Лабораторна робота №º11
з дисципліни "Архітектура комп'ютера"
для студентів напряму 6.050103 "Програмна інженерія"

Використання функцій BIOS для роботи з відео в текстовому та графічному режимах

МЕТА РОБОТИ: опанувати функції BIOS для роботи з відео в текстовому та графічному режимах; розвинути навики складання програм для виведення різнокольорових рядків символів та графічних зображень; відтранслювати і виконати в режимі відлагодження програми, складені відповідно до свого варіанту.

Функції BIOS переривання INT 10h для роботи з відео в текстовому режимі

У текстовому режимі прикладна програма може вивести інформацію на екран одним з таких способів.

• **За допомогою функцій MS DOS.** Якщо на комп'ютері встановлена система MS DOS або її емулятор, для виведення текстових даних на екран можна скористатися функціями переривання INT 21h. Дані функції дозволяють перенаправити потоки введення-виведення на будь-який інший пристрій, такий як принтер або диск. Виведення на екран за допомогою функцій переривання INT 21h виконується досить повільно, і колір символів змінити не можна.

• **За допомогою функцій BIOS.** Вивести символи на екран можна також за допомогою функцій переривання INT 10h, оброблення якого виконується системою BIOS, а не DOS. Вони виконуються набагато швидше, ніж функції переривання INT 21h, і дозволяють змінити колір тексту на екрані. При заповненні символами великих областей на екрані за допомогою функцій переривання INT 10h, можна помітити невелику затримку виведення. Крім того, дані, що виводяться на екран, не можна перенаправити на інший пристрій.

• **Прямий доступ в відеопам'ять.** Вивести символи на екран можна також шляхом переміщення їх безпосередньо в область пам'яті відеоадаптера. При цьому досягається максимальна швидкість виведення, проте дані також можна перенаправити на інший пристрій. На зорі розвитку ПК, коли основною операційною системою була MS DOS, в прикладних програмах, таких як текстові процесори і електронні таблиці, використовувався саме цей метод виведення даних на екран. Слід зазначити, що даний метод також можна використовувати при роботі програми в повноекранному режимі під керуванням операційних систем Windows NT, 2000 і XP.

Залежно від поставлених завдань, в застосунку може використовуватися один з трьох запропонованих вище способів виведення даних на екран. Якщо на перше місце ставиться швидкість виведення на екран, то потрібно скористатися прямим виведенням у відеопам'ять. В інших випадках слід віддати перевагу функціям BIOS. Функціями DOS варто користуватися тільки тоді, коли вихідний потік даних може бути перенаправлений на інший пристрій або коли екран спільно використовується кількома програмами. Слід зазначити, що для виведення даних на екран у функціях MS DOS використовуються функції BIOS, а у функціях BIOS - прямий доступ до відеопам'яті.

Запуск програм в повноекранному режимі

Програми, в яких використовуються відеофункції BIOS, можуть виконуватися в перерахованих нижче операційних системах і оболонках:

• "чистій" системі MS DOS;

• емуляторі DOS системи Linux;

• в повноекранному режимі в системі Windows.

У середовищі Windows в повноекранний режим можна перейти такими способами.

• Спочатку створити ярлик для виконуваного EXE-файлу програми. Потім відкрити вікно властивостей ярлика, перейти на вкладку Screen (Екран) і встановити перемикач Full-screen mode (Повноекранний режим). Після цього запустити програму за допомогою ярлика.

• Відкрити вікно командного рядка з меню Start (Пуск) і для перемикання в повноекранний режим натиснути клавіші <Alt + Enter>. Потім за допомогою команди CD (Change Directory, або Змінити каталог) перейти в каталог, що містить EXE-файл програми, і запустити програму з командного рядка, ввівши її ім'я і натиснувши клавішу <Enter>. Якщо знову натиснути клавіші <Alt + Enter>, вікно командного рядка перемикнеться з повноекранного у віконний режим.

Текстовий режим роботи відеоадаптера

Відеоадаптер комп'ютера може працювати в двох режимах: текстовому і графічному. Під час початкового завантаження MS DOS відеоадаптер переводиться в текстовий режим роботи і встановлюється відеорежим номер 3 (кольоровий текстовий режим, 25 рядків і 80 стовпців). Однак крім текстового, існує і ряд графічних відеорежимів, частину з яких перераховано в табл. 3.

У текстовому режимі рядка нумеруються з нуля зверху вниз екрану. Висота кожного рядка визначає розмір відображуваних на екрані символів і залежить від поточного використовуваного шрифту. Стовпці екрану нумеруються з нуля зліва направо. Ширина кожного стовпця, як і висота рядка, також визначає розмір відображуваних на екрані символів і також залежить від поточного використовуваного шрифту.

Шрифти. Зовнішній вигляд символів, що відображаються на екрані, визначається спеціальною таблицею, яка перебуває в пам'яті і містить образи символів шрифту. У перших версіях BIOS ця таблиця розташовувалася в ПЗУ, однак з часом в BIOS з'явилися функції, завдяки яким прикладна програма під час виконання може завантажити з пам'яті власний шрифт. Це стимулювало користувачів до розробки оригінальних шрифтів для текстового режиму роботи відеоадаптера.

Текстові відеосторінки.

Текстова пам'ять містить 8 відеосторінок і займає в адресному просторі комп'ютера (за межами звичайної пам'яті) 32 Кбайти від сегментної адреси B800h. Починається вона з відео 0, адреса якої збігається з адресою всієї відеопам'яті. Кожна сторінка займає 4 Кбайт; таким чином, сторінка 1 починається з сегментної адреси B900h, сторінка 2 - з адреси BAOOh тощо.

При включенні комп'ютера активною (видимою) стає відеосторінка 0. Зміна відеосторінок здійснюється викликом функції 05h переривання 10h BIOS.

Таким чином, у програми з'являється можливість під час відображення однієї сторінки виводити дані в іншу приховану відеосторінку, а потім швидко перемикнути вміст екрану. Раніше при створенні високопродуктивних застосунків для MS DOS часто доводилося зберігати в пам'яті відразу кілька копій текстових екранів. В даний час особливою популярністю користуються програми з графічним інтерфейсом, тому текстові відеосторінки втратили свою значимість. За замовчуванням використовується нульова відеосторінка.

Атрибути. Кожному символу, що відображається на екрані, призначається спеціальний байт атрибутів, який визначає його колір, а також колір розташованого за ним фону.

Кожній позиції екрану відповідає один символ і один атрибут кольору. Значення атрибута зберігається в окремому байті, який у відеопам'яті розташований відразу ж за символом. На рис. 1 показано розташування у відеопам'яті трьох символів "ABC", а також їх атрибутів.

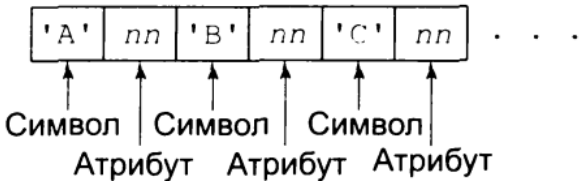

Рис. 1. Розташування символів і атрибутів у відеопам'яті

Приклад 1. Пряме програмування відеопам'яті

```
; Налаштуємо сегментний регістр ES на сторінку 0 відеопам'яті
mov AX, 0B800h  ;(1) Сегментна адреса відеопам'яті
mov ES, AX      ;(2) Завантажимо його в ES
mov BX, 0       ;(3) Зміщення до початку екрану
mov SI, 80*2*12+40*2  ;(4) Зміщення до центру екрану
mov DI, 80*2*25-2     ;(5) Зміщення до кінця екрану
mov word ptr ES:[BX], 0F0lh;  ( 6) Фізіономію на екран
mov word ptr ES:[SI], 3130h; (7) Нулик на екран
mov word ptr ES:[DI], 0E40Fh; (8) Зірку на екран
mov AH, 01h                   ; (9) Зупинка програми
int 21h                       ; {10) для спостереження результату
```

Зміна кольорів

Змішування основних кольорів

Колір кожного пікселя зображення визначається значенням струму трьох незалежних променів електронно-променевої трубки монітора: червоного, зеленого і синього. Існує ще один, четвертий канал керування кольором, який змінює загальну інтенсивність (тобто яскравість) всіх пікселів. Таким чином, значення всіх доступних кольорів в текстовому режимі можна визначити у вигляді одного 4-бітового двійкового числа, заданого в наступному форматі: I = інтенсивність, R = червоний, G = зелений, B = синій. На рис. 2 показано, як формується піксель білого кольору на екрані.

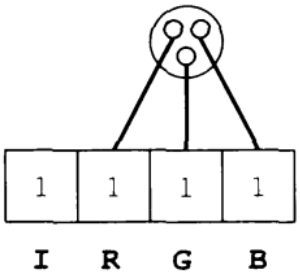

Рис. 2. Формування пікселя білого кольору на екрані

Щоб створити піксель нового кольору, необхідно змішати в різній пропорції три основних кольори, як показано в табл. 1. Змінюючи біт інтенсивності, можна керувати яскравістю нового кольору, в результаті чого буде трохи змінюватися його відтінок.

Табл. 1

Кодування кольорів тексту за допомогою чотирьох бітів

IRGB	Колір
0000	Чорний
0001	Синій
0010	Зелений
0011	Ціан
0100	Червоний
0101	Пурпурний
0110	Коричневий
0111	Світло-сірий
1000	Сірий
1001	Яскраво-синій
1010	Салатовий
1011	Блакитний
1100	Яскраво-червоний
1101	Фіолетовий
1110	Жовтий
1111	Білий

Байт атрибутів

У текстовому режимі колір кожної символьної комірки визначається значенням спеціального байта-атрибута, який складається з кодів двох 4-бітових кольорів: фону і символу (рис. 3).

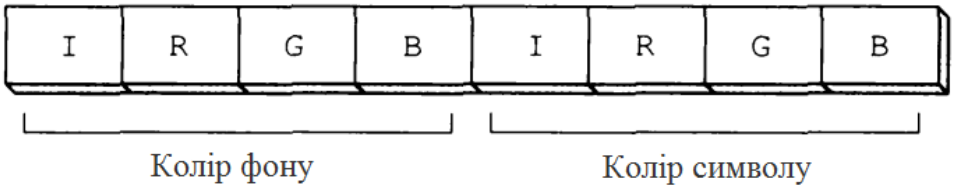

Рис. 3. Формат байта атрибутів

Миготіння. Якщо у відеоадаптері активізований режим миготіння, то старший біт байта атрибутів керує миготінням символу. Якщо встановити цей біт, символ почне блимати на екрані (рис. 4).

Рис. 4. Формат байта атрибутів при активізованому байті миготіння

Якщо режим миготіння відеоадаптера активізований, то для фону можна задати тільки кольори, в яких крайній лівий біт є нульовим (тобто чорний, синій, зелений, ціан, червоний, пурпуровий, коричневий і світло-сірий). За замовчуванням при завантаженні системи MS DOS для всіх символів екрану встановлюється значення байта атрибутів, рівне 00000111b (тобто світло-сірі символи на чорному фоні).

Визначення байта атрибутів. Для визначення байта атрибутів, який складається зі значення двох колірних констант (фону і символу), можна скористатися операторами асемблера SHL та OR. Константу, яка визначає колір фону, треба зсунути на 4 біти вліво за допомогою оператора SHL, а потім додати до неї за допомогою оператора OR константу, яка визначає колір символу. Як приклад в наведеному нижче фрагменті коду визначається константа для байта атрибутів, який відповідає сірим буквам на синьому фоні:

```
BLUE = 1
LIGHT_GRAY = 111b
mov bh, (BLUE SHL 4) OR LIGHT_GRAY          ; 00010111b
```

А в цьому фрагменті визначаються білі символи на червоному фоні:

```
WHITE = 1111b
RED = 100b
mov bh, (RED SHL 4) OR WHITE                ; 01001111b
```

А ось як можна визначити сині символи на коричневому фоні:

```
BLUE = 1
BROWN = 110b
mov bh, ((BROWN SHL 4) OR BLUE)             ; 01100001
```

Функції для роботи з відео переривання INT 10h

В табл. 2 наведені найбільш вживані функції переривання INT 10h.

Табл. 2. Найбільш вживані функції переривання INT 10h

Функція	**Опис**
00h	Встановити текстовий або графічний режим відео з заданим номером
01h	Встановити форму і розмір курсору, вказавши номери початкового і кінцевого рядка відображення
02h	Перемістити курсор в зазначену позицію на екрані
03h	Визначити положення і розмір курсору
04h	Визначити положення і стан світового пера (застаріла)
05h	Відобразити на екрані відеосторінку із зазначеним номером (використовується рідко)
06h	Прокрутити вікно поточної відеосторінки вгору на вказану кількість рядків, замінюючи витіснені рядки пробілами
07h	Прокрутити вікно поточної відеосторінки вниз на вказану кількість рядків, замінюючи витіснені рядки пробілами
08h	Прочитати з екрану символ і його байт атрибутів, який визначається поточним положенням курсора
09h	Вивести на екран символ і його байт атрибутів в позицію, яка визначається поточним положенням курсора
0Ah	Вивести на екран тільки символ (без його байта атрибутів) в позицію, яка визначається поточним положенням курсора
0Bh	Встановити палітру кольорів відеоадаптера (використовується рідко)
0Ch	Вивести на екран піксель в графічному режимі
0Dh	Визначити колір пікселя в зазначеній позиції екрану
0Eh	Вивести символ на екран в графічному режимі і перемістити курсор на одну позицію вправо (використовується рідко)
0Fh	Визначити параметри поточного режиму відео
10h	Перемкнути режим миготіння відеоадаптера на режим керування інтенсивністю і навпаки
13h	Вивести рядок на екран в режимі емуляції телетайпа

Функція 13h переривання INT 10h дозволяє вивести текстовий рядок на екран з вказаної у вигляді номера рядка і стовпця позиції. У рядку, крім символів, можуть бути вказані і байти атрибутів.

INT 10h, функція 13h

Опис - виводить рядок на екран в режимі емуляції телетайпа.

Параметри:

AH = 13h

AL = режим записування
BH = номер відеосторінки
BL = байт атрибутів (якщо AL = 00h або 01h)
CX = довжина рядка (лічильник символів)
DH, DL = номер рядка і стовпця
ES:BP = адреса рядка в формі "сегмент-зміщення"

В регістрі AL можна задати наступні режими записування:

• 00h - в заданому рядку вказані тільки ASCII-коди символів; після виведення на екран поточне положення курсора не змінюється; в регістрі BL вказаний байт атрибутів;

• 01h - в заданому рядку вказані тільки ASCII-коди символів; після виведення на екран поточне положення курсора коригується з урахуванням довжини рядка; в регістрі BL вказано байт атрибутів;

• 02h - в заданому рядку вказані ASCII-коди символів, слідом за якими упереміш розташовані байти атрибутів; після виведення на екран поточне положення курсора не змінюється;

• 03h - в заданому рядку вказані ASCII-коди символів, слідом за якими упереміш розташовані байти атрибутів; після виведення на екран поточне положення курсора коригується з урахуванням довжини рядка.

Приклад

```
.data
colorString   BYTE   'A', 1Fh, 'B', 1Ch, \
                     'C', 1Bh, 'D', 1Ch
row           BYTE   10
column        BYTE   20
.code
mov ax, SEG colorstring          ; Ініціалізуємо сегмент ES
mov es, ax
mov ah, 13h                      ; Функція виведення рядка
mov al, 2                        ; Режим записування
mov bh, 0                        ; Номер відеосторінки
mov cx, (SIZEOF colorString) / 2   ; довжина рядка
mov dh, row                      ; Початковий рядок
mov dl, column                   ; Початковий стовпець
mov bp, OFFSET colorString   ; Адреса рядка
int 10h
```

Приклад використання цієї функції наведено в програмі Colorst2.asm.

```
TITLE Color String Example            (ColorSt2.asm)
Comment !
Demonstrates INT 10h function 13h, which writes
a string containing embedded attribute bytes to the
video display. The write mode values in AL are:
0 = string contains only character codes; cursor not
   updated after write, and attribute is in BL.
1 = string contains only character codes; cursor is
   updated after write, and attribute is in BL.
2 = string contains alternating character codes and
   attribute bytes; cursor position not updated
```

```
    after write.
3 = string contains alternating character codes and
    attribute bytes; cursor position is updated
    after write.

!

.model small
.386
.stack
.data
colorString BYTE 'A',1Fh,'B',1Ch,'C',1Bh,'D',1Ch
row    BYTE  10
column BYTE  20
.code
extrn Clrscr:proc
main PROC
        mov  ax data
        mov  ds, ax
        call ClrScr
        mov  ax, SEG colorString
        mov  es, ax
        mov  ah, 13h        ; write string
        mov  al, 2          ; write mode
        mov  bh, 0          ; video page
        mov  cx, (SIZEOF colorString) / 2     ; string length
        mov  dh, row                ; start row
        mov  dl, column             ; start column
        mov  bp, OFFSET colorString  ; ES:BP points to string
        int  10h
        mov  ah, 2          ; home the cursor
        mov  dx, 0
        int  10h
        .exit
main ENDP
END main
```

Приклад: програма відображення кольорового рядка ColorStr.asm

Програма ColorStr.asm виводить на термінал текстовий рядок, причому для кожного символу використовується свій колір. В системі Windows цю програму потрібно запускати в повноекранному режимі, щоб побачити миготіння символів. За замовчуванням у програмі активізується режим миготіння, але можна закоментувати виклик функції EnableBlinking і подивитися, як буде виглядати текст на темно-сірому тлі.

TITLE Приклад відображення кольорового рядка (ColorStr.asm)

```
INCLUDE Irvinel6.inc
. data
ATTRIB_HI = 10000000b
string BYTE "ABCDEFGHIJKLMOP"
color BYTE (black SHL 4) OR blue
. code main PROC
mov ax, @data
mov ds, ax
call ClrScr
call EnableBlinking             ;  Це можна закоментувати
mov cx, SIZEOF string
mov si, OFFSET string
L1:
push cx                         ;  Збережемо лічильник циклу
mov ah, 9                       ;  Вивести символ і байт атрибутів
mov al, [si]                    ;  Завантажимо символ, що виводиться
mov bh, 0                       ;  Відеосторінка 0
mov bl, color                   ;  Байт атрибутів
or bl, ATTRIB_HI                ;  Встановимо старший біт, який
                                ;  керує миготінням/яскравістю
mov cx, 1                       ;  Виведемо один символ
int 10h
mov cx, 1                       ;  Перемістимо курсор на одну позицію
call AdvanceCursor              ;  вправо
inc color                       ;  Значення наступного атрибута кольору
inc si                          ;  Адреса наступного символу
pop cx                          ;  Відновимо лічильник циклу
loop L1
call CrLf
exit
main ENDP
; -----------------------------------------------------------------------------
EnableBlinking PROC
; Активізує режим миготіння відеоадаптера, який визначається
; значенням старшого біта байта атрибутів.
; У середовищі Windows працює тільки в повноекранному режимі.
; Передається: нічого
; Повертається: нічого
; -----------------------------------------------------------------------------
push ax
push bx
mov ax, 1003h                   ;  Активізуємо режим миготіння
mov bl, 1
int 10h
pop bx
pop ax
ret
EnableBlinking ENDP
```

```
; ---------------------------------------------------------------------------
AdvanceCursor PROC
; Переміщує курсор на n позицій вправо.
; Передається: CX = кількість позицій
; Повертається: нічого
; Процедуру AdvanceCursor можна використовувати в будь-якій
; програмі, в якій виводиться текст на термінал за допомогою
; функцій переривання INT 10h.
; ---------------------------------------------------------------------------
pusha
L1:
push cx                 ;  Збережемо лічильник циклу
mov ah, 3               ;  Визначимо поточну позицію курсора
mov bh, 0               ;  Вона повертається в DH, DL
int 10h                 ;  Змінюється регістр CX!
inc dl                  ;  Збільшимо на 1 значення стовпця
mov ah, 2               ;  Встановимо положення курсора
int 10h
pop cx                  ;  Відновимо лічильник циклу
loop L1                 ;  Наступна позиція курсора
popa
ret
AdvanceCursor ENDP
END main
```

Графічний режим роботи відеоадаптера

Відображення графічних зображень

За допомогою функції 0Ch переривання INT 10h можна просто вивести на екран прості графічні об'єкти, такі як точки та лінії. Для простоти спочатку розглянемо процес відображення пікселів на екрані монітора за допомогою цієї функції, а потім покажемо, як можна вивести пікселі на екран монітора шляхом прямого записування даних у відеопам'ять. Перш ніж виводити пікселі на екран, відеоадаптер потрібно перемикнути на один із стандартних графічних режимів роботи, описаних у табл. 3. Нагадаємо, що відеорежим встановлюється за його номером за допомогою функції 00h переривання INT 10h.

Таблиця 3. Список графічних відеорежимів переривання INT 10h

Відеорежим	Роздільна здатність (стовпці х рядки)	Кількість кольорів
06h	640x200	2
0Dh	320x200	16
0Eh	640x200	16
0Fh	640x350	2
10h	640x350	16

11h	640x480	2
12h	640x480	16
13h	320x200	256
6Ah	800x600	16

Координати пікселів. Для кожного графічного відеорежиму встановлюється певна роздільна здатність екрана монітора, що виражається в максимальній кількості точок по горизонталі та вертикалі XMax, YMax відповідно, які здатний відтворити монітор. Початок координат, тобто точка з координатами x = 0, y = 0, міститься у верхньому лівому кутку екрана. Точка з максимально можливими значеннями координат (x = XMax - 1, y = YMax - 1) міститься в нижньому правому кутку екрана.

Функції переривання INT 10h для роботи з пікселями

Виведення пікселя на екран (функція 0Ch)

Цю функцію можна використовувати лише після того, як відеоадаптер переведено у графічний режим. Вона призначена для виведення одного пікселя на екран. Слід зазначити, що ця функція працює досить повільно, тому при послідовному виведенні за її допомогою великої кількості пікселів робота програми помітно сповільнюється. Саме з цієї причини в більшості графічних програм використовується прямий доступ до відеопам'яті, адреса якої обчислюється залежно від координат пікселя, кількості кольорів, що підтримуються, і горизонтальної роздільної здатності. Опис параметрів цієї функції наведено нижче.

INT l0h, функція 0Ch	
Опис	Виводить піксель на екран
Параметри	AH = 0Ch AL = значення пікселя BH = номер відеосторінки CX = значення горизонтальної координати (X) DX = значення вертикальної координати (Y)
Щ о повертається	Нічого

Приклад	mov ah, 0Ch
	mov al, pixelValue
	mov bh, videoPage
	mov cx, x_coord
	mov dx, y_coord
	int 10h

Примітка	Працює лише у графічному режимі. Значення пікселя залежить від кількості кольорів, що підтримуються на екрані, і становить: 0—1 для двоколірних режимів, 0—15 для 16-колірних режимів. Якщо встановити 7-й біт регістра AL, нове значення пікселя буде скомбіновано з поточним значенням (з тим, що відображається на екрані) за допомогою операції XOR. Це дозволяє стерти піксель з екрану

Приклад програми - програма DrawLine

Ця програма переводить відеоадаптер у графічний режим за допомогою функції 00h переривання INT 10h, а потім креслить на екрані пряму горизонтальну лінію.

В середовищі Microsoft Windows ця програма повинна запускатися тільки в повноекранному режимі. Нижче наведено повний лістинг програми:

```
TITLE Програма DrawLine (Pixell.asm)
; Рисує пряму лінію за допомогою виклику функції OCh переривання INT 10h.

INCLUDE Irvinel6.inc
; Константи відеорежимів
Mode_06 = 6             ; 640 x 200, 2 кольори
Mode_0D = 0Dh           ; 320 x 200, 16 кольорів
Mode_0E = 0Eh           ; 640 x 200, 16 кольорів
Mode_0F = 0Fh           ; 640 x 350, 2 кольори
Mode_10 = l0h           ; 640 x 350, 16 кольорів
Mode_11 = 11h           ; 640 x 480, 2 кольори
Mode_12 = 12h           : 640 x 480, 16 кольорів
Mode_13 = 13h           ; 320 x 200, 256 кольорів
Mode_6A = 6Ah           ; 800 x 600, 16 кольорів
.data
saveMode   BYTE ?           ; Збережений поточний відеорежим
currentX   WORD 100         ; Номер стовпця (координата X)
currentY   WORD 100         ; Номер рядка (координата Y)
color      BYTE 1           ; Стандартне значення кольору
; У двоколірних відеорежимах color=l означає білий колір
; У 16-колірних відеорежимах color=l означає синій колір
.code
 main PROC
 mov ax, @data
 mov ds, ax
; Збережемо номер поточного відеорежиму
 mov ah, 0Fh
 int l0h
 mov saveMode, al
; Перемкнемося в графічний режим
 mov ah, 0                  ; Функція встановлення відеорежиму
```

```
 mov al, Mode_11
 int l0h
; Рисуємо пряму лінію
LineLength = 100
 mov dx, currentY
 mov cx, LineLength            ; Лічильник циклу
L1:
 push cx
 mov ah, 0Ch                   ; Функція виведення пікселя
 mov al, color                 ; Колір пікселя
 mov bh, 0                     ; Відеосторінка 0
 mov cx, currentX
 int l0h
 inc currentX
; inc color                     ; Розкоментуйте під час роботи
                               ; в багатоколірному режимі
 pop cx
 Loop L1

; Чекаємо натиснення будь-якої клавіші
 mov ah,10h
 int 16h
; Відновимо попередній відеорежим
 mov ah, 0                     ; Функція встановлення відеорежиму
 mov al,saveMode               ; Номер збереженого відеорежиму
 int l0h
 exit
 main ENDP
END main
```

Відображення графіки за допомогою безпосереднього записування у відеопам'ять

Основний недолік описаного способу виведення пікселів на екран за допомогою однієї з функцій переривання INT 10h полягає у дуже повільній швидкості виведення. Справа в тому, що для виведення одного пікселя щоразу доводиться викликати переривання INT 10h, на оброблення якого операційна система витрачає багато часу. Набагато ефективніший метод відображення пікселів на екрані полягає у безпосередньому їх записуванні у відеопам'ять. Ця методика зазвичай називається прямим доступом до відеопам'яті.

Відеорежим 13h: 320x200, 256 кольорів

Найзручнішим відеорежимом під час використання прямого доступу до відеопам'яті є режим 13h. При його встановленні кожен піксель екрану займає один байт у відеопам'яті, яка подається у вигляді двовимірного масиву байтів, що відповідають рядкам та стовпцям зображення. Піксель, розташований у лівому верхньому кутку екрана, відповідає першому байту масиву, що має нульове зміщення. У першому рядку екрану міститься 320 пікселів, які

відповідають першим 320 байтам масиву відеопам'яті. Наступні 320 байтів масиву відповідають пікселям другого рядка екрану тощо. Останньому байту масиву відповідає піксель, розташований у правому нижньому куті екрану. Чому кожному пікселю відведено цілий байт? Вся справа в тому, що для подання 256 різних кольорів потрібно 256 значень цілих чисел, що відповідає 8 бітам або одному байту.

Команда OUT. Керування роботою відеоадаптера здійснюється програмно за допомогою команд OUT (Output to port або Виведення в порт). З їх допомогою встановлюється колірний режим, роздільна здатність екрану та палітра кольорів. Перед виконанням команди OUT в регістр DX завантажується 16-розрядна адреса порту, а в регістр AL - значення, що виводиться в порт. До прикладу, регістр, який керує палітрою кольорів, має адресу порту 3C8h. У наведеному нижче фрагменті програми в цей порт виводиться значение 20h.

```
mov dx, 3C8h        ; Адреса порту
mov al, 20h         ; Значення, що виводиться
out dx, al          ; Команда записування в порт
```

Індекси кольору. Одна з особливостей використання відеорежиму номер 13h полягає в тому, що цілі числа, що визначають один із 256 кольорів пікселів, не прямо, а опосередковано впливають на їх колір. Насправді ці числа є індексами, за значенням яких вибирається реальний колір зі спеціальної таблиці кольорів, яка називається палітрою (palette). Кожен елемент палітри кольорів складається із трьох цілих чисел, значення яких міститься в діапазоні 0-63. Ці числа визначають інтенсивність кожного із трьох променів: червоного, синього та зеленого (RGB). Нульовий елемент кольорової палітри визначає колір фону екрана.

Таким чином, використовуючи палітру кольорів, можна створити 262144, або 64^3 різних кольорів. Однак тільки 256 з них можуть одночасно відображатися на екрані. Тим не менш, у програмі можна швидко перемикнути палітру кольорів і таким чином впливати на кольори, що відображаються на екрані.

Кольори RGB. При формуванні RGB-кольорів використовується адитивна колірна модель, при якій яскраво-білий колір отримується шляхом змішування всіх трьох кольорів в рівних пропорціях. Крім адитивної, використовується також субтрактивна модель кольору, яка використовується при кольоровому друкуванні на папері. У ній кольори утворюються шляхом віднімання з яскраво-білого кольору одного з основних кольорів.

При використанні адитивної колірної моделі чорний колір виходить за повної відсутності колірних складових. Для отримання білого кольору необхідно встановити максимальну інтенсивність основних колірних складових, тобто. присвоїти їм у палітрі кольорів значення 63. Якщо одночасно змінювати значення складових всіх трьох кольорів, то отримаються різні відтінки сірого кольору, як показано в табл. 4.

Таблиця 4. Одержання відтінків сірого кольору в адитивній колірній моделі

Червоний (R)	Зелений (G)	Синій (B)	Колір
0	0	0	Чорний
20	20	20	Темно-сірий

35	35	35	Сірий
50	50	50	Світло-сірий
63	63	63	Білий

Для отримання чистих кольорів необхідно значення всіх складових кольорів, крім однієї, встановити в нуль. Щоб отримати світлі відтінки будь-якого кольору, необхідно збільшити в рівних пропорціях складові двох інших кольорів. У табл. 5 показано, як можна отримати різні відтінки червоного кольору.

Таблиця 5. Отримання відтінків червоного кольору

Червоний (R)	Зелений (G)	Синій (B)	Колір
63	0	0	Яскраво-червоний
10	0	0	Темно-червоний
30	0	0	Червоний
63	40	40	Рожевий

Відтінки двох інших кольорів (синього та зеленого) отримуються за аналогією з червоним. Для отримання інших кольорів, таких як фіолетовий або жовтий, доведеться змішати в різних пропорціях основні кольори, як показано в таблиці. 6.

Таблиця 6. Отримання різних кольорів

Червоний (R)	Зелений (G)	Синій (B)	Колір
0	30	30	Блакитний/лазуровий
30	30	0	Жовтий
30	0	30	Фіолетовий
40	0	63	Бузковий/ліловий

Програма прямого виведення даних у відеопам'ять

Ця програма виводить на екран у графічному режимі 13h 10 пікселів з використанням прямого доступу до відеопам'яті.

```
TITLE Програма прямого доступу до відеопам'яті (Model3.asm)
INCLUDE IrvinelG.inc
.data
saveMode   BYTE      ?          ; Збережений відеорежим
xVal       WORD      ?          ; Координата X
yVal       WORD      ?          ; Координата Y
; В основній процедурі встановлюється відеорежим 13h, колір фону,
; виводиться кілька пікселів на екран, а потім відновлюється відеорежим.
.code
main PROC
 mov ax, @data
```

```
 mov ds, ax
 call SetVideoMode
 call SetScreenBackground
 call Draw_Some_Pixels
 call RestoreVideoMode
 exit
main ENDP
SetScreenBackground PROC
; Встановлює колір екрану фону.
; Як колір фону використовується нульовий елемент палітри кольорів.
 mov dx, 3c8h            ; Порт вибору палітри кольорів (3C8h)
 mov al, 0               ; Встановимо індекс палітри кольорів
 out dx, al
; Для керування палітрою кольорів використовуються два регістри виводу.
; Значення, записане в порт 3C8h, визначає номер елемента палітри кольорів,
; який планується змінити.
; Після записування індексу в порт 3C8h, в порт 3C9h записують значення
; кольорів.
; Встановимо темно-синій фон екрану
 mov dx, 3c9h            ; Значення кольорів виводяться  в порт 3C9h
 mov al, 0               ; Значення червоного кольору
 out dx, al
 mov al, 0               ; Значення зеленого кольору
 out dx,al
 mov al, 35              ; Значення синього кольору
 out dx, al              ; (інтенсивність 35/63)
ret
SetScreenBackground ENDP
SetVideoMode PROC
; Зберігає значення поточного відеорежиму,
; перемикає відеоадаптер у режим 13h і завантажує
; в регістр ES сегментну адресу відеобуфера.
 mov ah, 0Fh             ; Визначимо поточний відеорежим
 int 10h
 mov saveMode, al        ; Збережемо його
 mov ah, 0               ; Встановимо новий відеорежим
 mov al, 13h             ; номер 13h
int 10h
push 0A000h              ; Сегментна адреса відеобуфера
pop es                   ; ES = A000h (відеосегмент)
ret
SetVideoMode ENDP
RestoreVideoMode PROC
; Чекає натиснення будь-якої клавіші, а потім відновлює
; початковий відеорежим
 mov ah, 10h             ; Чекаємо натиснення клавіші
 int 16h
 mov ah, 0               ; Відновимо старий відеорежим
 mov al, saveMode
```

```
 int 10h
 ret
RestoreVideoMode ENDP

Draw_Some_Pixels PROC
; Встановлює колір окремого елемента палітри кольорів і
; креслить на екрані кілька пікселів
; Змінимо колір елемента палітри, що визначається індексом 1,
; на білий (63,63,63)
 mov dx, 3c8h          ; Порт індексу палітри кольорів (3C8h)
 mov al, 1             ; Встановимо індекс 1
 out dx, al
 mov dx, 3c9h          ;Значення кольорів виводяться у порт 3C9h
 mov al, 63            ; Червоний колір
 out dx, al
 mov al, 63            ; Зелений колір
 out dx, al
 mov al, 63            ; Синій колір
 out dx, al
; Обчислимо зміщення першого пікселя у відеобуфері.
; Воно характерне для поточного відеорежиму 13h, роздільна здатність якого
; складає 320x200.
 mov xVal, 160         ; Середина екрану
 mov yVal, 100
 mov ax, 320           ; Кількість пікселів у рядку
 mul yVal              ; множимо на координату Y,
 add ax, xVAl          ; і додаємо координату X.
; Помістимо значення індексу кольору у відеопам'ять.
 mov cx, 10            ; Виведемо 10 пікселів
 mov di, ax            ; в AX – зміщення відеобуфера
; Рисуємо пряму завдовжки 10 пікселів.
DPI:
 mov BYTE PTR es:[di], 1 ; Записуємо індекс кольору
; За замовчуванням при зверненні до пам'яті через регістр DI процесор
; використовує сегментний регістр DS. У нашому випадку ми використовували
; команду заміни сегмента (es:[di]), щоб повідомити процесору про те, що при
; зверненні до пам'яті замість регістра DS потрібно використовувати регістр ES.
; Нагадаємо, що в регістрі ES зберігається сегментна адреса відеобуфера.
 add di, 5             ; Зсунемось праворуч на 5 пікселів
 Loop DPI
 ret
Draw_Some_Pixels ENDP
END main
```

Реалізувати нашу програму дость просто, оскільки всі пікселі розміщуються в одному рядку. Якщо нам потрібно було б розмістити пікселі в одному вертикальному стовпці, то до значення регістра DI потрібно було б додати значення 320, щоб перейти на наступний рядок пікселів. Щоб накреслити діагональну лінію зі зсувом пікселів у сусідніх рядках на одиницю, до регістру DI потрібно додати значення 361.

Порядок виконання роботи

1. Опишіть рядки символів, в яких вкажіть прізвище, ім’я, по батькові.

2. Сформуйте байти атрибутів (різні) для кожного символу в кожному рядку символів.

3. Очистіть екран.

4. В текстовому режимі початковий номер рядка на екрані визначається як остача від ділення номера в списку групи на 10.

5. Початковий номер стовпця на екрані визначається як сума номера групи і номера в списку групи.

6. Шляхом безпосереднього записування тексту в першу сторінку текстової відеопам'яті виведіть рядок символів з прізвищем у початковий рядок, починаючи з початкового стовпця.

7. Рядок символів з іменем виведіть в рядок, номер якого дорівнює початковому+2, починаючи з стовпця, номер якого дорівнює початковому+3.

8. Рядок символів з по батькові виведіть в рядок, номер якого дорівнює початковому+6, починаючи з стовпця, номер якого дорівнює початковому+8.

9. Зробіть копію екрану.

10. В графічному режимі 13h побудуйте прямокутник, ліва верхня вершина якого розміщується в рядку, номер якого дорівнює кількості літер у прізвищі, і в стовпці, номер якого дорівнює кількості літер в по батькові. Довжина горизонтальної сторони прямокутника дорівнює потроєному номеру групи+номер в списку групи. Довжина вертикальної сторони прямокутника дорівнює подвоєному номеру групи+номер в списку групи. Колір прямокутника виберіть за остачею від ділення номера групи на 3: 0 – червоний; 1 – зелений; 2 – синій.

11. Перевірте результат роботи програми.

12. Зробіть копію екрану.

13. У звіті наведіть текст програми.

Див. книгу: Кип Ирвин Язык ассемблера для процессоров Intel 4-е изд 2005.