і змінюють інформацію про останню адресу. Тому може існувати різниця між полем 2 PSP і значенням, отриманим перериванням 12h BIOS, що витягає інформацію про наявну пам'ять.

Поле 3. Довжина 1, зсув 4. Резервовано.

Поле 4. Довжина 5, зсув 5. Дальній перехід (FAR JMP 000C0h) дає можливість виконати деякі вже існуючі програми, що використовують інший метод звернення до системних функцій. При цьому методі номер функції записується в CL замість AL, інші параметри задаються відповідно до опису функції. Потім виконується JMP (у поточному сегменті програми). Потім виконується необхідна функція, після чого керування повертається програмі. AX завжди губиться. Оскільки нові програми, що застосовують цей метод, майже не застосовуються, практично поле 4 не цікаве як засіб для виконання системних функцій. З іншого боку, друге слово поля містить корисну інформацію, яку можна використовувати з іншою метою, наприклад, як

індикатор, що володіє сегментом довжини не менше 64К. Якщо він менше, то поле містить його розмір.

Поля 5,6,7. Довжини 4, зсуви A,E,12 містять вектори переривань, якими вони були до запуску програми. Це INT 22h (адреса завершення), INT 23h (переривання програми за допомогою <Ctrl/C>), INT 24h(обробка критичних помилок). Програма може змінювати ці вектори як завгодно. При її завершенні система відновлює їхній попередній зміст з PSP. Наприклад, програма може створити власний обробник <Ctrl/C>, але після її завершення буде встановлена стандартна обробка. Якщо програма змінює деякі з цих полів у PSP, зміни залишаться і після її завершення.

Поле 8. Довжина 2, зсув 16 сегментна адреса батьківського PSP, таблиця файлів завдання JFT.

Поле 9. Довжина 2, зсув 2C містить адресу оточення програми. Зазначена тільки адреса сегменту, зсув = 0.

Поле 10. Довжина 24, зсув 2E. Робоча область

2E, 4б - SS:SP програми при вході в останній виклик INT 21h,

32h, 2б - число елементів у JFT (за замовчанням 20),

34h, 4б - адреса JFT (за замовчанням PSP:018h),

40h, 2б - версія в форматі функції 30h.

Поле 11. Довжина 3, зсув 50 містить інструкції INT 21h і RET FAR, що дає можливість виконувати побічно обслуговуючі функції системи. Для виконання цієї можливості програма повинна підготувати все необхідне для використання потрібної системної функції і потім здійснювати дальній перехід до PSP+50h.

Поле 12. Довжина 2, зсув 53. Резервовано.

Поля 13,14,15,16**.** Довжина 7,9,7,9 зсув 55,5C,65,6C відповідно містить префікси першого і другого FCB (поля 13,15) і самі перший і другий FCB (поля 14,16).

Ці поля допомагають програмам, що використовують для роботи з файлами метод FCB. Основна їхня мета - полегшити обробку параметрів з командного рядку програми. Передбачається, що ці параметри - імена файлів, що обробляються програмою. Система конструює за їх допомогою два невідкритих FCB у PSP. Якщо вони потрібні програмі, вона може їх відкрити і використовувати.

Можливі ускладнення: два FCB перекриваються як один з одним, так і з DTA у PSP, якщо використовується тільки перший з них і зміниться адреса DTA, то все добре. У протилежному випадку, один або обидва FCB потрібно перемістити з PSP до їхнього відкриття.

Поля 17,18. Довжини 1 і 127, зсув 80,81. Поле 17 містить довжину параметрів командного рядку, поле 18 - параметри командного рядку. Поля забезпечують доступ програми до всіх параметрів командного рядку.

Ці два поля перекриваються полем 19 - DTA, тому їх потрібно обробити до можливого руйнування.

Поле 19. Довжина 128, зсув 80 використовується як DTA за замовчанням. Після витягу необхідної інформації програма може використовувати частину PSP (від зсуву 5C і до кінця) як робочу область.