Тема 6 VESA BIOS: специфікація VESA, відеорежими VESA, функції VESA BIOS

Специфікація VESA

На початковому етапі розвитку відеоадаптерів SVGA серйозною проблемою була відсутність єдиного підходу до реалізації розширених по відношенню до VGA, відеоадаптерів. Відеоконтроллери SVGA між фірмами-виробниками, мали вельми істотні відмінності в своїй конструкції, а саме :

- різну нумерацію розширених відеорежимів
- різну регістрову модель (призначення додаткових, по відношенню до архітектури VGA, регістрів введення-виведення і механізми їх адресації
- різні методи логічної організації відеопам'яті (спосіб розбиття на банки і методи їх перемикання)

В результаті стало неможливим створення універсальної прикладної програми, яка могла б однаково успішно працювати з різними відеоадаптерами в розширених відеорежимах (із стандартними режимами VGA проблем не виникало). Найнадійніший апаратно-незалежний спосіб роботи з відеоадаптером – використовування відеосервісу BIOS. Проте, відсутність єдиної нумерації розширених відеорежимів не дозволяла використовувати навіть його. Проблему вирішила асоціація VESA. Суть запропонованих нею заходів зводилася до наступного :

- рахувати відеорежими SVGA тільки такі режими, для реалізації яких потрібно понад 256 Кбайт відеопам'яті.
- використовувати єдину для всіх відеоадаптерів SVGA нумерацію розширених відеорежимів.
- використовувати однаковий механізм перемикання банків відеопам'яті.
- доповнити стандартну Video BIOS спеціальним розширенням, в якому зберігаються програми, реалізовуючи розширені функції Video BIOS, тобто управління відеосистемою в розширених відеорежимах.

При розв'язувані проблеми класифікації і нумерації відеорежимів фахівці VESA врахували ту обставину, що практично всі ПК оснащені відеоадаптерами SVGA, мають 32-розрядні ЦП (80386 і вище). Це значить, що молодші і старші частини регістрів загального призначення (наприклад AL і AH регістра AX) мають в цих процесорах по 16 розрядів, тобто вміщають по 2 байти. Отже, номер відеорежиму, який при виклику функції 00h переривання Int 10h поміщається в AL, для розширених відеорежимів можна зробити двобайтовим. Двобайтова нумерація розширених відеорежимів, названих надалі VESA – режимами, стало для різних виробників і відеоадаптерів SVGA зручним компромісом : з одного боку всі вони зберегли свою оригінальну нумерацію SVGA – режимів, використовуючи для них однобайтні номери в межах 14h – 7Fh, з другого-кожний SVGA-режим отримав другий, єдиний для всіх відеоадаптерів номер, що складається з двох байт. Двобайтові номери VESA – режимів починаються з номера 100h. Відеоадаптер може встановлювати необхідний відеорежим при вказівці одного з двох номерів – власного однобайтного або стандартного двобайтового.

Крім єдиної нумерації розширених відеорежимів, стандартом VESA визначені способи їх використовування. З цією метою стандартний Video BIOS адаптера VGA був доповнений програмами, що управляють відеоадаптером в режимах SVGA. Це доповнення називається VESA – розширенням BIOS.

Таблиця 3.1 ***Відеорежим VESA***

Код режиму	Тип	Дозвіл	Глибина кольору	Палітра
100h	Граф	640*400	8	256
101h	Граф	640*480	8	256
102h	Граф	800*600	4	16

103h	Граф	800*600	8	256
104h	Граф	1024*768	4	16
105h	Граф	1024*768	8	256
106h	Граф	1280*1024	4	16
107h	Граф	1280*1024	8	256
108h	Текст	80*60	4	16
109h	Текст	132*25	4	16
10Ah	Текст	132*43	4	16
10Bh	Текст	132*50	4	16
10Ch	Текст	132*60	4	16
10Dh	Граф	320*200	15	32768
10Eh	Граф	320*200	16	65536
10Fh	Граф	320*200	24	16777216
110h	Граф	640*480	15	32768
111h	Граф	640*480	16	65536
112h	Граф	640*480	24	16777216
113h	Граф	800*600	15	32768
114h	Граф	800*600	16	65536
115h	Граф	800*600	24	16777216
116h	Граф	1024*768	15	32768
117h	Граф	1024*768	16	65536
118h	Граф	1024*768	24	16777216
119h	Граф	1280*1024	15	32768
11Ah	Граф	1280*1024	16	65536
11Bh	Граф	1280*1024	24	16777216
11Ch	Граф	1600*1200	15	32768
11Dh	Граф	1600*1200	16	65536
11Eh	Граф	1600*1200	24	16777216
1FFh	-	-	-	-

Функції VESA BIOS

Стандарт VESA уніфікував деякі найважливіші операції при роботі з відеоконтроллером: установку відеорежимів і управління відеопам'яттю. Починаючи з версії 2.0 даний стандарт підтримує роботу з лінійним буфером відеопам'яті. Повний опис VESA 3.0 на англійській мові можна вільно завантажити з Інтернету з серверу асоціації VESA (www.Vesa.org).

Звернення до VESA BIOS виконується по перериванню 10h з номером функції 4Fh. Після виконання виклику в регістрі AX буде повернено код результату. В AL буде знаходиться основний код повернення . Якщо AL = 4Fh – виклик успішно виконаний, якщо AL = 4Fh – викликана функція не підтримується даною версією BIOS. В AH буде записаний код, пояснюючий результат:

0 – функція успішно виконана;
1 – виклик не виконаний;
2 – функція не підтримується в даній апаратній конфігурації;
3 – виклик функції неможливий в даному відеорежимі;

Розглянемо деякі найкорисніші функції VESA BIOS 2.0, які підтримуються поширені в даний час моделями відеоконтроллерів.

Переривання INT 10h, функція 4Fh, підфункція 00h: отримати інформацію про версію VESA BIOS

Функція призначена для зчитування інформації про версію VESA BIOS. Вхідним параметром є адреса масиву розміром 256 байтів, при
виконанні запиту в нього записуються дані про відеоадаптер. Повна адреса цього масиву указується в регістрах es:di. Форма запису es:di загальноприйнята, вона означає, що в регістрі es знаходиться сегмент пам'яті, а в регістрі di – розташування (зсув) масиву в цьому сегменті.

Перед викликом переривання потрібно занести в регістри наступні значення:

- в AX – код 4F00h;
- в ES:DI – адреса буфера об'ємом 512 байт для збереження інформації про версію VESA BIOS (інформація зберігається у вигляді структури, описаної в таблиці 3.3).

Підфункцію 00h необхідно викликати перед початком використовування інших VESA-функцій. Вона дозволяє визначити, чи є у відеоконтроллера підтримка команд VESA, а якщо так–то якої версії. Щоб була можливість використовування лінійного буфера, версія VESA повинна не бути нижчою 2.0. Спочатку потрібно переконається, що функція була виконана: після виклику функції в регістрі AL повинен знаходиться код 4Fh. Потім слід перевірити наявність сигнатури 'VESA' в чотирьох перших байтах що повертається функцією блоку даних, а також переконається, що байт із зсувом 05h від початок блоку (старший байт номера версії) вміщує значення не менше 2

Таблиця 3.3 Формат інформації VESA BIOS

Зсув	Розмір	Описи
00h	4 байти	Сигнатура 'VESA'
04h	Word	Номер версії VESA (0200h і 0300h)
06h	Dword	Вказівник на рядок ЕОМ (з найменуванням мікросхеми відеоконтроллера)
0Ah	4 байти	Можливість графічного контролера Битий 0: 0-6-розрядний ЦАП, 1-8-розряд; Битий 1: 0- контроль VGA- сумісний, 1-сумісний; Битий 2: 0- звичайна робота з RAMDAC, 1- при програмування великих блоків інформації в RAMDAC використовувати байт очищення функції 09h біти 3-31 – зарезервовані
0Eh	DWORD	Вказівник на список відеорежимів, підтримуваних VESA і ЕОМ
12h	WORD	Число 64- кілобайтних блоків пам'яті (об'єм відеопам'яті, ділений на 64 Кбайт)
14h	WORD	Код версії програмної реалізації VESA
16h	DWORD	вказівник на рядок з назвою фірми виготівника
1Ah	DWORD	Вказівка на рядок з назвою виробу
1Eh	DWORD	Вказівник на рядок з кодом версії виробу
22h	222 байти	Зарезервований для подальших версій
100h	256 байт	Область даних для рядків ЕОМ

В першому стовпці даної таблиці вказані зсуви полів щодо початку масиву, адреса якого знаходиться в регістрах es:di.

Переривання Int 10h, функція 4Fh, підфункція 01h: отримати інформацію про параметри відеорежимів

Функція дозволяє перевірити наявність в даній версії BIOS режиму із заданим номером і визначити його властивості. Перед викликом переривання потрібно занести в регістри наступні значення:

в AX – код 4F01h;
в CX – код відеорежиму, параметри якого потрібно визначити;
в ES:DI – код буфера об'ємом 256 байт для збереження інформації про відеорежим (таб. 3.4);

Підфункцію 01h викликають, звичайно перед установкою нового відеорежиму, щоб упевнитися в його реалізації в даному відеоконтроллері і в можливості використовування лінійної адресації відеопам'яті в цьому режимі. Після виклику функції в регістрі AL повинен знаходиться код 4Fh, а в AH – код 0. Після цього потрібно перевірити біт 7 слова атрибутів режиму (лінійна адресація підтримується якщо він рівний 1), а потім прочитати адресу лінійного відеобуфера з 32- розрядного (подвійного) слова із зсувом 28h від початку структури даних.

Таблиця 3.4 Формат інформації про відеорежим

Зсув	Розмір	Опис
00h	WORD	Атрибути режиму Біт 0: 0-режим не підтримується; 1-режим підтримується; Біт 1: 0- завжди один зарезервований; Біт 2: 0- TTY функції не підтримуються; 1- TTY функції підтримуються; Біт 3: 0- монохромний режим, 1- графічний; Біт 4: 0- текстовий режим; 1- графічний; Біт 5: 0- VGA- сумісний режим; 1- VGA- несумісний; Біт 6: 0- режим віконної адресації відеопам'яті підтримується, 1- не підтримується; Біт 7: 0- режим лінійної адресації відеопам'яті не підтримується; 1- підтримується; Біти 8-15 – зарезервовані;
02h	BYTE	Атрибути вікна A Байт 0: 0- вікно не переміщуване, 1- підтримується переміщення вікна; Байт 1: 0- читання з вікна заборонено, 1- дозволено; Байт 2: 0- запис у вікно заборонений, 1- дозволено; Байт 3-7: - зарезервовані;
03h	BYTE	Атрибути вікна B (аналогічно байту атрибута A);
04h	WORD	Гранульованість (дробова) вікна (якнайменша величина в кілобайтах, на яку можна перемістити вікно);
06h	WORD	Розмір вікна в кілобайтах;
08h	WORD	Початковий сегмент вікна A в адресному просторі процесора;
0Ah	WORD	Початковий сегмент вікна B;
0Ch	DWORD	Far-адреса функції позиціонування вікна (вона еквівалентна функції 4Fh з підфункцією 05h);
10h	WORD	Число байтів, яке відводиться на даний рядок розгортки;
12h	WORD	Горизонтальний дозвіл екрану в пікселях (в графічному режимі) або знакомісцях (в текстовому режимі);
14h	WORD	Вертикальне розміщення екрану в пікселях або знакомісцях;
16h	Byte	Ширина знакомісця в пікселях;

17h	Byte	Висота -//-//-//-//-
18h	Byte	Число площин пам'яті;
19h	Byte	Число бітів на піксель;
1Ah	Byte	Число банків пам'яті;
1Bh	Byte	Тип моделі пам'яті: 00h- текстовий режим; 01h- графічний режим CGA; 02h- графічний режим Hercules; 03h- планарный режим; 04h- режим упакованих пікселів; 05h- ланцюжковий режим №4, 256 кольорів; 06h- режим Direct Color (HiColor або TrueColor); 07h- режим YUV; 08h- 0Fh- зарезервоване для доповнень VESA; 10h- FFh- зарезервоване для доповнень EOM;
1Ch	Byte	Розмір банку пам'яті в кілобайтах;
1Dh	Byte	Число повних відеосторінок (екранів), що поміщаються у відеопам'яті ;
1Eh	Byte	Зарезервований;

Поля DirectColor

1Fh	Byte	Розмір компоненту червоного кольору в бітах;
20h	Byte	Початкова бітова позиція компоненти червоного кольору;
		Дописати
25h	Byte	Розмір резервної компоненту в бітах;
26 h	Byte	Початкова бітова позиція резервної компоненти;
27 h	Byte	Атрибути режиму Direct Color;

Поля присутні тільки в VBE 2.0 і пізніших реалізаціях стандарту

28h	DWORD	Фізична (абсолютна) 32-розрядна адреса буфера кадру;
2C h	WORD	Вказівник на початок за екранної пам'яті, тобто на початок другої відеосторінки (введений в 2.0, але відмінений в 3.0) тепер поле даних вважається зарезервованим і містить значення 0;

Поля, присутні тільки в VBE 3.0 і вище

32h	WORD	Число байтів, які доводяться на один рядок розгортки в лінійному режимі;
34h	Byte	Кількість відеосторінок в сторінковому режимі;
35h	Byte	Кількість відеосторінок в лінійному режимі;
36h	Byte	Розмір компоненти червоного кольору в бітах в лінійному режимі;
37h	Byte	Початкова бітова позиція компоненти червоного кольору в лінійному режимі;
38h	Byte	Розмір компоненти зеленого кольору в бітах в лінійному режимі;
39h	Byte	Початкова бітова позиція компоненту зеленого кольору в лінійному режимі;
40h	Byte	Розмір компоненти синього кольору в бітах в лінійному режимі;
41h	Byte	Початкова бітова позиція компоненти синього кольору в

		лінійному режимі;
42h	Byte	Розмір резервної компоненти в бітах в лінійному режимі;
43h	Byte	Початкова бітова позиція резервної компоненти в лінійному режимі;
44h	DWORD	Максимальна частота виведення пікселів в графічних режимах (в Гц);
48h	184 байти	Зарезервовано;

Підфункції 00h і 01h необхідно використовувати, коли немає повної впевненості в тому, що відеоконтроллер підтримує що використовується у вашій програмі відеорежим.

Приклад 3.1 Написати фрагмент прикладу, перевіряючого відповідного відеоадаптера стандарту VESA, а також перевірити, чи підтримується вибраний вами відеорежим чи ні і одночасно прочитати в масив INFO інформацію про нього. Перевірити об'їм відеопам'яті для підтримки відеорежиму:

```
test_ 1: push es              ; збереження вмісту es;
         mov es, Info                 ; значення сегментного буфера Info;
         xor   di, di                 ; адреса початку буфера;
         mov ax, 4F00h                ; код запиту функції;
         int    10h                   ; звернення до BIOS;
         cmp ax, 4Fh          ; стандарт VESA підтримується?;
         jz     test_2                ; так, підтримується;
         pop  es                      ; ні, виконання задач неможливе;
                                      ; аварійне повідомлення;
test_2: mov ax, 4F01h         ; код запиту функції;
         mov cx, Newmode              ; код потрібного відеорежиму;
         int 10h              ; звернення до BIOS;
         pop es               ; відновлення вмісту es;
         cmp ax, 4Fh                  ; потрібний режим підтримується?;
         jz    test_3                 ; так, продовження перевірок;
                                      ; ні, вказаний відеорежим не
                                      ; підтримується
    test_3: mov fs, info      ; сегмент з даними про режим;
    test byte ptr fs:[di],1   ; об'їм відеопам'яті достатній?;
       jne stmd                    ; так, кінець перевірок;
                                   ;недостатньо відеопам'яті для підримки
                                   потрібного відеорежиму;
```

Пояснення:

По–перше, в стеку зберігається вміст регістра es, якщо він ще не використовувався і його вміст не має значення, то команди push і pop можна виключити. Далі в регістр es записується значення сегменту, що містить буфер Info, а di очищається для розміщення даних з початку буфера. Потім в регістр ax запиту код запиту функції і відбувається звернення до переривання BIOS Int 10h.Якщо відеоадаптер відповідає стандарту VESA, то при поверненні з BIOS в ax знаходиться код 4Fh . Це і перевіряє команда cmp ax, 4Fh. Якщо результат перевірки позитивний, то наступна команда jz виконує перехід на мітку test_2. Якщо результат негативний (код відрізняється від 4Fh), то далі виконання задачі неможливе, на екран треба вивести повідомлення. У разі успішного виконання запиту, в буфері Info знаходиться загальна інформація про видеоадаптер. Нас цікавить тільки об'єм відеопам'яті, вказаний в слові масиву Info із зсувом 12h. Об'єм відеопам'яті у разі його використовування при виконанні задачі треба зберегти в області даних, оскільки вже на наступному кроці вміст масиву Info зміниться. Якщо відеоадаптер відповідає стандарту VESA, то треба перевірити, підтримується вибраний відеорежим чи ні і одночасно прочитати в масиві Info інформацію

про нього. Спочатку другого блоку вказується код запиту функції і потрібного відеорежиму, після чого відбувається звернення до BIOS для виконання запиту. Після повернення в задачу відновлюється збережений в стеку початковий вміст регістра es. Якщо відеоадаптер розраховано на підтримку вибраного відеорежиму, то в ax буде знаходиться код 4Fh, це і перевіряє передостання команда прикладу. Якщо умова виконана, то відбудеться перехід на мітку test_3 для продовження перевірок. Якщо код в регістрі ax відрізняється від 4Fh, то відеоадаптер підтримує необхідний відеорежим. Залежно від конкретних особливостей задачі її виконання може бути або перервано, або зроблена спроба перейти на роботу в іншому режимі, вимагаючи менший об'єм пам'яті.

Якщо в регістрі ax знаходиться код 4Fh, то залишається перевірити реально існуючий об'єм відеопам'яті. BIOS виконує потрібні для перевірки обчислення і порівняння, результат яких поміщається в нульовий розряд нульового байта масиву Info. Задачі потрібно тільки перевірити стан цього розряду.. Після блоку test_2 був відновлений початковий стан регістра es і доступ до масиву Info з використанням цього регістра вже неможливий . Для подальшої роботи з даними про режим потрібно виділити інший сегментний регістр (наприклад, fs або gs, який не використовується функціями BIOS).

III блок: Регістр di очищений і вказує початок буфера, що містить дані про режим. Для доступу до цих даних використовується сегментний регістр fs, тому в нього записується значення змінної info. В команді test запис byte ptr байт явно вказує, що операндом є байт. Тип операнду вказується в тих випадках, коли Макроасемблер не може його визначити по запису команди. Якщо нульовий розряд байта встановлений, то об'єм пам'яті достатній і команда jne передасть управління на мітку stmd (слід. приклад). Якщо нульовий розряд очищений, то об'єм відеопам'яті не достатній для підтримки відеорежиму.

Переривання INT 10h, функція 4Fh, підфункція 02h:
Встановити відеорежим із заданим номером.

Функція встановлює відеорежим із заданим номером і режим адресації відеопам'яті (щоб використовувати лінійну адресацію, потрібно встановить біт 14 в коді режиму). Перед викликом переривання потрібно занести в регістри наступні значення:

в AX – код 4F02h;
у BX – код відеорежиму;

Код режиму має наступний формат:

- біт 0-8- номер режиму (якщо біт 8=0, то це режим виробника відеоконтроллера, якщо 1- VESA- режим);
- біт 9-13- зарезервовані для подальших розширень і рівні 0;
- біт 14- тип режиму адресації відеопам'яті: 0- сторінковий режим (пам'ять адресується ланками по 64 Кбайт), 1- лінійний режим;
- біт 15- ознака очищення відеопам'яті при перемиканні режимів (0- очищати пам'ять, 1- не очищати);

Стандартизовані номери режимів приведені в табл. 3.1

При роботі з функцією установки відеорежиму потрібно враховувати наступне.

- Режими з номерами менше 100h визначаються виробником відеоадаптера і не обробляються VESA- фукціями. Якщо функції установки відеорежиму буде посланий код із значенням менше 100h, то вона передасть його на обробку перериванню BIOS 10h.
- Відеоадаптери різних виробників можуть інтерпретувати режими TrueColor з номерами 10Fh, 112h, 115h, 118h, 11Bh як 24 або 32- бітові, тобто дані режими визначені не однозначно. Для уточнення числа байт, виділених на одну крапку, потрібно викликати функцію 01h, отримати інформацію про відеорежим і перевірити розмір резервної компоненти (байт із зсувом 25h): якщо 0- 24- бітовий режим, якщо 8- 32-бітовий.

– Коди режимів з розширенням більше 1280*1024 не стандартизовані і їх доводиться визначати по таблиці доступних режимів, заданій виробником відеоадаптера. Режим 81FF служить для обчислення доступу до всієї наявної відеопам'яті без зміни її вмісту.

Приклад. Написати фрагмент програми збереження початкового і установки нового відеорежиму.

```
Stmd  mov ax, 0F00h     ; код функції "Читання відеорежиму"
      int 10h           ; BIOS читає поточний відеорежим;
      mov oldMode, al   ; збереження коду відеорежиму;
      mov bx, NewMode; код одного з режимів VESA;
      mov ax, 4F02h     ; код запиту функції BIOS;
      int 10h           ; BIOS виконує запит ;
      cmp ax, 4Fh       ; режим встановлений?;
      jz succ           ; так, на продовження програми;
; Помилка при встановлені відеорежиму;
```

В даному фрагменті використані імена OldMode і NewMode. Перше з них може бути тільки ім'ям байта, розташованого в області даних програми. NewMode може бути ім'ям розташованого в області даних слова або константи, якій зарані привласнено конкретне значення, скажемо NewMode=110h. Крім того, код режиму може бути вказаний в команді явно, наприклад:

```
mov bx, 110h
```

Якщо при виконанні задачі відеорежим встановлюється тільки один раз, то вибір способу вказівки NewMode довільний. Проте використовування змінних є більш універсальним. Перші 3-и команди останнього фрагмента зчитують його в байті OldMode. Наступні три команди встановлюють новий режим, в якому працюватиме задача. Після другого повернення з BIOS аналізується вміст регістра ax. Якщо в ньому записаний код 4Fh, то потрібний режим встановлено і відбувається перехід на початок наступного прикладу, інакше виникає надзвичайна ситуація, оскільки попередньо виконані перевірки указують, що відеоадаптер підтримує потрібний режим.

Для визначення значення початкового відеорежиму видається запит 0Fh переривання Int 10h. При його виконанні BIOS просто зчитує в регістр al вміст байта 0000:0449, що відноситься до області даних BIOS. Виконати ці дії можна безпосередньо в задачі без звернення до BIOS. У такому разі виконуються приблизно 30 команд які BIOS виконує при розшифровці запиту, збереженні і відновленні вмісту регістрів.

Переривання Int 10h, функція 4Fh, підфункція 03h: визначити код поточного відеорежиму.

Розмір коду режиму VESA перевищує розмір байта, тому при установці цих режимів в байт 0000:0449 записується так званий код EOM, тобто код вибраний по розсуду розробників відеоадаптера. В ROM BIOS є дві таблиці відповідно для перетворення кодів відеорежимів з VESA EOM і навпаки. Ніяких угод відносно значень кодів EOM не існує, крім того, його розмір не перевищує семи розрядів, а значення відрізняється від кодів відеорежимів IBM. Функція 4F03h читає код поточного відеорежиму з байта 0000:0449 і перетворює його в код режиму VESA по таблиці відповідності, що зберігається в ROM BIOS. Якщо у вказаному байті знаходився код одного з відеорежимів IBM, то його значення не змінюється. В цьому відношенні функція 4F03h більш універсальна ніж функція 0Fh.

Перед викликом переривання потрібно занести в регістр Ax код 4F03h. Після виконання функції в регістрах будуть розміщені наступні значення:

в AX – код результату виконання функції;

у BX – код поточного відеорежиму (табл.3.1);

Переривання Int 10h, функція 4Fh, підфункція 04h: зберегти або відновить стан відеоконтроллера

Мається на увазі збереження поточного вмісту регістрів кольору відеоадаптера (DAC) і деяких величин, що зберігаються в області даних BIOS. (Регістри DAC використовуються тільки при роботі в режимах packed pixel graphics). Функція 4F04h виконує копіювання вищеназваних величин у вказаний масив або, навпаки, в область даних BIOS. Крім того, вона дозволяє визначити розмір масиву, необхідний для розміщення величин, що зберігаються. Перед викликом функції заповнюється декілька регістрів. Обов'язково заповнюються регістри CX і DX.

в AX – код 4F04h;

в CX – прапори, що уточнюють, стан яких саме блоків відеоконтроллера потрібно зберегти або відновити (біт 0- стан апаратури, біт 1- стан даних BIOS, біт 2- стан DAC, біт 3- стан регістрів, біти 4-15- зарезервовані);

в DX – код запрошуваної дії (0- отримати розмір буфера, необхідного для збереження стану, 1- зберегти стан, 2- відновити стан);

Якщо регістр DX очищений, то регістри BX і ES не заповнюються. Після виконання запиту в DX знаходиться кількість байтів пам'яті, яку треба виділити для збереження вказаних в CX групи величин. Тепер можна вибрати розташування масиву в пам'яті і запитати збереження стану. Якщо в DX заданий код 1 або 2, то в парі регістрів es:dx указується повна адреса масиву, в якому потрібно відновити раніше збережені величини.

Після виконання функції в регістр AX буде поміщений код повернення, у BX – число блоків по 64 байти, необхідні для збереження стану.

Підфункція 04h дозволяє заповнити стан відеоконтроллера перед переходом в який-небудь інший режим, щоб потім можна було з її допомогою повернуться в початковий стан.

Процедури для роботи з вікнами відеопам'яті

На відеоадаптері обов'язково розташована оперативна пам'ять, яку прийнято називати ***відеопам'яттю.*** Відеоконтроллер безперервно виводить вміст частини відеопам'яті на екран монітора, причому розмір цієї частини залежить від встановленого відеорежиму. Специфічною особливістю сімейства IBM є обмеження простору доступних адрес розміром 1 Мбайт. Воно ділиться на сегменти, граничний розмір яких складає 64 Кбайт (1 Кбайт = 1024 байтам). Всього в просторі адрес поміщається 16 сегментів граничного розміру, 10 з них займає оперативна пам'ять ПК. Для доступу до відеопам'яті виділяється один сегмент розміром в 64 Кбайт, який частіше за все має адресу A000h для графічних і B800h для текстових режимів. Тільки при вказівці цих сегментів графічна або текстова інформація буде направлена у відеопам'ять або зчитана з неї, тому їх значення ні за яких умов не повинні змінюватися задачею. Розмір відеосегменту значно менше реального об'єму відеопам'яті. Для того, щоб задачі могли працювати зі всією пам'яттю, весь простір відеопам'яті ділиться на сегменти розміром 64 Кбайт, які пронумеровані, починаючи з нуля. Такі сегменти називаються вікнами або відеовікнами. В спеціальному регістрі відеоконтроллера зберігається номер поточного вікна. При зверненні до відеопам'яті відеоконтроллер додає його до 16- розрядної адреси, вказаної задачею, і одержує абсолютну адресу. Кількість розрядів в абсолютній адресі залежить від граничного об'єму пам'яті, який може бути встановлений на відеоадаптері. Таким чином, повна адреса відеопам'яті складається з двох частин. Молодша частина є відносною адресою (зсувом в сегменті), а старша частина номером поточного вікна, що зберігається в одному з регістрів відеоконтроллера. Задача може змінювати поточне вікно за допомогою функції 4F05h.

Переривання Int 10h, функція 4Fh, підфункція 05h: управління вікном відеопам'яті.

Дана функція читає або змінює номер поточного вікна відеопам'яті. Наявність цієї функції дозволяє задачам працювати із всім простором відеопам'яті.

Перед викликом переривання потрібно занести в регістри наступні значення:

в AX – код 4F05h;

у BX – номер вікна і запрошувана дія. Нуль в регістрі BH (старший байт регістра вх) означає установку нового вікна з номером, що вказується в DX. Одиниця в регістрі BH означає читання номера поточного вікна.

Стандартом VESA передбачена можливість існування у відеоадаптера двох вікон A і B. Нуль в регістрі BL (молодшому байті регістра bx) відповідає вікну A, а одиниця – вікну B.

Після виконання функції в регістрі AX буде розміщений код повернення, а в DX – адреса вікна відеопам'яті.

Підфункція 05h застосовується у віконному режимі адресації, коли в кожний момент часу процесору доступна тільки одна ділянка (сегмент) відеопам'яті розміром в 64К байт. Її необхідно викликати кожного разу, коли потрібно перейти до роботи з іншою ділянкою.

Пряме звернення до BIOS

Відеоадаптери розрізняються не тільки адресами регістрів, в яких зберігається номер поточного вікна, але і тим, в яких розрядах цих регістрів він розташовується. Тому при описі функції 4F05h обумовлено, що номер вікна виражається в одиницях приросту його значень, вона зберігається в змінній GrUnit, вона рівна 1, тільки для одного відеоадаптера вийшло інше значення.

Приклад

Написати фрагмент програми установки вікна, використовуючи спосіб прямого звернення до BIOS.

```
mov dx, Cur_win   ; запис в dx значення вікна
xor bx, bx        ; ознака установки вікна
mov ax, 4F05h     ; код функції BIOS
int 10 h          ; звернення до BIOS
```

Для установки вікна за допомогою функції 4F05h його номер поміщається в регістр dx, а регістр bx очищається. Після в ax записується код функції і виконується команда Int 10h. Поточний номер вікна вибирається із змінної Cur_win. Вона має розмір слова і розміщається в розділі даних задач.

Прискорення роботи з вікнами

В описі стандарту VESA не рекомендується використовувати пряме звернення до BIOS для установки вікна. Причина в тому, що при зверненні до BIOS з використанням переривань, виконується багато допоміжних дій, пов'язаних із збереженням регістрів і розшифровкою коду запиту. Специфічною особливістю переривання Int 10h є те, що на звернення до нього реагують компоненти DOS і деякі резидентні (постійно що знаходяться в оперативній пам'яті) задачі. В результаті кількість додаткових непотрібних дій виявляється значно більше кількості корисних дій, що виконують запис або читання вікна. З цієї причини розробники стандарту VESA рекомендують використовувати тільки процедуру BIOS, яка виконує читання або запис вікна і складається всього з 10 – 15 команд. В описі стандарту процедура називається Video Memory Control (VMC). Для цього потрібно зберегти в області даних задачі адреси підпрограми BIOS, що виконує роботу з відео вікнами. Адреса складається з сегменту і зсуву, тому при його пересилці як посередник використовується 32- розрядний регістр eax.

```
mov eax,   es:[di+0ch]   ; читаємо на eax адресу підпрограми
mov VMC, eax             ; і зберігаємо його в VMC
```

При виклику процедури VMC для читання або установки вікна регістри bx і dx заповнюються також як і при зверненні до функції 4F05h.

Приклад Фрагмент читання вікна прямим зверненням до процедури BIOS.

```
mov bx, 100h  ; ознака читання вікна
call [VMC]    ; звернення до процедури
```

Переривання Int 10h, функції 4Fh, підфункції 06h: отримати або встановити довжину логічного рядка розгортки

Перед викликом переривання потрібно занести в регістри наступні значення:

в AX – код 4F06h

у BL - код виконуваної операції (0 – встановити довжину рядка розгортки в пикселях,1- отримати поточну довжину рядка розгортки в пікселях, 2 – встановити довжину рядка розгортки в байтах, 3 – отримати максимально можливу довжину рядка в пікселях).

в CX – необхідну ширину рядка (задається тільки для операцій 0 і 2: при BL =0 в пік селях, при BL=2 – в байтах.

Після виконання функції в регістрі знаходяться наступні величини :

в AX – код повернення.

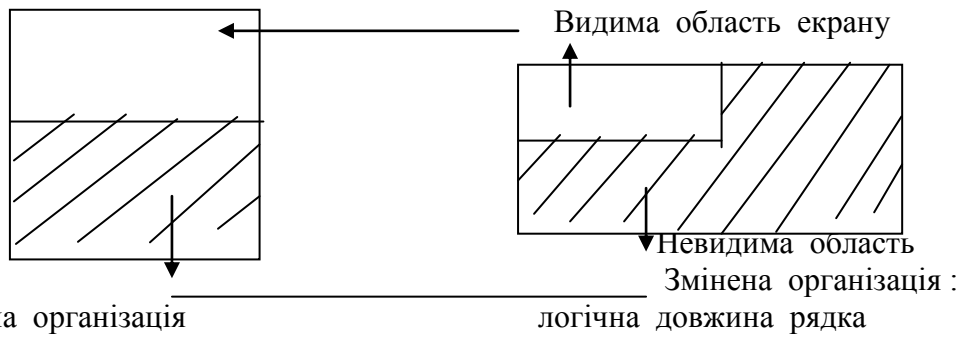
у BX – кількість байтів в рядку.

в CX – кількість точок в рядку.

в DX – максимально можлива кількість рядків вказаного розміру.

Процедура BIOS обчислює вміст DX шляхом ділення об'єму пам'яті, встановленої на відеоадаптері, на розмір рядка в байтах. На практиці вміст DX використовується украй рідко.

Підфункція 06h використовується для зміни організації відеопам'яті з метою спрощення адресації (шляхом вирівнювання довжини рядка на величину, рівну $2^n = 1024$ або 2048). Відеоконтроллери дозволяють робити довжину рядка у відеопам'яті (логічну) більше реальної (фізичної) довжини рядка екрану за рахунок реорганізації невживаних областей пам'яті. Звичайно, невидима область розташовується під видимою, проте після збільшення логічної довжини рядка відбувається перерозподіли пам'яті – з'являються невидима ділянка справа, а висота нижньої ділянки ,яка не відображається, зменшується.



видеопам. більше фізичної

Вирівнювання логічної довжини рядка на 2^n дозволяє :

- спростити обчислення адрес пам'яті (використовуючи зсув і логічні операції замість множення і додавання).
- Реалізувати (при необхідності) плавну прокрутку зображення по вертикалі апаратними засобами відеоконтроллера.
- Істотно спростити контроль меж сегментів при віконному режимі адресації відеопам'яті.

Переривання Int 10h, функції 4Fh, підфункції 07h: отримати або встановити координати лівого верхнього кута екрану.

Функція встановлює або читає координати лівого верхнього кута видимої області відеопам'яті, що виражені у вигляді номерів рядка і стовпця.

Перед викликом переривання вимагається занести в регістри наступні значення:

В AX – код 4F07h

у BH – код виконуваної операції

00h – встановити початок екрану

01h – отримати початок екрану

80h – встановити початок екрану в процесі зворотного ходу променя по кадру

в CX – перший піксель, що відображається, на рядку розгортки (задається тільки при BL=00h або 80h)

в DX – перший рядок розгортки, що відображається

Після виконання функції в регістрах будуть розміщені наступні значення:

В AX – код повернення

В CX – перший піксель, що відображається, на рядку розгортки (тільки при BL=01h)

в DX – перший рядок розгортки, що відображається (при BL=01h)

Підфункція 07h має 2-і основні області застосування: прокрутка екрану і перемикання екрану. В обох випадках відбувається переміщення по відеопам'яті області, що відображається на екран, але в першому випадку плавно, а в другому стрибкоподібно. Таке переміщення можливо завдяки тому, що система управління пам'яттю відеоконтроллера дозволяє довільно вибирати точку, з якої починається виведення на екран (позицію лівого верхнього кута екрану) щодо відеопам'яті. Для реалізації прокрутки зображення вліво необхідно перед виведенням чергового кадру додавати 1 до номера початкової точки, для реалізації прокрутки вправо – віднімати 1. Для реалізації прокрутки вгору необхідно до номера початкової точки додати число точок в рядку, для реалізації прокрутки вниз – віднімати це число точок.

Для виконання перемикання необхідно обчислити зсув другого вікна щодо початку відеопам'яті. Воно рівне числу точок в рядку зображення, помножене на число рядків на екрані і на число байт, що доводяться на одну точку. Далі для кожного нового кадру відбувається почергова установка номера першого пікселя, що виводиться (0 або величина зсуву другого вікна). Поки одне вікно виводиться на екран, відбувається перезапис інформації в інше вікно. Таким чином, можна виключити ефекти "снігу" і розрізання зображення, що виникають при одночасному записі в ділянку відеопам'яті і виведені на екран інформації з цієї ж ділянки. Можна сказати, що функція 4F06h дозволяє створювати умови для прокрутки в горизонтальному напрямі, а функція 4F07h виконує вказану прокрутку.

Приклад Написати фрагмент програми установки нового початку області пам'яті, що відображається .

```
xor bx, bx          ; ознака зміни початку області, що
                    ; відображається
mov cx, BaseCol     ; номер точки в початковому рядку
mov dx, BaseRow     ; номер початкового рядка
mov ax, 4F07h       ; код ф-ції
Int 10h             ; звернення до BIOS
```

В даному прикладі в регістр cx поміщається номер її першої точки, в dx – порядковий номер першого рядка, що відображається .

Всі номери починаються з 0. Окрім цього регістр bx очищається, що є ознакою зміни початку області, що відображається . В регістрі ax поміщається код 4F07h – функції, яка дозволяє перемістити початок робочої області в будь-яку точку відеопам'яті (обмеження – від вибраної точки до кінця відеопам'яті повинен залишатися простір, достатній для розміщення робочої області). Координати вибраної точки указуються
у вигляді рядка і стовпця. В прикладі значення координат початку робочої області вибираються із змінних BaseCol і BaseRow, які повинні бути описаний в сегменті даних програми.

Регістри кольору відеоадаптера

У відеоадаптера є 256 регістрів DAC, в яких зберігаються коди базових кольорів. Вони застосовуються тільки при роботі у відеорежимах pasked pixel graphics і не використовуються в режимах direct color. Базових кольорів три–червоний , зелений і синій. Відповідно до стандарту IBM VGA код базового кольору займає 6 двійкових розрядів. У деяких сучасних відеокарт з'явилася можливість збільшення розміру коду до 8 розрядів.

Переривання Int 10h, функції 4Fh, підфункції 08h: отримати або змінити формат регістрів палітри.

Функція призначена для визначення або зміни розміру коду базових кольорів, що зберігаються в регістрах кольору.

Перед викликом переривання необхідно занести в регістри наступні значення:

В AX – код 4F08h
В BL – код виконуваної операції
(0 – встановити формат регістрів палітри, 1 – отримати формат регістрів палітри)
у BH (при BL=0) – необхідна розрядність регістрів ЦАП в бітах (допустимі тільки значення 6 і 8)

Після виконання функції в регістрах будуть розміщені наступні значення:

В AX – код повернення
У BH – поточна розрядність регістрів ЦАП в бітах.

В даний час функція застаріла, оскільки для роботи з широкою колірною палітрою набагато ефективно використовувати режим True Color. Єдиною областю її застосування, ймовірно, є графічні редактори, що працюють з форматом файлів PCX.

Переривання Int 10h, функції 4Fh, підфункції 09h: зберегти або змінити вміст регістрів ЦАП.

Функція зчитує або встановлює вміст таблиці регістрів цифро-аналогового перетворювача (DAC), який ставить у відповідність
кодам кольору реальні значення по червоному, зеленому і синьому компонентах.

Перед викликом функції необхідно занести в регістри наступні значення:

в AX – код 4F09h

в BL – код виконуваної операції
(00h – встановити палітру, 01h – прочитати (зберегти) палітру, 02h – встановити додаткову палітру, 03h – прочитати додаткову палітру, 80h – встановити палітру в процесі зворотного ходу променя по кадру).

в CX – число завантажуваних (або що зберігаються) регістрів палітри
(максимум 256).

в DX – номер першого із завантажуваних (що зберігаються) регістрів.

В ES:DI – вказівник на таблицю, з якої завантажується або в якій зберігається вміст регістрів палітри.

Формат рядка таблиці палітри наступний: байт вирівнювання (нульовий), байт інтенсивності червоного кольору, байт інтенсивності зеленого кольору, байт інтенсивності синього кольору.