МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
"ХАРЬКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

А. І. ПОВОРОЗНЮК, М. В. МЕЗЕНЦЕВ, О. А. ПОВОРОЗНЮК

АРХІТЕКТУРА КОМП'ЮТЕРІВ

Лабораторний практикум

Затверджено
редакційно-видавничою
радою НТУ «ХПІ»,
протокол № 2 від 27.06.2024 р.

Харків
НТУ "ХПІ"
2024

УДК 004.3(075)
П 42

Рецензент: В.В. Усик, к.т.н., професор кафедри «Мультимедійні та інтернет технології і системи» НТУ «ХПІ»

Поворознюк А.І.
П42 Архітектура комп'ютерів: лабораторний практикум / А.І. Поворознюк, М.В. Мезенцев, О.А. Поворознюк – Харків: НТУ «ХПІ», 2024 – 131 с.

У цьому виданні наведено методичні вказівки до лабораторних робіт 1–15, які включають роботи з визначення конфігурації ПК, керування таймером та звуком, клавіатурою, накопичувачами на магнітних дисках, керування відеомонітором у текстовому та графічному режимах.

Тематика лабораторних робіт змістовно співпадає з цільовою задачею курсу та відповідає навчальній програмі вивчення дисципліни. Кожне заняття містить теоретичний матеріал, вирішення типових задач по данній темі та індивідуальні завдання.

Призначено для студентів усіх спеціальностей бакалаврату 123 "Комп'ютерна інженерія", денної та заочної форми навчання, а також може бути корисним як для початківців, так і для досвідчених програмістів при створенні ефективного програмного забезпечення.

Іл. 39, Табл. 32, Бібліогр. 7 назв.

УДК 004.3(075)
ББК 32.973-02я7

© А. І. _ Поворознюк ,
М. В. Мезенцев ,
О. А. Поворознюк,
НТУ «ХПІ» 2024

ЗМІСТ

ВСТУП

Лабораторний практикум до лабораторних робіт з курсу "Архітектура комп'ютерів" містить опис 15 двогодинних лабораторних робіт.

Лабораторні роботи спрямовані на використання IBM PC сумісних ПЕОМ, використовують мікропроцесори Intel та його клони.

Мета лабораторного практикуму

Метою лабораторних робіт є закріплення практичних навичок керування модулями ПЕОМ на низькому рівні шляхом програмування адаптерів пристроїв на рівні портів. Переривання BIOS та API -функції використовуються лише у випадках, коли керування на рівні портів вимагає складних алгоритмів управління із завданням часових затримок.

Так як сучасні ОС обмежують, а в багатьох випадках забороняють, доступ до ресурсів ПК, то для забезпечення необхідного рівня доступу створюється віртуальна машина з Windows 98, яка забезпечує необхідні дії.

Організація та проведення лабораторних робіт

Студенти групи об'єднуються у бригади по 2-3 особи, які працюють на закріпленому комп'ютері.

Кожен студент отримує індивідуальне завдання відповідно до номера у журналі та оформлює звіт з лабораторної роботи.

Виконання лабораторної роботи передбачає попереднє вивчення відповідного розділу курсу та методичних вказівок до заданої роботи.

Для допуску до виконання лабораторної роботи студент повинен ознайомитись з темами для опрацювання та попередньо написати текст програми, відповідно до індивідуального завдання.

Текст програми складається однією з мов програмування Турбо-Сі або Турбо-Паскаль з урахуванням глибини знань студентами конкретної мови.

При складанні програм користуватись рекомендаціями пункту "Особливості програмування" для цієї лабораторної роботи.

Протягом виконання лабораторної роботи студент повинен відповісти на контрольні запитання щодо попередньої лабораторної роботи.

До лабораторної роботи не допускаються студенти, які не здали більше двох лабораторних робіт.

Пропущені лабораторні роботи виконуються наприкінці семестру.

У процесі виконання лабораторних робіт слід обмежити переміщення у лабораторії.

Лабораторна робота 1

Тема: Ознайомлення з технологіями віртуальних машин.

Мета роботи: Отримання практичних навичок встановлення та налаштування віртуальних машин.

1. 1. Опис роботи.

Усі лабораторні роботи побудовані на отриманні інформації або програмуванні апаратури, що входить до складу ПЕОМ, безпосередньо через порти вводу/виводу, переривання чи ділянки ОЗП. Сучасні операційні системи (ОС) не дають такої можливості, тому для виконання лабораторних робіт слід використовувати ті ОС, які це дозволяють або використовувати програми віртуалізації відповідних ОС. Наприклад, можна використовувати програмні продукти: DOSBox, Virtual Box, VMWare Workstation та ін. Далі буде описано налаштування VMWare Workstation для виконання лабораторних робіт.

Після встановлення VMWare Workstation (версія 8) необхідно вибрати пункт меню «Файл/Нова віртуальна машина» або натиснути комбінацію Ctrl+N. На першій сторінці майстра створення віртуальної машини необхідно вибрати пункт «Звичайний». На другій сторінці необхідно вказати шлях до інсталяційного диска гостьової ОС (у даному випадку буде використовуватися Windows 98) або до файлу - образу диска з ОС (рис. 1.1).

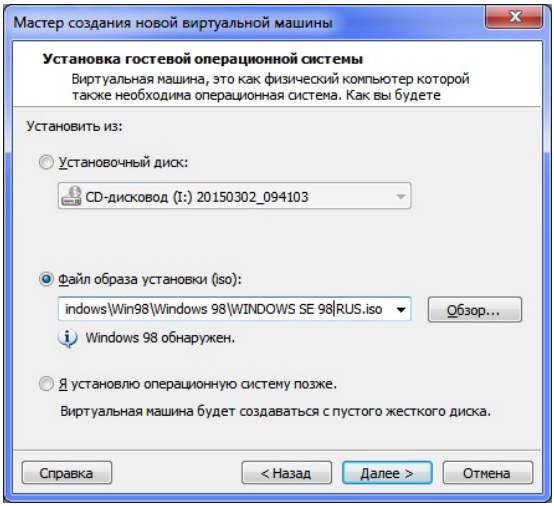


Рис. 1.1 - Вибір настановного диска гостьової ОС

На наступному вікні майстра потрібно вибрати ім'я для гостьової ОС та каталог для її розміщення на диску (рис. 1.2).

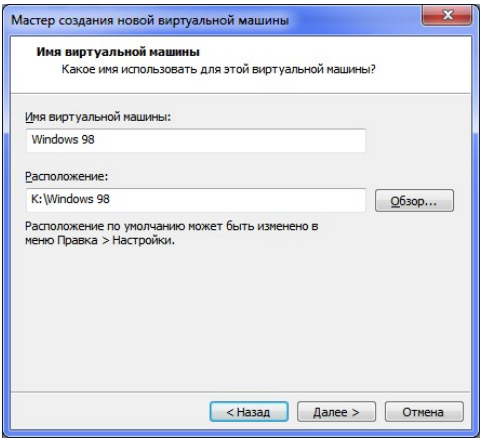


Рис. 1.2 - Вибір каталогу розміщення файлів віртуальної машини

Після цього визначити розмір диска для гостьової ОС та варіант його зберігання (одним чи кількома файлами) (рис. 1.3).

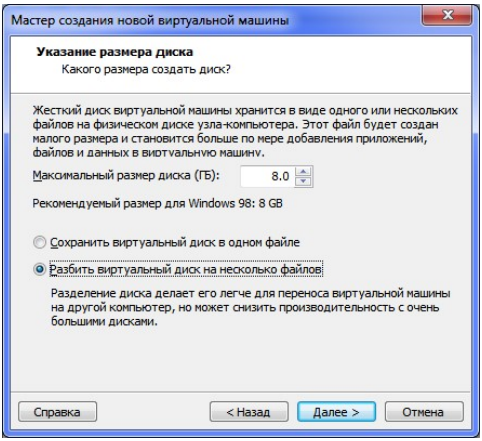


Рис. 1.3 - Вибір параметрів диска для гостьової ОС

Після цього процес створення віртуальної машини завершено (рис. 1.4).

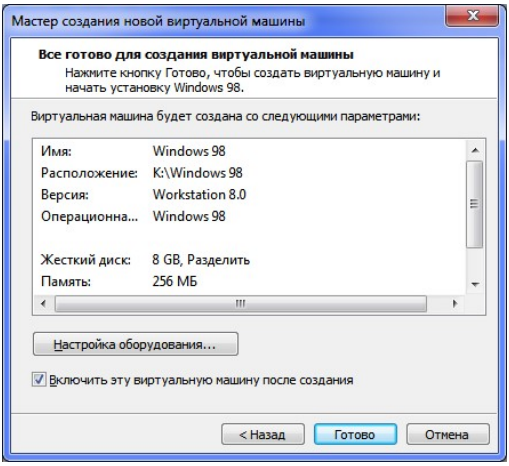


Рис. 1.4 - Закінчення створення віртуальної машини

Можна налаштувати ще обладнання (натиснувши на кнопку «Налаштування обладнання…» рис. 1.4), де можна вказати стандартні параметри (рис. 1.5), такі як: обсяг ОЗП, параметри процесора, тип CD-приводу, мережний адаптер, USB-контролер, звукову карту та дисплей або встановити додаткове обладнання (натиснувши кнопку «Додати…» на рис. 1.5) . Після цього , натиснувши кнопку «Готово» (рис. 1.4), необхідно встановити гостьову ОС.

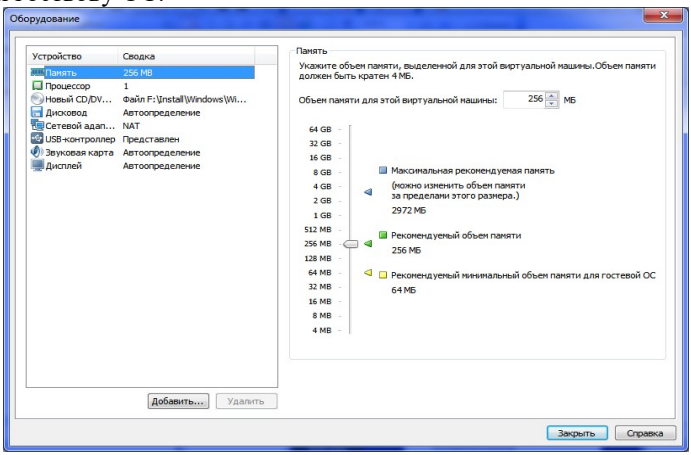


Рис. 1.5 - Налаштування обладнання гостьової ОС

Далі, якщо було обрано опцію «Включити віртуальну машину після створення» (рис. 1.4), розпочнеться процес встановлення ОС. Цей процес нічим не відрізняється від процесу встановлення ОС на реальну ПЕОМ. Розглянемо налаштування ОС Windows 98.

Процес встановлення починається з вибору пристрою завантаження. В даному випадку необхідно вибрати опцію 2: Boot from CD-ROM (Рис. 1.6).

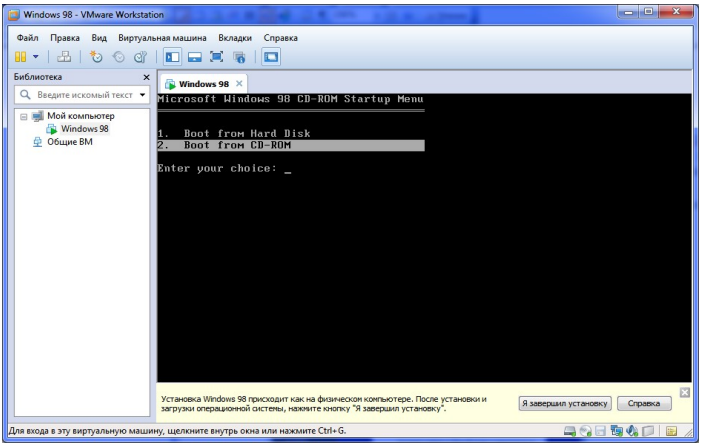


Рис. 1.6 – Вибір варіанта завантаження

Далі необхідно вибрати 1 пункт: Start Windows 98 Setup from CD - ROM (рис. 1.7).

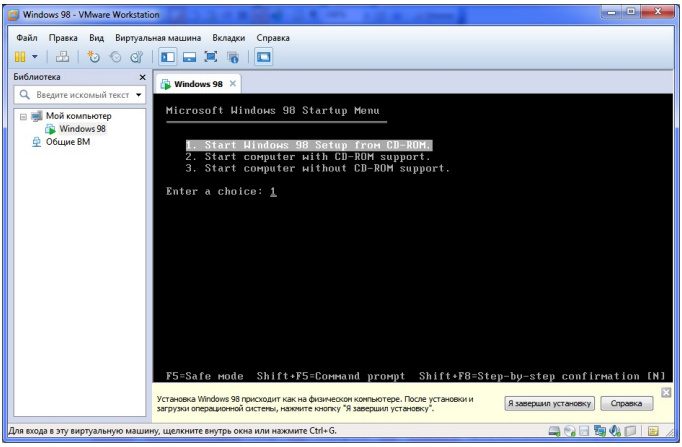


Рис. 1.7 – Вибір варіанта завантаження та встановлення ОС

На наступному етапі необхідно погодитися з установкою, натиснувши клавішу ENTER (рис. 1.8).

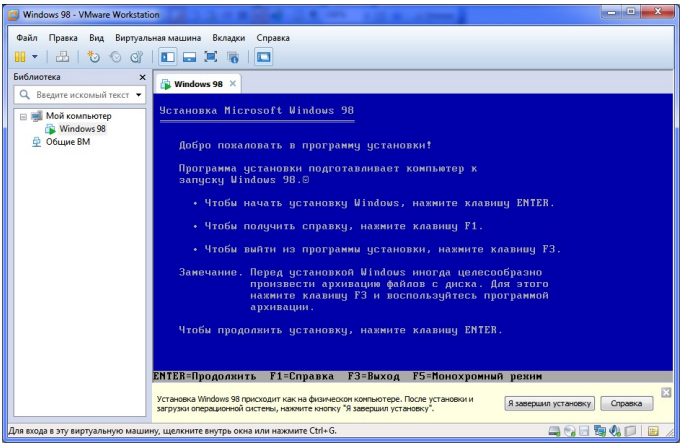


Рис. 1. 8 – Вікно з тарту установки ОС

Далі установник ОС запропонує підготувати вільний простір на дисках. Потрібно підтвердити цю дію натисканням клавіші ENTER (рис. 1.9).

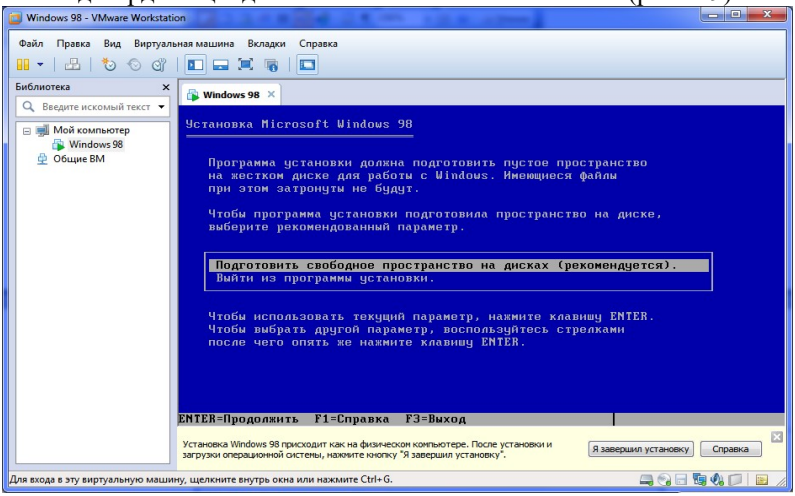


Рис. 1. 9 - Підготовка вільного простору на дисках

Потім необхідно включити підтримку великих дисків (рис. 1.10).

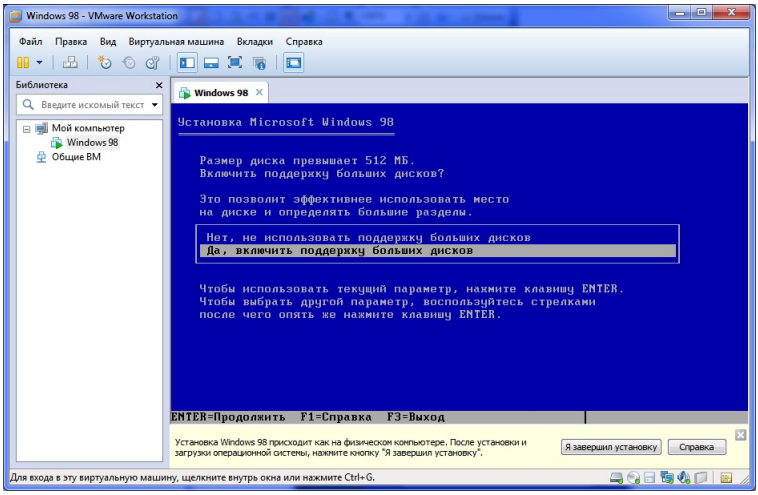


Рис. 1.10 – Увімкнення підтримки великих дисків

Потім майстер установки ОС запропонує перезавантаження ОС. Необхідно підтвердити цю дію натисканням клавіші ENTER (рис. 1.11).

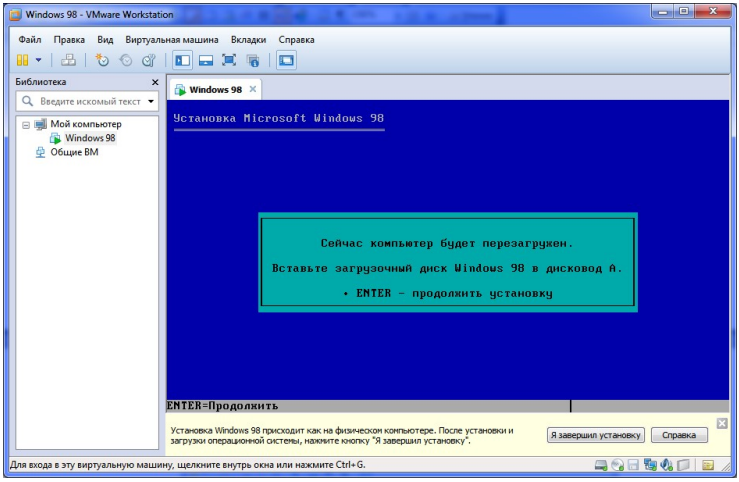


Рис. 1.11 - Завершення підготовки установки

Після перезавантаження необхідно вибрати ті самі варіанти, що й були на рис. 1.6 та рис. 1.7. Потім майстер виконає форматування диска (рис. 1.12).

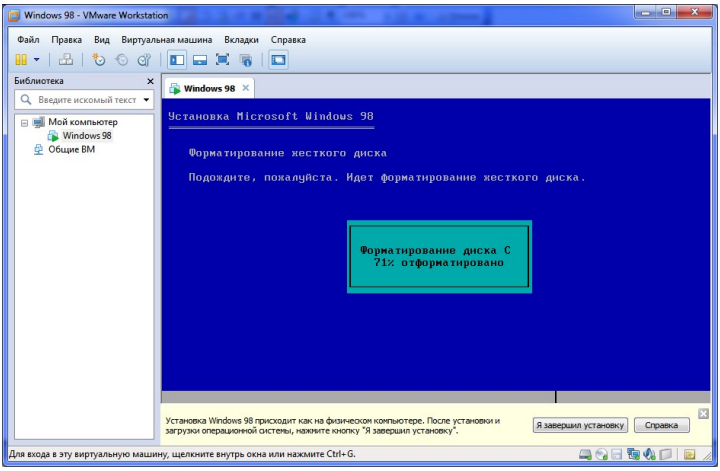


Рис. 1.12. – Форматування диска

Після цього необхідно погодитися з початком процесу встановлення натисканням клавіші ENTER (рис. 1.13).

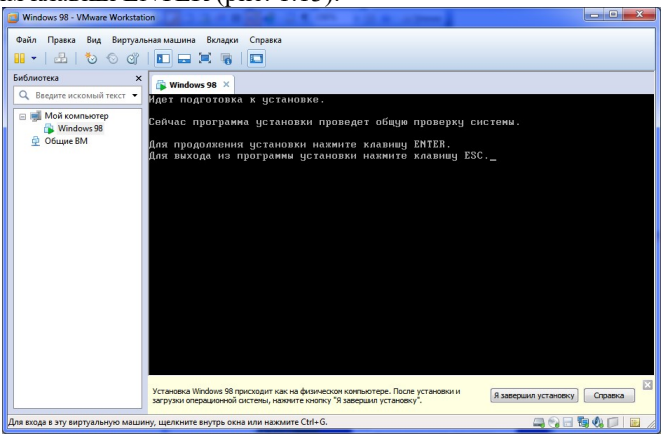


Рис. 1.13. – Згода на процес встановлення

Далі майстер виконає перевірку диска та перейде на процес встановлення ОС (рис. 1.14).

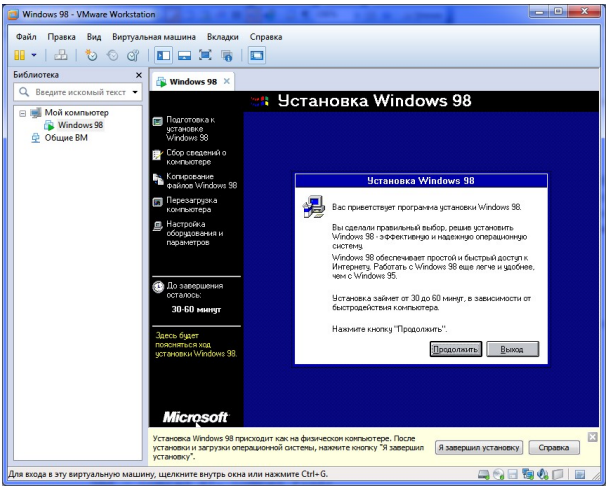


Рис. 1.14 - Встановлення гостьової ОС

При цьому слід вибрати папку установки ОС (рис. 1.15).

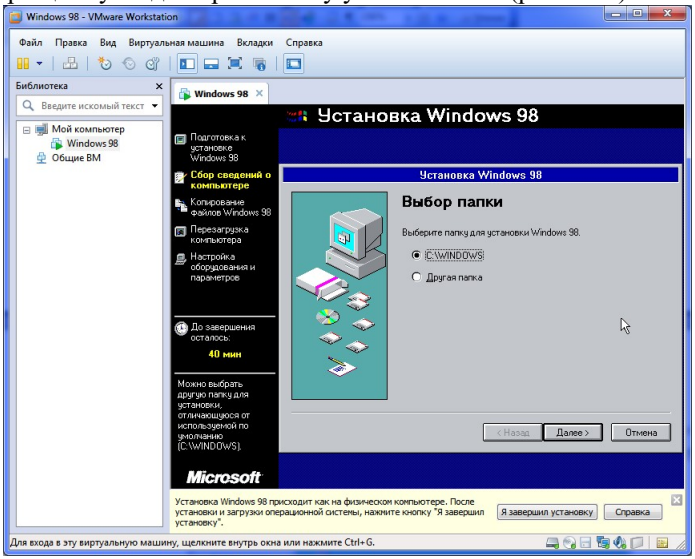


Рис. 1.15 – Вибір папки встановлення ОС

Потім потрібно вибрати варіант установки ОС (рис. 1.16) та погодиться із встановленням основних компонентів (рис. 1.17).

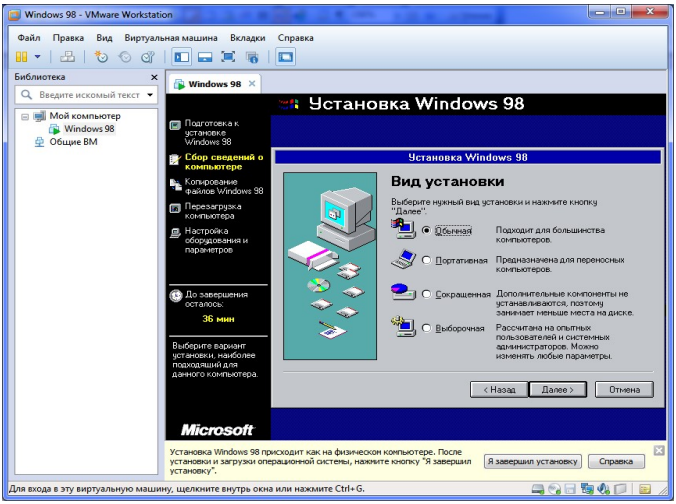


Рис. 1.16 – Вибір папки встановлення ОС

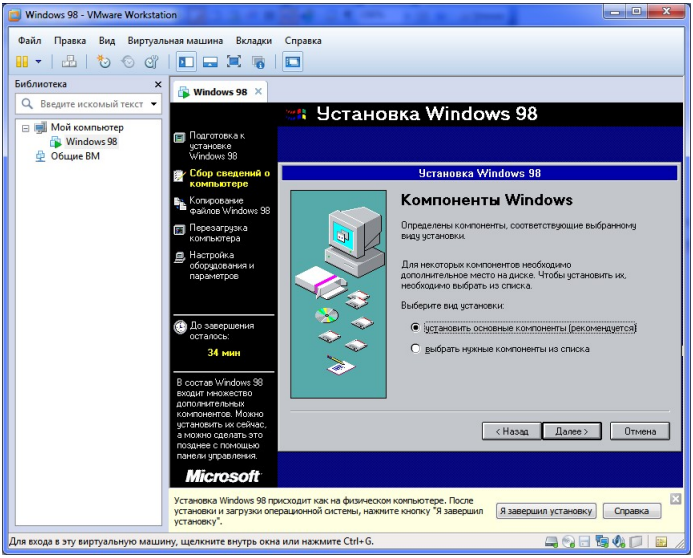


Рис. 1.17 - Вибір установки основних компонентів ОС

Потім потрібно вказати ім'я та опис даної робочої машини (рис. 1.18).

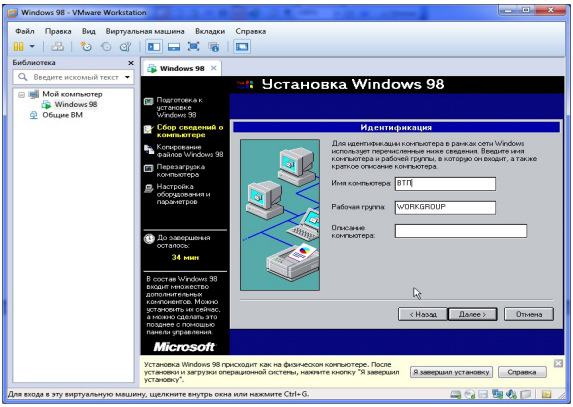


Рис. 1.18 - Встановлення імені та опису комп'ютера гостьової ОС

Далі необхідно вибрати місцезнаходження та погодитися з початком процесу встановлення, при цьому майстер переходить на процес копіювання файлів ОС (рис. 1.19).

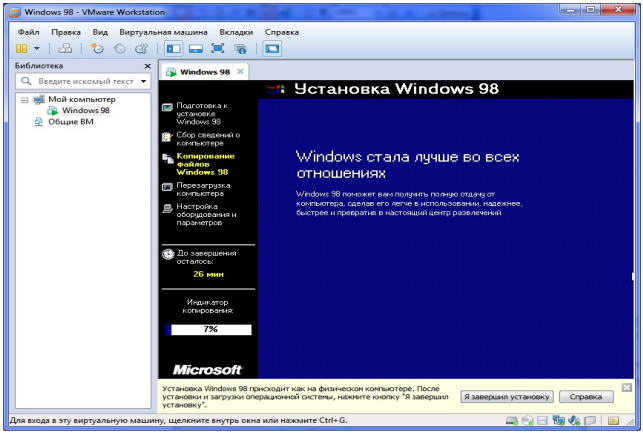


Рис. 1.19 - Процес копіювання файлів гостьової ОС на віртуальну машину

Після копіювання фалів ОС на диск, буде виконано перезавантаження гостьової ОС. Потім потрібно заповнити інформацію про користувача (перемикання між розкладками клавіатури здійснюється натисканням комбінації LeftAlt+LeftShift) , погодитися з ліцензійною угодою і ввести ліцензійний ключ та погодиться з продовженням установки, натиснувши кнопку «Готово» (рис. 1.20).

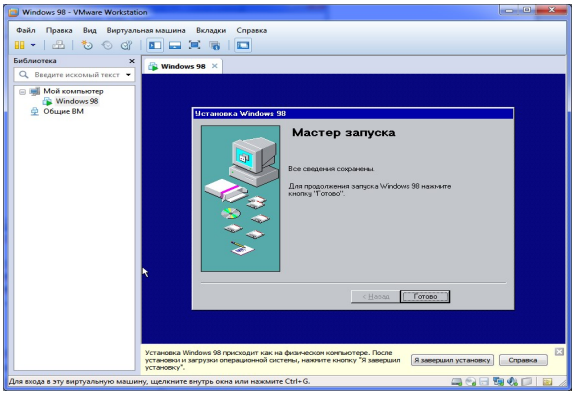


Рис. 1.20 – Продовження процесу встановлення

Потім майстер налаштує обладнання та після 2-х перезавантажень, ОС буде встановлена. Після інсталяції ОС, її запуск із VMWare Workstation здійснюється вибором її з бібліотеки та натисканням на кнопку «Увімкнути/Відновити» ( ) або комбінацією «Ctrl+B» (Рис. 1.21) .

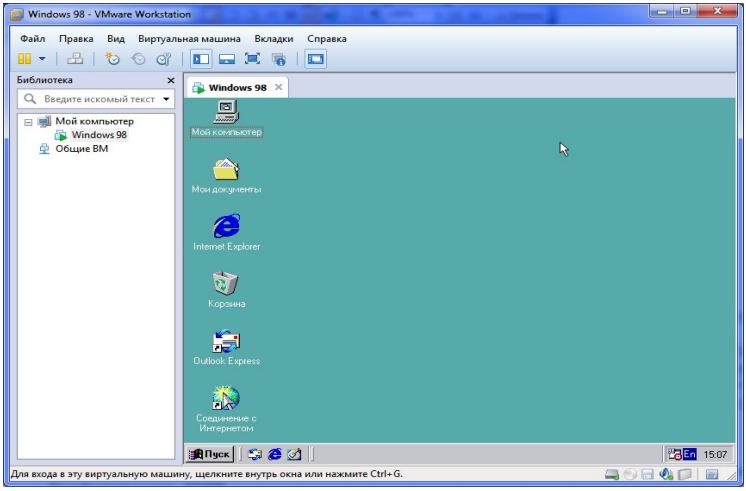


Рис. 1.21 – Вікно VMWare Workstation із запущеною ОС Windows 98

Після встановлення гостьової ОС необхідно встановити пакет VMWare Tools для цієї ОС. Даний пакет встановлює необхідні додаткові драйвери

(зокрема для Windows 98 встановлюється драйвери дисплея та миші), а також надає можливість копіювання файлів з гостьової ОС в хостову простими операціями перетягування. Для встановлення VMWare Tools необхідно при запущеній гостьовій ОС вибрати пункт меню "Віртуальна машина\Встановити VMWare Tools".

Після встановлення VMWare Tools загальний вигляд Windows 98 представлено на рис. 22. Система готова до виконання лабораторних робіт з курсу «Архітектура комп'ютерів».

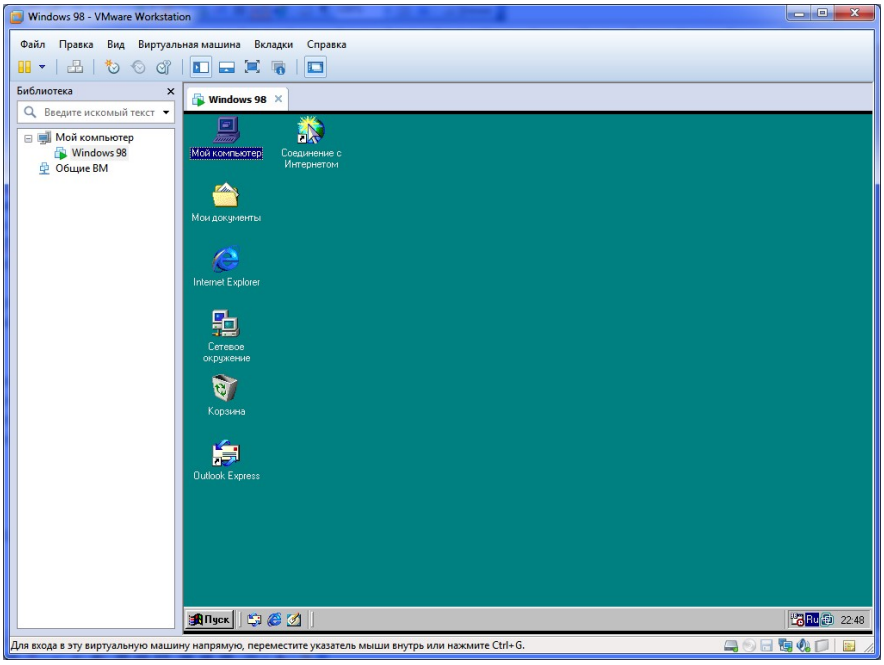


Рис. 1.22 – Загальний вигляд ОС Windows 98

Каталог із встановленою гостьовою ОС (тобто сама ОС) може бути використаний і на іншому комп'ютері із встановленим VMWare Workstation. Для цього можна просто повністю скопіювати весь каталог гостьової ОС на інший комп'ютер і в VMWare Workstation вибрати його з меню "Файл \ Відкрити ..." або комбінацією Ctrl + O.

Після встановлення гостьової ОС до неї необхідно додати компілятори мов програмування для виконання лабораторних робіт. Цю дію можна виконати простим перетягуванням файлів з хостової ОС на гостьову ОС.

1. 2. Зміст звіту
1. 2.1. Мета роботи.
1. 2.2. Хід установки гостьової ОС
1. 2.3. Висновки.

Лабораторна робота 2

Тема: Конфігурація персонального комп'ютера

Мета роботи: Отримання практичних навичок визначення конфігурації та основних характеристик ПЕОМ та її модулів.

2.1.Теми для попереднього опрацювання

Склад, призначення та характеристики основних модулів ПЕОМ. Конфігурація ПЕОМ.

2.2. Опис роботи

При вирішенні деяких завдань необхідно знати тип ПЕОМ, тип мікропроцесора, склад зовнішніх пристроїв машини та їх технічні характеристики (наприклад, спроба звернення програми до неіснуючого пристрою може призвести до «зависання» операційної системи). Ця інформація міститься у певних осередках ОЗП, ПЗП та КМОП пам'яті та наведена нижче.

2. 2.1 . Визначення типу ПЕОМ.

ПЗП BIOS містить за адресою F000:FFFEh 1 байт, що дозволяє ідентифікувати тип ПЕОМ:

FF h – оригінальний IBM PC;
FE h - XT, Portable PC;
FD h - PCjr;
FC h - AT;
FB h - XT з пам'яттю 640 К на системній платі;
FA h – PS/2 модель 25 чи 30;
F9h - Convertible PC;
F8h - PS/2 моделі 55SX, 70, 80;
9Ah - Comrad XT, Compaq Plus;
30h - Sperry PC;
2Dh – Compaq PC, Compaq Deskpro.

2. 2.2 . Визначення типу мікропроцесора.

Алгоритм визначення типу мікропроцесора ґрунтується на відмінностях у регістрах прапорів (РгП) мікропроцесорів (МП) 8086, 80286 та 80386 і полягає в наступному:

У регістр прапорів записується 0. Якщо біти 12–15 РгП встановлюються в 1 – це МП 8086. Якщо ні, то у регістр прапора записується код F000h. Якщо після цього біти 12–15 РгП залишаються в 0, це МП 80286, інакше – 80386 і старші .

Для МП вище 80386 існує команда визначення типу та параметрів МП: CPUID . Код цієї команди у машинному коді 0Fh 0A2h. Отримати інформацію про виробника МП можна, задавши значення в регістрі EAX = 0. Тоді CPUID повертає у регістрах EBX, EDX та ECX ідентифікатор виробника процесора (Vendor ID) у вигляді 12 символів ASCII. Для отримання повного найменування МП необхідно послідовно викликати команду CPUID з параметрами EAX = 80000002h, 80000003h і 80000004h. На кожен запит у регістрах EAX:EBX:ECX:EDX повертатиметься фрагмент 48-символьного рядка повного найменування процесора

2.2.3. Визначення дати створення BIOS.

Дата створення BIOS займає в ПЗП BIOS 8 байтів, починаючи з адреси F000:FFF5h і зберігається у форматі ASCII у вигляді мм/дд/рр, де

мм – номер місяця; дд – день; рр – рік.

2.2.4. Визначення конфігурації IBM PC AT.

2.2.4.1. Дані, що зберігаються в CMOS(КМОП)-пам'яті.

CMOS (КМОП)-пам'ять організована на базі мікросхеми MC146818 фірми Motorola і має 64 8-розрядні комірки (регістру). У табл. 2.1 наведено вміст тих комірок пам'яті, в яких зберігаються дані про конфігурацію та стан машини.

Звертання до CMOS (запис та зчитування) здійснюється наступним чином: спочатку в порт 70 h заноситься адреса комірки (номер регістру). Потім залежно від виконуваної операції у порт 71 h або записуються дані, або з порту 71 h дані зчитуються.

Таблиця 2. 1 - Вміст комірок

Адреса комірки	Вміст
0E h	Байт стану діагностики при включенні живлення
10 h	Тип використовуваного НГМД
14 h	Конфігурація обладнання
15 h- 16 h	Об'єм основної пам'яті
17 h- 18 h	Об'єм розширеної (extended) пам'яті

2. 2.4.1.1. Байт стану діагностики.

Байт стану діагностики містить результати виконання діагностики під час включення живлення машини. Формат байта стану наведено у табл. 2. 2.

Таблиця 2. 2 - Вміст байта діагностики

Біти	Значення
0-1	Не використовується, дорівнює 0
2	0 – неправильне встановлення годинника реального часу; 1 - годинник встановлений правильно .
3	1 – несправність НМД, неможливо завантажити операційну систему із жорсткого диска; 0 - НМД справний .
4	1 – фактичний розмір оперативної пам'яті не відповідає зазначеному в КМОП-пам'яті; 0 – розмір оперативної пам'яті вказано правильно .
5	1 – помилка у конфігурації системи; 0 – конфігурація вказана правильно .
6	1 – помилка у контрольній сумі КМОП-пам'яті; 0 – контрольна сума КМОП-пам'яті правильна .
7	1 - розрядка акумулятора, що живить КМОП-пам'ять і годинник реального часу; 0 – акумулятор справний та заряджений .

2. 2.4.1.2. Тип використовуваних флоппі-дисків.

Молодша та старша тетради цього байта описують відповідно другий та перший НГМД:

0000 - дисковод не встановлений;
0001 - дисковод на 360К;
0010 - дисковод на 1.2М;
0011 - дисковод на 720К;
0100 - дисковод на 1.44м.

2. 2.4.1.3. Конфігурація обладнання.

Таблиці 2. 3 - Байт конфігурації

Біти	Значення
0	1 – у системі встановлені НГМД; 0 - НГМД не використовуються.
1	1 – встановлений арифметичний сопроцесор; 0 - сопроцесор не встановлений.
2-3 _	Не використовуються, дорівнюють 0.
4-5	Тип відеоконтролера та його режим: Біти: 5 4 0 0 - не використовуються або EGA; 0 1 – CGA, EGA, VGA у режимі 40x25; 1 0 – CGA, EGA, VGA у режимі 80x25; 1 1 - монохромний контролер .
6-7	Кількість використовуваних НГМД .

2. 2.4.1.4. Об'єм основної пам'яті.

Комірка 15 h містить молодший байт, а комірка 16 h – старший байт розміру основної пам'яті в кілобайтах.

2. 2.4.1.5. Об'єм додаткової пам'яті.

Комірки 17 h і 18 h містять відповідно молодший і старший байти розміру додаткової пам'яті (розташованої вище 1М) в кілобайтах.

2. 2.4.1.6. Дані із RTC.

Дані у регістрах RTC зберігаються у двійково-десятковому форматі (BCD), тобто кожні 4 біти містять один десятковий розряд. Тому для виведення інформації з RTC необхідно виконувати перетворення з BCD у двійковий формат і навпаки. У табл. 2. 4 наведено вміст регістрів RTC.

Таблиця 2. 4 – Дані RTC

Адреса комірки	Вміст
0h	Поточний час, секунди.
1 h	Секунди будильника.
2h	Поточний час, хвилини.
3 h	Хвилини будильника.

Продовження таблиці 2. 4.

Адреса комірки	Вміст
4 h	Поточний час, години.
5 h	Години будильника.
6 h	Поточний день тижня.
7 h	Поточний день місяця.
8 h	Поточний місяць.
9 h	Поточний рік.

2. 2.4.2. Дані, що зберігаються в області BIOS оперативної пам'яті.

BIOS у процесі ініціалізації опитує стан перемичок та аналізує вміст КМОП-пам'яті. Після аналізу BIOS записує у свою область пам'яті за адресою 0040:0010h слово конфігурації. Призначення окремих біт цього слова наведено в табл. 2.5.

Таблиця 2.5 – Вміст слова конфігурації

Біти	Значення
0	1 – у системі встановлені НГМД; 0 - НГМД не використовуються.
1	1 – встановлений арифметичний сопроцесор; 0 - сопроцесор не встановлений.
2-3 _	Об'єм основної пам'яті, встановленої на системній платі: Біти: 3 2 0 1 - 16К; 1 0 - 32К; 1 1 - 64К і більше.
4-5	Тип відеоконтролера та його режим: Біти: 5 4 0 0 - не використовуються або EGA; 0 1 – CGA, EGA, VGA у режимі 40x25 1 0 – CGA, EGA, VGA у режимі 80x25 1 1 – монохромний контролер
6-7	Кількість НГМД, що використовуються.
8	1 - використовується контролер прямого доступу до пам'яті; 0 – контролер прямого доступу до пам'яті не використовується.
9-11	Кількість встановлених портів послідовної передачі.

Продовження таблиці 2.5.

Біти	Значення
12	1 - застосовується ігровий адаптер (джойстик); 0 – ігровий адаптер не застосовується.
13	1 – встановлений послідовний принтер (лише для PCjr).
14-15	Кількість встановлених принтерів .

2.3. Порядок виконання роботи

2.3.1. Відповідно до індивідуального завдання, використовуючи довідкову інформацію про технічні характеристики ПЕОМ, наведену в розділі 3, визначити адресу або область ПЗП, ОЗП або КМОП-пам'яті, що містять необхідну інформацію.

2.3.2. Написати програму визначення необхідних показників ПЕОМ.

2.4. Особливості програмування

2.4.1. Мовою Паскаль.

2.4.1.1. Для звернення до комірок ОЗП та ПЗП застосовуються попередньо визначені масиви Mem та MemW. Наприклад, для читання слова з комірки ОЗП 0040:0010 h використовується вираз wo:=MemW[$0040:$0010], де wo – змінна типу word; для читання байта з ПЗП за адресою F000:FFF5 h використовується вираз b:=Mem[$f000:$fff5] (b змінна типу byte).

2.4.1.2. Для звернення до портів ПЕОМ використовуються попередньо визначені масиви Port і PortW. Наприклад, для запису значення 10h порт 70h використовується вираз Port[$70]:=$10; для читання з порту 71h використовується вираз data:=Port[$71], де data - змінна типу byte.

2.4.1.3. Для виділення певного розряду в байті використовується бітова маска, що містить одиницю в розряді, що перевіряється, і нулі - в інших. Бітова маска складається з байтом за схемою порозрядної кон'юнкції AND: у результаті отримуємо значення 0, якщо заданий біт містить 0 і значення 1 у протилежному випадку. Наприклад, для визначення значення 0-го розряду змінної b (типу byte) необхідно записати:

```
if(b and $01)=0 then writeln( '0- й розряд байта b містить 0')
    else writeln(' 0- й розряд байта містить 1') ;
```

2. 4.2. Мовою С.

2. 4.2.1. Для звернення до комірок ОЗП та ПЗП використовуються дальні покажчики, які оголошуються в програмі таким чином:

```
char far *uk; – для роботи з байтами
int far *uk; -для роботи з двобайтними словами (змінні типу int).
```

При цьому першим слідує молодший байт, потім – старший байт.

Наприклад, для читання слова з ОЗП за адресою 0040:0010h використовується вираз:

```
uk = (int far *) 0x00400010;
wo=*uk;
```

де: wo – змінна типу int.

Для читання байта із ПЗП за адресою F000:FFF5h використовується вираз:

```
uk = (char far *) 0xF000FFF5;
b = * uk;
```

де: b - Змінна типу char.

2.4.2.2. Для виклику машинних команд необхідно використовувати вбудований асемблер. Так код програми, що визначає виробника МП буде мати вигляд:

```
char CPUVendor[1 2 ];
 asm {
mov eax, 0h;
db 0fh,0A2h
mov dword ptr CPUVendor, ebx
mov dword ptr CPUVendor+4,edx
mov dword ptr CPUVendor+8,ecx
}
```

2.4.2.3. Для звернення до портів ПЕОМ застосовуються функції читання та запису у порт, що зберігаються у бібліотеці <dos.h>. Бібліотека підключається директивою

```
#include <dos.h>
```

після чого, наприклад, для запису значення 10h порт 70h використовується функція:

outportb(0x70,0x10).

Для читання з порту 71h використовується функція:
data=intportb(0x71);
де data – змінна типу char.

2.4.2.4. Для виділення певних розрядів у байті використовується бітова маска, яка складається з байтом за схемою порозрядної кон'юнкції & (не плутати з логічною кон'юнкцією &&), в результаті чого виходить значення істини (1), якщо заданий біт містить 1 і хибне значення (0) у протилежному випадку. Наприклад, для визначення значення 0-го розряду змінної b (типу char) необхідно записати:

```
If (b&0x1) printf("0-й розряд b містить 1\n");
    else printf("0-й розряд b містить 0\n").
```

2.5. Індивідуальні завдання

1. Визначити тип IBM; число НГМД; перевірити, чи акумулятор справний.
2. Визначити день створення BIOS; обсяг основної пам'яті та кількість підключених принтерів.
3. Визначити місяць створення BIOS та кількість підключених послідовних портів; перевірити, чи встановлений сопроцесор.
4. Визначити тип 1-го НГМД та рік створення BIOS; перевірити, чи ПЕОМ має контролер прямого доступу до пам'яті.
5. Визначити тип відеоконтролера, його режим та дату створення BIOS; перевірити справність НМД
6. Визначити тип IBM; визначити, чи має ПЕОМ відеоконтролер EGA; перевірити, чи правильно встановлено годинник реального часу.
7. Визначити обсяг додаткової пам'яті; визначити день створення BIOS; перевірити, чи встановлений сопроцесор.
8. Визначити тип 2-го НГМД (якщо він є); обсяг основної пам'яті на системній платі та рік створення BIOS.
9. Визначити кількість підключених принтерів та дату створення BIOS; перевірити, чи правильно встановлено годинник реального часу.
10. Визначити поточний час; визначити місяць створення BIOS; перевірити, чи ПЕОМ має контролер прямого доступу до пам'яті.
11. Визначити тип мікропроцесора та тип ПЕОМ; перевірити, чи акумулятор справний.
12. Визначити поточну дату; визначити рік створення BIOS та кількість

підключених принтерів.

13. Визначити тип IBM; обсяг додаткової пам'яті та кількість підключених послідовних портів.

14. Визначити тип відеоконтролера, його режим; перевірити, чи встановлений сопроцесор; визначити місяць створення BIOS

15. Визначити поточний місяць; визначити кількість підключених принтерів та дату створення BIOS.

16. Визначити тип IBM; обсяг додаткової пам'яті; перевірити, чи ПЕОМ має контролер прямого доступу до пам'яті.

17. Визначити поточну дату, рік створення BIOS та кількість підключених принтерів.

18. Визначити тип мікропроцесора та дату створення BIOS; перевірити, чи правильно встановлено годинник реального часу.

19. Визначити тип IBM; визначити тип відеоконтролера, його режим; перевірити, чи встановлений сопроцесор.

20. Визначити поточний час; обсяг основної пам'яті на системній платі та тип мікропроцесора.

2. 6. Зміст звіту

2. 6.1. Тема лабораторної роботи
2. 6.2. Мета роботи.
2. 6.3. Індивідуальне завдання.
2. 6.4. Текст програми.
2. 6.5. Результати роботи програми.
2. 6.6. Висновки.

Лабораторна робота 3

Тема: Організація переривань у ПК

Мета роботи: Вивчення механізму переривань, їх типів, алгоритму та засобів їхньої обробки, а також набуття практичних навичок створення власних програм обробки переривань.

3.1. Теми для попереднього опрацювання

3.1.1. Робота контролера переривань. Встановлення вектору переривання.
3.1.2. Маскування переривань.
3.1.3. Резидентні програми.

3.2. Опис роботи

3.2.1. Механізм та типи переривань.

Для обробки подій, що відбуваються асинхронно по відношенню до виконанню програми, найкраще підходить механізм переривань. Переривання можна розглядати як деяку особливу подію у системі, яка потребує моментальної реакції. Наприклад, добре спроектовані системи підвищеної надійності використовують переривання по аварії в мережі живлення для виконання процедур запису вмісту регістрів і оперативної пам'яті на магнітний носій, щоб після відновлення живлення можна було б продовжити роботу з того ж місця.

Оскільки переривання можливі найрізноманітніші з різних причин, кожному перериванню присвоюється номер переривання. З кожним номером переривання пов'язують ту чи іншу подію. Система вміє розпізнати яке переривання, з яким номером сталося, і запускає відповідну номеру процедуру.

За джерелом та характером виникнення переривання можна поділити на групи (рис. 3.1).

Програмні переривання викликають самі програми, тому вони є асинхронними. Для цього використовують команду INT.

Програмні переривання зручно використовувати для організації доступу до окремих загальних для всіх програм модулів. Наприклад, програмні модулі операційної системи доступні прикладним програмам саме через переривання, і немає необхідності при виклику цих модулів знати їхню поточну адресу в пам'яті. Прикладні програми самі можуть встановлювати свої обробники переривань для їхнього подальшого використання іншими програмами. Для цього оброблювані переривання повинні бути резидентними в пам'яті.

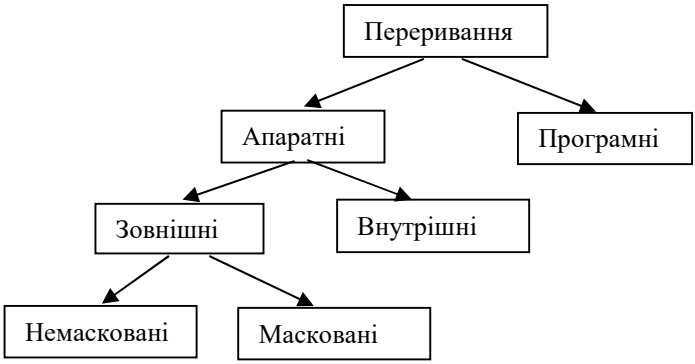


Рис. 3.1 – Типи переривань

Апаратні переривання викликаються фізичними пристроями та приходять асинхронно. Ці переривання інформують систему про події, пов'язані з роботою пристроїв, наприклад про те, що завершився друк символу на принтері і непогано було б видати наступний символ, або про те, що необхідний сектор диска вже прочитаний, його зміст доступний програмі.

Використання переривань при роботі з повільними зовнішніми пристроями дозволяє поєднати введення/виведення з обробкою даних у центральному процесорі та в результаті підвищує загальну продуктивність системи.

Зовнішні апаратні переривання викликані сигналами, зовнішніми відносно центрального процесора, і подаються на його входи INT і NMI.

Переривання по входу INT відносяться до апаратних переривань, що маскуються, оскільки можуть бути дозволені або заборонені прапором IF регістра прапорів. Номер вектора переривань переривань, що маскуються, передається в процесор по його восьми молодшим розрядам шини даних.

Вхід немаскованого переривання NMI зазвичай використовується для повідомлень про "катастрофічні" події (відключення живлення, виявлення помилок пам'яті і т.д.). Номер цього переривання дорівнює 2.

Внутрішні (логічні) переривання формуються самим процесором, коли він зустрічається з деякими особливими подіями на кшталт поділу на 0. Це переривання з номерами 0, 1, 3, 4 (див. табл. 3.1).

У табл. 3.1 дано опис найважливіших переривань.

Таблиця 3. 1 – Опис переривань

Номер	Опис
0	Помилка поділу. Викликається автоматично після виконання команд DIV або IDIV, якщо в результаті поділу відбулося переповнення. DOS зазвичай під час роботи цього переривання виводить повідомлення про помилку і зупиняє виконання програми. Для процесора 8086 адреса повернення вказує на наступну після команди поділу команду, а в процесорі 80286 - на перший байт команди, що викликала переривання.
1	Переривання покрокового режиму. Виробляється після виконання кожної машинної команди, якщо у слові прапорів встановлено біт покрокового трасування TF. Використовується для налагодження програм. Переривання не виробляється після виконання команди MOV у сегментні регістри або після завантаження сегментних регістрів командою POP.
2	Апаратне переривання, що не маскується. Це переривання може використовуватися по-різному на різних машинах. Зазвичай виробляється при помилці парності в оперативній пам'яті та при запиті переривань від сопроцесора.
3	Переривання для трасування. Це переривання генерується під час виконання однобайтової команди з кодом CCh і зазвичай використовується відладниками для встановлення точки переривання.
4	Переповнення. Генерується машинною командою INTO, якщо встановлено прапор OF. Якщо прапор не встановлений, команда INTO виконується як NOP. Це переривання використовується для обробки помилок під час виконання арифметичних операцій.
5	Друк копії екрану. Генерується при натисканні на клавіатурі клавіші PrtScr. Зазвичай використовується для друку зображення. Для процесора 80286 генерується при виконанні машинної команди BOUND, якщо значення, що перевіряється, вийшло за межі заданого діапазону.
6	Невизначений код операції або довжина команди більше 10 байт (для процесора 80286).
7	Особливий випадок відсутності сопроцесора.
8	IRQ0 – переривання інтервального таймера, що виникає 18,2 рази на секунду.
9	IRQ1 – переривання від клавіатури. Генерується при натисканні та при відтисканні клавіші. Використовується для читання даних з клавіатури.
Ah	IRQ2 – використовується для каскадування апаратних переривань у машинах класу AT.
Bh	IRQ3 – переривання асинхронного порту COM2.
Ch	IRQ4 – переривання асинхронного порту COM1.

Продовження таблиці 3.1

Номер	Опис
Dh	IRQ5 – переривання від контролера жорсткого диска XT.
Eh	IRQ6 – переривання генерується контролером флоппі-диска після завершення операції.
Fh	IRQ7 – переривання принтера. Генерується принтером, коли він готовий до виконання чергової операції. Багато адаптерів принтера не використовують це переривання.
10h	Обслуговування відеоадаптера .
11h	Визначення конфігурації пристроїв у системі.
12h	Визначення розміру оперативної пам'яті у системі.
13h	Обслуговування дискової системи .
14h	Послідовне введення/виведення .
15h	Розширений сервіс для комп'ютерів AT.
16h	Сервіс клавіатури.
17h	Обслуговування принтера .
18h	Запуск BASIC у ПЗП, якщо він є.
19h	Завантаження операційної системи
1Ah	Обслуговування годинника .
1Bh	Обробник переривань Ctrl-Break .
1Ch	Переривання виникає 18,2 рази на секунду, викликається програмно обробником переривання таймера .
1Dh	Адреса відеотаблиці для контролера відеоадаптера 6845.
1Eh	Вказівник на таблицю параметрів дискети.
1Fh	Вказівник на графічну таблицю для символів із кодами ASCII 128-255 .
20h - 5Fh	Використовуються DOS або зарезервовані для DOS.
60h - 67h	Переривання, зарезервовані для користувача .
68h - 6Fh	Не використовуються .
70h	IRQ8 - переривання від годинника реального часу.
71h	IRQ9 – переривання від контролера EGA.
72h	IRQ10 – зарезервовано .
73h	IRQ1 1 – зарезервовано .
74h	IRQ1 2 – зарезервовано .
75h	IRQ13 - переривання від сопроцесора .
76h	IRQ14 – переривання від контролера жорсткого диска.
77h	IRQ15 – зарезервовано .
78h - 7Fh	Не використовуються .
80h - 85h	Зарезервовані для BASIC .
86h - F0h	Використовуються інтерпретатором BASIC .
F1h - FFh	Не використовуються

3.3.2. Таблиця переривання векторів.

Щоб зв'язати номер переривання з адресою програми обробки переривань (обробника переривань), використовується таблиця векторів переривань, що займає перший кілобайт оперативної пам'яті - адреси від 0000:0000 до 0000:03FFh. Таблиця складається з 256 елементів – FAR-адрес обробників переривань. Ці елементи називають векторами переривань. У першому слові елемента таблиці записано зсув, тоді як у другому – сегмент адреси обробника переривань.

Перериванню з номером 0 відповідає адреса 0000:0000, перериванню з номером 1 - 0000:0004 і т.д. Перериванню з номером n відповідатиме адреса 0000:4*n. Ініціалізація таблиці відбувається частково програмою BIOS після тестування апаратури, частково під час завантаження ОС. ОС може переключити на себе деякі переривання BIOS.

3.3.3. Обробка переривань.

Незважаючи на різноманіття типів переривань, алгоритм обробки переривання процесором однаковий і зображений на рис. 3. 2.

3. 3.4. Маскування переривань.

Часто під час виконання критичних ділянок програм, щоб гарантувати виконання певної послідовності команд, доводиться забороняти переривання. Це можна зробити командою CLI (скидання прапора роздільної здатності переривання). Її потрібно помістити на початок критичної послідовності команд, а наприкінці розташувати команду STI (установити прапор дозволу переривання). Команда CLI забороняє тільки переривання, що маскуються, немасковані завжди обробляються процесором.

Якщо ви використовуєте заборону переривань за допомогою команди CLI, слідкуйте за тим, щоб переривання не відключалися на тривалий період часу, тому що це може призвести до небажаних наслідків, наприклад, відставатиме годинник.

Якщо треба заборонити не всі переривання, а лише деякі, наприклад, від клавіатури, то для цього треба скористатися послугами контролера переривань.

3.3.5. Зміна таблиці векторів переривань.

Якщо програмі потрібно організувати обробку деяких переривань, для цього необхідно перепризначити необхідний вектор переривань на свій обробник. Це можна зробити, змінивши зміст відповідного елемента таблиці векторів переривань. Дуже важливо не забути до завершення роботи програми відновити вміст змінених векторів, оскільки після завершення роботи

програми пам'ять, яка їй була розподілена, вважається вільною і може бути використана для завантаження іншої програми. Якщо вектор не було відновлено, і прийшло переривання, то робота системи може порушитись – вектор тепер вказує на область, яка може містити все, що завгодно.

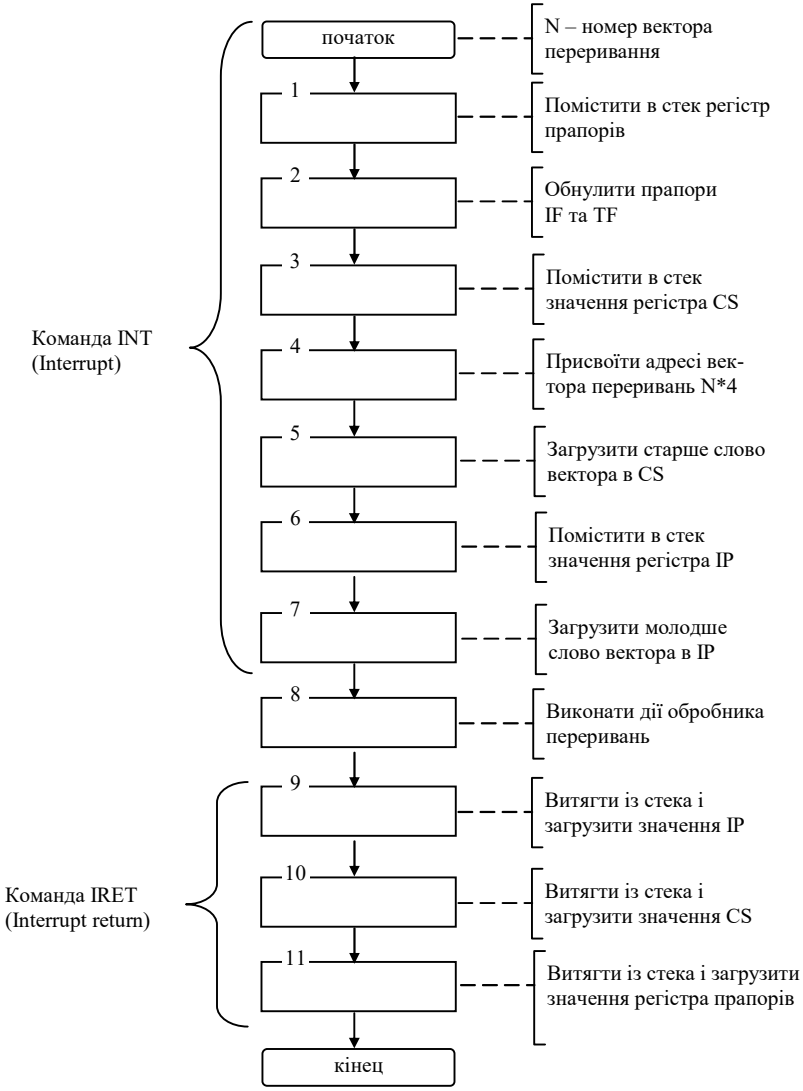


Рис. 3.2 – Алгоритм обробки переривань

Тому послідовність дій для нерезидентних програм, що обробляють переривання, має бути такою:

– прочитати вміст елемента таблиці векторів переривань для вектора з номером переривання, яке потрібно обробити;

– запам'ятати цей вміст (адресу старого обробника переривань) в області даних програми;

– встановити нову адресу в таблиці векторів переривань так, щоб вона відповідала початку програми обробки переривань користувача;

– перед завершенням роботи програми прочитати з області даних адресу старого обробника переривання та записати його до таблиці векторів переривань.

Зазначимо, що для резидентних програм (тобто програм, що постійно знаходяться в оперативній пам'яті) останній пункт виконувати не потрібно.

Операція зміни вектора переривань має бути безперервною у тому сенсі, що під час зміни не має відбутися переривання з номером, для якого здійснюється заміна програми обробки. Якщо, наприклад, буде записано нове значення зсуву, а сегмент адреси не встигне оновитися, то за якою адресою буде передано управління у разі переривання і що при цьому станеться?

Тому перед записом у вектор переривань потрібно заборонити переривання командою CLI, а після запису – дозволити командою STI.

Для полегшення роботи із заміни переривання DOS має спеціальні функції для читання елемента таблиці переривань і запису до неї нової адреси. Правильність операцій при цьому буде гарантовано.

Для читання вектора використовуйте функцію 35h DOS (переривання 21h).

Перед викликом регістр AL повинен містити номер вектора в таблиці. Після виконання функції в регістрах ES:BX буде потрібна адреса обробника переривання.

Функція 25h DOS встановлює для вектора з номером, що знаходиться в AL, обробник переривань DS:DX.

3.3.6. Корекція системних обробників переривань.

Якщо необхідно додати будь-які власні дії до тих, що виконує стандартний обробник переривання, можна організувати ланцюжок переривань.

Для організації ланцюжка переривань необхідно записати в таблицю векторів адресу власного оброблювача, не забувши зберегти колишній вміст таблиці. Обробник користувача отримує керування перериванням, виконує будь-які дії, потім передає керування стандартному оброблювачу. Можна зробити й інакше: обробник користувача викликає стандартний обробник, а потім виконує додаткові дії. Тобто, можна вставити додаткову обробку як до виклику стандартного оброблювача, так і після його виклику.

3.3.7. Стан кнопок-перемикачів.

У табл. 3.2 наведено формат байта за адресою 0040:0017h , що містить поточні стани клавіш-перемикачів.

Таблиця 3.2 – Стан клавіш-перемикачів

Біт	Значення, коли біт = 1
0	натиснута клавіша правий Shift
1	натиснута клавіша лівий Shift
2	натиснута клавіша Ctrl
3	натиснута клавіша Alt
4	режим ScrollLock увімкнено
5	режим NumLock увімкнено
6	режим CapsLock увімкнено
7	режим вставки увімкнено

3.4. Порядок виконання

3.4.1. Розробити власну процедуру обробки переривання (див. 3.5.1.4), яка б виконувала такі дії:

– виклик програми обробки переривання (див. п. 3.5.1.3) з одним із номерів переривання, зарезервованих для користувача (див. табл. 3.1);

– аналіз стану заданої клавіші-перемикача згідно з таблицею 3.2;

– при натисканні кнопки – виведення на екран її назви.

3.4.2. Розробити програму, що виконує такі дії:

- зберегти адресу системної програми обробки переривання Int9;

– записати адресу цієї системної програми до таблиці векторів переривань за вибраним номером переривання користувача;

– записати адресу власної процедури обробки переривання до таблиці векторів за номером 9;

– завершити програму та залишити її в пам'яті резидентною.

3.5. Особливості програмування

3.5.1. Мовою Паскаль.

3.5.1.1. Для читання вектора переривання та зміни таблиці векторів переривань використовуйте попередньо визначений масив MemW. Наприклад,

для читання зсуву вектора 9 можна застосувати вираз

```
ofs_:= MemW[0000:4*9];
```

а для запису сегмента адреси обробника переривань у $60-й вектор переривань

```
Memw[0000:4*$60+2]:=Seg(Int9d);
```

де: ofs_:word – змінна для тимчасового зберігання значення зсуву;
Int9d – ім'я процедури обробки (корекції) переривання 9.

3.5.1.2. Для заборони та дозволу переривань використовуйте або команди на асемблері:

```
begin
. . . . .
asm
Cli {переривання заборонено}
end
. . . . .

asm
Sti {переривання дозволено}
end
. . . . . .
```

або оператори Inline з машинними командами:

```
 begin
. . . . .
Inline($fa) { Cli }
. . . . . .
Inline($fb) { Sti }
 . . . . . .
Inlin
```

3.5.1.3. Для виклику програм обробки заданого переривання використовується процедура Intr.

Процедура Intr має таку форму:

```
Intr(IntNo: byte; var Reg: Registers),
```

де: IntNo - номер переривання (0 ... 255),
Reg – запис типу Registers, визначеного у модулі DOS, що містить

значення регістрів процесора. Треба вказати у програмі модуль DOS після uses або включити до програми наступний опис типу запису:

```
 type
Registers = record
case integer of
0:(AX, BX, CX, DX, BP, SI, DI, DS, ES,
Flags: word);
1: (AL, AH, BL, BH, CL, CH, DL, DH: byte);
end;
```

де : AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags,AL,AH,BL,BH,CL,CH,DL,DH
- змінні, відповідні одноіменним регістрам процесора.

При виконанні програмного переривання (виконання процедури Intr) процедура Intr завантажує регістри AX, BX, CX, DX, BP, SI, DI, DS та ES процесора значеннями із запису Regs (задаються користувачем). Коли обробка переривання завершується, вміст регістрів AX, BX, CX, DX, BP, SI, DI, DS, ES та Flags повертається до запису Regs.

Обмеження: програмні переривання, які залежать від певних значень SP або SS на вході, або змінюють значення SP або SS на виході, не можуть бути виконані за допомогою цієї процедури.

3.5.1.4. Власна процедура обробки переривань має такий вигляд:

```
 procedure NewIntr(Flags, CS, IP, AX, BX, CX, DX, SI, DI, DS,
ES, BP: Word); Interrupt;
begin
...
 end.
```

Примітка:

- ім'я процедури задається користувачем;

– будь-яка кількість параметрів (у тому числі й усі) у процедурі можна опускати, але тільки поспіль з початку.

Виклик процедури обробки переривання здійснюється за допомогою процедури Intr (див. п. 3.5.1.2). Попередньо адреса процедури користувача має бути занесена до таблиці векторів переривання відповідно до обраного номера переривання або шляхом прямого запису (див. 3.5.1.1), або за допомогою процедури SetIntVec (див. 3.5.1.5), а адреса системної процедури обробки даного переривання збережена або в змінній (див. 3.5.1.1), або за допомогою процедури GetIntVec (див. 3.5.1.4).

3.5.1.5. Процедура GetIntVec повертає адресу заданого вектора

переривання і має такий вигляд:

GetIntVec(IntNo:byte; var Vector:pointer),

де: IntNo задає номер переривання (0...255) та адреса повертається у параметрі Vector).

3.5.1.6. Процедура SetIntVec встановлює заданий вектор переривання за заданою адресою і має такий вигляд:

SetIntVec(IntNo:byte; Vector:pointer),

де: IntNo задає номер переривання (0...255) та Vector задає адресу (при заданій адресі можна скористатися оператором @ або функцією Addr).

3.5.1.7. Щоб зробити програму резидентною, тобто. такою, яка постійно знаходилася б в оперативній пам'яті (всі програми обробки переривання повинні бути такі), необхідно використовувати в кінці програми процедуру Keep(0), яка завершує програму та залишає її в пам'яті (параметр у процедурі вказує значення коду повернення).

Оскільки програма залишається в пам'яті, включаючи сегмент даних, сегмент стека і купу, необхідно перед програмою вказати директиву компілятора, що мінімізує резидентну область пам'яті:
{$m $800,0,0}.

3.5.2. Мовою С.

3.5.2.1. Для читання вектора переривання та зміни таблиці векторів переривань використовуйте дальні покажчики (char far *). Наприклад, для встановлення покажчиків на вектори 9 та 60 можна застосувати вираз

```
char far *adr_p, *int9_p;
adr_p = (char far *) (4 * 9);
int9_p = (char far *) (4 * 60);
```

а для запису адреси обробника 9 переривань у 60-й вектор переривань

```
*int9_p=*adr_p;
```

3.5.2.2. Для виклику програм обробки заданого переривання використовується процедура

geninterrupt(IntNo),

де: IntNo - номер переривання (0...255).

Потрібно вказати у програмі підключення бібліотеки директивою
#include <dos.h>

3.5.2.3. Власна процедура обробки переривань повинна бути оброблена спеціальним модифікатором Interrupt, який автоматично зберігає значення всіх регістрів і відновлює їх перед поверненням управління основній програмі. При цьому використовується для повернення керування команда IRET замість звичайної команди RET.

Власна процедура обробки переривань має бути описана як:

void interrupt New_intr();
Тоді при виклику автоматично визначається її початкова адреса, якій надається значення покажчика

adr_p=(char far) New_intr;

3.5.2.4. Щоб зробити програму резидентною, тобто, такою, яка постійно знаходилася б в оперативній пам'яті (всі програми обробки переривань повинні бути такі), необхідно використовувати в кінці програми функцію 31h переривання 21h, яка завершує програму та залишає її в пам'яті резидентної.

```
union REGS in, out;
in.h.ah=0x31;/*завдання 31h функції завершити та залишити резидентною
в реєстр ah*/
in.h.al=0; /*код повернення*/
in.x.dx=size; /*об'єм виділеної пам'яті в 16-байтних параграфах */
int86(0x21,&in,&out); /*звернення до 21 переривання з параметрами*/
```

3.6. Індивідуальні завдання

3.6.1. Здійснити корекцію переривання Int 9 (від клавіатури) так, щоб:
а) виконувались усі дії системного оброблювача даного переривання;
б) при натисканні однієї із клавіш-перемикачів вивести на екран назву цієї клавіші.

3.6.2. Клавішу-перемикач вибрати за таким правилом:

нк := нс mod 8,

де: нк – номер клавіші-перемикача у табл. 3.2 ;
нс – номер студента у журналі.

3.7. Зміст звіту

3.7.1. Тема лабораторної роботи
3.7.2. Мета роботи.
3.7.3. Індивідуальне завдання.
3.7.4. Текст програми.
3.7.5. Результати роботи програми.
3.7.6. Висновки.

Лабораторна робота 4

Тема: Організація роботи клавіатури ПК

Мета роботи: Вивчення організації та принципів роботи клавіатури та набуття практичних навичок управління мікропроцесором клавіатури.

4.1. Теми для попереднього опрацювання

4.1.1. Організація та принцип роботи клавіатури ПЕОМ.

4.2. Опис роботи

4.2.1. Принципи роботи клавіатури На рис. 4.1 представлена спрощена схема клавіатури.

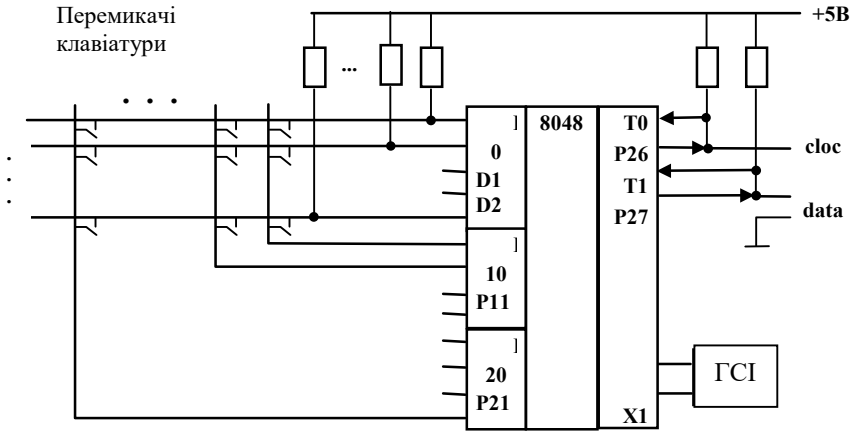


Рис. 4.1 – Схема клавіатури

В якості контролера клавіатури використовується спеціалізований мікропроцесор Intel 8042 на материнській платі, а на самій клавіатурі – Intel 8048, що має два порти – вхідний та вихідний. Вхідний порт (X) підключений до горизонтальних ліній матриці, а вихідний (Y) – до вертикальних. Усі горизонтальні лінії також підключені через резистори до джерела живлення +5В.

Встановлюючи по черзі на кожній із вертикальних ліній рівень напруги, що відповідає логічному нулю, клавіатурний контролер Intel 8048 опитує стан горизонтальних ліній. Якщо натиснутих клавіш немає, рівень напруги на всіх

горизонтальних лініях відповідає логічній 1 (бо всі ці лінії підключені до джерела живлення +5В через резистори). Якщо користувач натисне якусь клавішу, то відповідні вертикальна і горизонтальна лінії виявляться замкнутими. Коли на цій вертикальній лінії процесор встановить значення логічного нуля, рівень напруги на горизонтальній лінії також буде відповідати логічному нулю.

Як тільки на одній із горизонтальних ліній з'явиться рівень логічного нуля, клавіатурний процесор фіксує натискання клавіші.

Номер клавіші, що фіксується, однозначно пов'язаний з розпаюванням клавіатурної матриці і не залежить безпосередньо від позначень, нанесених на поверхню клавіш. Цей номер називається скан-кодом (Scan Code). Аналогічні дії виконуються, коли користувач відпускає раніше натиснуту клавішу. Зазначимо, що скан-код віджатої кнопки відрізняється від скан-коду натиснутої кнопки одиницею у старшому біті (тобто більше на 128).

Слово scan ("сканування") наголошує, що клавіатурний процесор сканує клавіатуру для пошуку натиснутої клавіші.

Отримавши по інтерфейсу клавіатури від Intel 8048 скан-код, Intel 8042 посилає в центральний процесор запит на переривання (Int 09h) і посилає скан-код в порт 60h.

Якщо натиснути і не відпускати клавішу, клавіатура перейде в режим автоповтору. У цьому режимі центральний процесор автоматично через деякий проміжок часу, званий періодом автоповтору, посилається код натиснутої клавіші.

4.2.2. Порти для роботи із клавіатурою.

Для роботи з клавіатурою ПК використовує порти з адресами 60h та 64h.

Порт 64h служить для запису команд контролеру 8042 (якщо команда містить байт даних, він передається через порт 60h), під час читання порт 64h містить слово стану 8042. Оскільки контролер 8048 немає безпосереднього зв'язку з системною шиною, то команди йому передаються через контролер 8042, для чого і команда і дані передаються через порт 60h. Порт 60h під час читання містить скан-код останньої натиснутої клавіші.

4.2.3. Команди контролера клавіатури 8048.

Контролер 8042 обслуговує не лише клавіатуру, а й інші системи комп'ютера (наприклад, з його допомогою виконується повернення центрального процесора із захищеного режиму роботи у реальний). Усі свої дії він здійснює, виконуючи відповідні команди.

Для посилки команди необхідно спочатку переконатися, що його внутрішня черга команд порожня (якщо команда складається з кількох байтів,

перевірка здійснюється для кожного з байтів). Це можна зробити, прочитавши слово стану 8042 із порту з адресою 64h. Біт з номером 1 повинен дорівнювати нулю (біти нумеруються з 0). Після того як програма дочекається готовності контролера 8042, вона може надіслати йому команду, записавши її у відповідний порт з адресою 64h або 60h (це робиться для кожного байту команди). Для керування клавіатурою контролер 8048 використовує наступні команди, які надсилаються в порт 60h:

4.2.3.1. Команда установки параметрів режиму автоповтору.

Команда складається із двох байтів. Перший байт: код команди – F3h; другий байт визначає характеристики режиму, біти якого мають таке призначення:

– 4-0: кількість повторів за секунду (див. табл. 4.1);

- 6-5: початкова затримка до генерації автоповтору в мс (00 = 250, 01 = 500, 10 = 750, 11 = 1000);

- 7: зарезервований, повинен дорівнювати 0.

Таблиця 4.1 – Параметри автоповтору

Константа	Частота	Константа	Частота	Константа	Частота
00h	30.0	0Bh	10.9	16h	4.3
01h	26.7	0Ch	10.0	17h	4.0
02h	24.0	0Dh	9.2	18h	3.7
03h	21.8	0Eh	8.6	19h	3.3
04h	20.0	0Fh	8.0	1Ah	3.0
05h	18.5	10h	7.5	1Bh	2.7
06h	17.1	11h	6.7	1Ch	2.5
07h	16.0	12h	6.0	1Dh	2.3
08h	15.0	13h	5.5	1 Eh	2.1
09h	13.3	14h	5.0	1 Fh	2.0
0Ah	12.0	15h	4.6		

Період автоповтору визначає кількість посилок скан-коду, що генеруються процесором клавіатури за одну секунду.

Спочатку при ініціалізації системи період затримки включення режиму автоповтору встановлюється модулями BIOS рівним 500 мс при періоді автоповтору, рівним 10 повторів в секунду.

4.2.3.2 . Команда управління світлодіодами клавіатури.

Команда складається із двох байтів. Перший байт: код команди – EDh. Для ввімкнення або вимкнення світлодіодів після коду команди (EDh) через порт 60h надсилається байт даних, біти якого мають наступне призначення (табл. 4.2):

Таблиця 4.2 - Значення байта управління світлодіодами

Біти	Значення
0	1 - увімкнути ScrollLock ;
1	1 - увімкнути NumLock ;
2	1 - увімкнути CapsLock ;
3-7	не використовуються.

4.2.3.3 . Команда повернення до значень, що задаються BIOS. Код команди – F6h.

4.3. Порядок виконання

4.3.1. Відповідно до індивідуального завдання:
– встановити відповідні період та затримку автоповтору;
- здійснити миготіння відповідного світлодіода.

4.3.2. Шляхом читання з порту 60h скан-коду визначити клавішу, скан-код натискання якої збігається з номером студента в журналі. Визначити скан-код віджимання цієї клавіши. При визначенні скан-кодів можна скористатися алгоритмом, наведеним на рис. 4. 2.

4.3.3. Після виконання пп. 3.1 та 3.2 відновити характеристики клавіатури у вихідний стан.

4.4. Особливості програмування

4.4.1. Мовою Паскаль.

4.4.1.1. Для читання даних із порту та запису даних в порт використовуйте попередньо визначений масив Port. Наприклад, для читання з порту 64h застосовується вираз b:=Port[$64], а для запису в порт 60h - вираз Port[$60]:=b, де b - змінна типу byte.

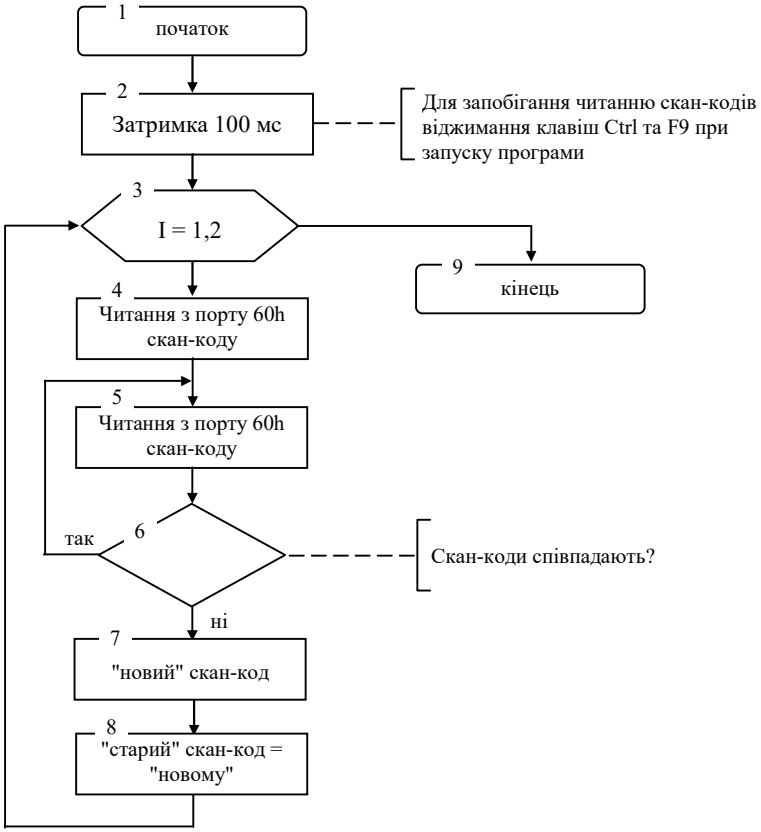


Рис. 4.2 – Алгоритм визначення скан-кодів натискання та віджимання

4.4.2. Мовою С.

4.4.2.1. Для звернення до портів ПЕОМ застосовуються функції читання та запису в порт, що зберігаються у бібліотеці <dos.h>. Бібліотека підключається директивою

#include <dos.h>

після чого, наприклад, для запису значення b в порт 60h використовується вираз:

outportb(0x60,b);

Для читання з порту 64h використовується вираз:

b = intportb (0x64);

де: b - Змінна типу char.

4.5. Індивідуальні завдання

4.5.1. Індивідуальні завдання вибрати відповідно до табл. 4.3:

Таблиця 4. 3 – Варіанти індивідуальних завдань

Номер у журналі	Період автоповтору	Затримка автоповтору, (мс)	Найменування світлодіода
1	30,0	250	NumLock
2	26,7	500	CapsLock
3	24,0	750	ScrollLock
4	20,0	1000	NumLock
5	2,0	250	CapsLock
6	3,0	500	ScrollLock
7	30,0	750	NumLock
8	26,7	1000	CapsLock
9	24,0	250	ScrollLock
10	20,0	500	NumLock
11	2,0	750	CapsLock
12	3,0	1000	ScrollLock
13	5,0	250	NumLock
14	6,0	500	CapsLock
15	8,0	750	ScrollLock
16	10,0	1000	NumLock

4.6. Зміст звіту

4.6.1. Тема лабораторної роботи
4.6.2. Мета роботи.
4.6.3. Індивідуальне завдання.
4.6.4. Текст програми.
4.6.5. Результати роботи програми.
4.6.6. Висновки.

Лабораторна робота 5

Тема: Буфер клавіатури ПК

Мета роботи: Вивчення організації буфера клавіатури та набуття практичних навичок визначення збережених у буфері ASCII та скан-кодів клавіш клавіатури

5.1. Теми для попереднього опрацювання

5.1.1. Організація буфера клавіатури.

5.1.2. Формування кодів ASCII.

5.2. Опис роботи

Записана в ПЗУ BIOS підпрограма обробки переривання 09h, викликаного контролером клавіатури при натисканні клавіші, читає скан-код з порту 60h і, якщо це скан-код клавіші-перемикача (правий Shift, лівий Shift, Ctrl, Alt , CapsLock і Insert), то змінюється вміст комірок пам'яті 0040:0017h і 0040:0018h , що зберігають стани клавіш-перемикачів. У буфер клавіатури нічого не пишеться (виняток становить клавіша Insert).

У табл. 5.1 наведено формат байта за адресою 0040:0017 h , а в табл. 5.2 – за адресою 0040:0018 h .

Таблиця 5.1 – Формат байта за адресою 0040:0017 h

Біт	Значення, коли біт = 1
0	натиснута клавіша правий Shift
1	натиснута клавіша лівий Shift
2	натиснута клавіша Ctrl
3	натиснута клавіша Alt
4	режим ScrollLock увімкнено
5	режим NumLock увімкнено
6	режим CapsLock увімкнено
7	режим вставки увімкнено

Таблиця 5.2 – Формат байта за адресою 0040:0018 h .

Біт	Значення, коли біт = 1
0	натиснені клавіші лівий Shift разом із Ctrl
1	натиснути клавіші лівий Shift разом з Alt
2	натиснута клавіша SysReq
3	натиснуто клавіші Ctrl разом з NumLock
4	режим ScrollLock увімкнено
5	режим NumLock увімкнено
6	режим CapsLock увімкнено
7	режим вставки увімкнено

Статуси клавіш-перемикачів можуть бути змінені програмно.

Якщо ж це скан-код клавіші, що не є перемикачем, то відповідно до її скан-коду та стану клавіш-перемикачів формується або однобайтний ASCII код даної клавіші, який разом зі скан-кодом записується в буфер клавіатури, або двобайтний розширений код, першим байтом якого є 0, а другий байт у більшості випадків збігається із скан-кодом. Ці два байти також записуються в буфер клавіатури. При відтисканні кнопки в буфер нічого не записується і змінюється тільки стан кнопок-перемикачів.

Програма обробки переривання 09h також відстежує деякі комбінації клавіш. У табл. 5.3 наведено ці комбінації та дії, що виконуються обробником переривання при їх виявленні:

Таблиця 5.3 – Функції клавіш

Комбінація клавіш	Дія, що виконуються
Ctrl-Alt-Del	Скидання та перевантаження системи
Ctrl-NumLock	Переведення машини в стан очікування
Shift-PrtSc	Друк на принтері вмісту екрана (обробка переривання 05h)
Ctrl-Break	Завершення виконання програми

Структуру буфера клавіатури зображено на рис. 5.1. Буфер клавіатури розташований за адресами 0040:001Ah – 0040:003Ch оперативної пам'яті та дозволяє накопичувати дані (ASCII та скан-коди) про 15 натискань клавіш.

У комірці 0040:001Ah (покажчик початку) зберігається адреса першого введеного символу, а в комірці 0040:001Ch (покажчик хвоста) – адреса першої вільної комірки після останнього введеного символу.

Адреси зберігаються у вигляді байта зсуву (другий байт комірки не використовується). Якщо буфер порожній, вміст покажчика початку збігається зі вмістом покажчика хвоста.

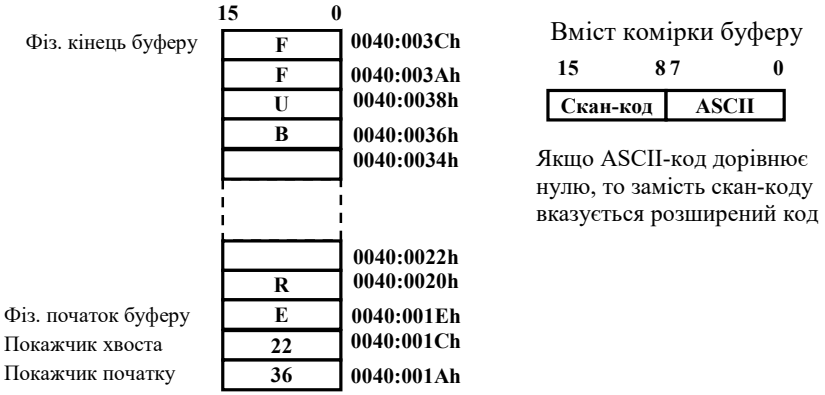


Рис. 5.1 – Структура буфера клавіатури

Запис у буфер здійснюється за адресою покажчика хвоста, при цьому вміст покажчика збільшується на 2. Читання з буфера проводиться за адресою покажчика початку, причому вміст покажчика також збільшується на 2 (якщо покажчик містить адресу останньої комірки буферу – 3Ch, то замість збільшення на 2 до неї необхідно записати адресу початку буфера - 1Eh).

5.3. Порядок виконання

5.3.1. Написати програму, що реалізує зображений на рис. 5.2 алгоритм читання з буфера клавіатури ASCII та скан-кодів.

5.3.2. Відповідно до індивідуального завдання визначити ASCII і скан-код потрібних клавіш.

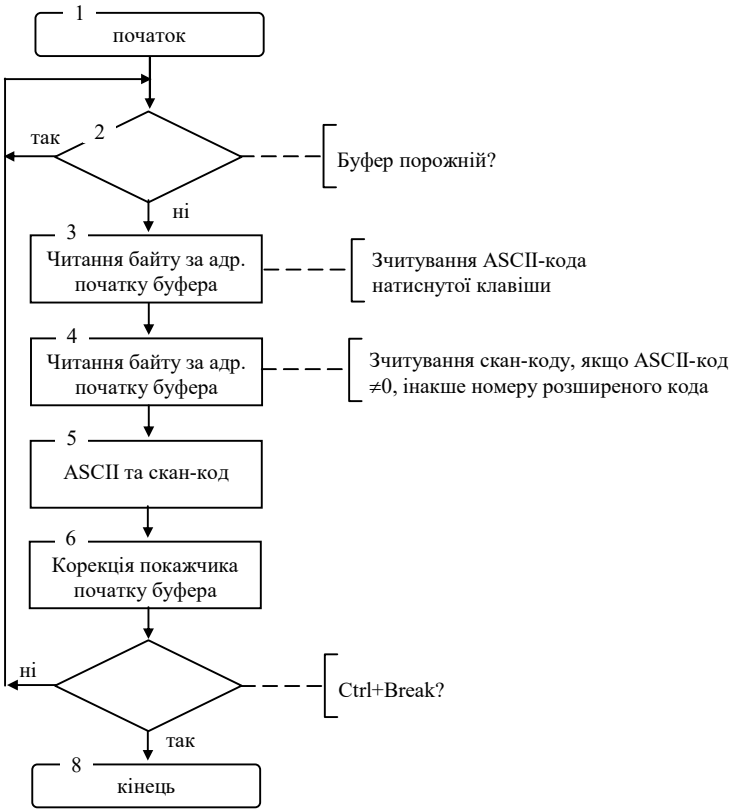


Рис. 5.2 – Алгоритм перегляду буфера клавіатури

5. 4. Особливості програмування

5. 4.1. Мовою Паскаль.

5.4.1.1. Для читання даних із буфера клавіатури використовуйте попередньо визначений масив Mem. Наприклад, для читання вмісту покажчика хвоста застосовується вираз

Mem[$0040:$001C],

а для читання ASCII коду з початку буфера –

Mem[$0040:Mem[$0040:$001A]].

5.4.2. Мовою С.

5.4.2.1. Для читання даних із буфера клавіатури використовуються дальні покажчики, які оголошуються у програмі так:

```
char far *uk;
```

Наприклад, для читання вмісту покажчика хвоста застосовується вираз

```
uk = (char far *) 0x0040001C;
b=*uk;
```

а для читання ASCII коду з початку буфера

```
uk = (char far *) 0x0040001A;
b=*uk;
```

де: b-змінна типу char.

5.5. Індивідуальні завдання

5.5.1. Визначте ASCII код української літери, номер якої в алфавіті збігається з номером студента в журналі.

5.5.2. Визначити скан-код цієї клавіши.

5.6. Зміст звіту

5.6.1. Тема лабораторної роботи
5.6.2. Мета роботи.
5.6.3. Індивідуальне завдання.
5.6.4. Текст програми.
5.6.5. Результати роботи програми.
5.6.6. Висновки.

Лабораторна робота 6

Тема: Структура магнітних дисків

Мета роботи: Вивчення фізичної та логічної структури магнітних дисків та набуття практичних навичок визначення місцезнаходження основних елементів логічної структури дисків – кореневого запису, кореневого каталогу та таблиці розміщення файлів.

6.1. Теми для попереднього опрацювання.

6.1.1. Фізична та логічна структура магнітних дисків.

6.2. Опис роботи.

6.2.1. Фізична структура диску.

Усі диски як гнучкі, так і жорсткі організовані однаковим чином. Дані завжди записуються на магнітній поверхні диска у вигляді концентричних кіл, званих доріжками (циліндрами), а доріжки у свою чергу діляться радіально на сектори. Гнучкі диски мають дві сторони. Жорсткі (фіксовані) диски можуть мати більше двох сторін. Кількість інформації з одного боку диска залежить від кількості доріжок (яка зветься щільністю) і кількості секторів на одній доріжці. Щільність може суттєво змінюватися від диска до диска. Наприклад, диски подвійної щільності мають 40 доріжок, а диски вчетверенної (високої) щільності – 80 доріжок. Якщо кількість доріжок та кількість сторін є конструктивними особливостями дисків, то розташування, число та розмір секторів задаються програмно за початкової розмітки (форматування кожної доріжки диска). Стандартний розмір сектора дорівнює 512 байтам.

Файл розташовують на такій кількості секторів, яка потрібна, щоб вмістити його повністю. Лише кілька секторів на зовнішньому обідку дискети зарезервовані для спеціальних потреб. Інші доступні на основі правила "перший підійшов – першого обслужать". Це означає, що в міру заповнення диска даними сектора поступово заповнюються у напрямку до центру диска. При знищенні файлу сектора звільняються і згодом вільні області стають розкиданими по диску, розбиваючи нові файли та уповільнюючи доступ до них під час читання та запису.

6.2.2. Логічна структура гнучкого диска (флеш-диск).

Процес форматування диска поділяє загальну кількість секторів на чотири частини, що утворюють безперервні сегменти. Ці сегменти (у порядку розташування на диску) називаються завантажувальний запис (Boot Record), таблиця розміщення файлів (FAT – File Allocation Table), каталог (Directory) та простір даних.

6.2.2.1. Завантажувальний запис.

На дискетах перший сектор (доріжка 0, сторона 0, сектор 1) містить запис початкового завантаження, який є невеликою програмою, що дозволяє комп'ютеру зчитувати з дискового накопичувача інші частини операційної системи, а також дані про структуру диска. Формат сектора представлений у табл. 6.1.

Таблиця 6.1 – Формат завантажувального запису (BOOT)

Зсув	Довжина	Познач.	Вміст
+0	3	JMPxxNEAR	перехід на код завантаження
+3	8		ім'я компанії та версія системи
+0 Bh	2	SectSize	число байтів на сектор початок BPB
+0 Dh	1	ClustSize	кількість секторів на кластер
+0 Eh	2	ResSect	число резервних секторів +1 (число секторів перед першою FAT)
+10h	1	FatCnt	кількість копій FAT
+11h	2	RootSiz	максимальна кількість 32-байтових елементів кореневого каталогу
+13h	2	TotSecst	загальна кількість секторів у розділі DOS
+15h	1	Media	дескриптор розділу (те, що 1-й байт FAT)
+16h	2	FatSize	розмір таблиці FAT (кількість секторів)
+18h	2	TrkSec	кількість секторів на доріжці (циліндрі)
+1Ah	2	HeadCnt	число головок читання/запису (сторін)
+1Bh	2	HidnSec	число захованих секторів кінець BPB
+1Eh			початок програми завантаження.

Примітки: підмножина BPB кореневого (BOOT) сектора використовується драйверами пристроїв.

6.2.2.1.1. Адреса області FAT.

Адреса області FAT обчислюється так:

Адреса FAT = початок розділу + кореневий сектор + кількість резервних секторів – 1.

6.2.2.1.2. Адреса кореневого каталогу.

Адреса кореневого каталогу обчислюється таким чином:

Адреса каталогу = початок розділу + кореневий сектор + число резервних секторів + число FAT х число секторів однієї FAT –1.

Необхідно зауважити, що при обчисленні адреси області FAT і каталогу з використанням тривимірних координат для завдання адреси сектора (головка, циліндр, сектор) спочатку змінюється значення сектора, потім голівки, останнім змінюється номер циліндру. Для FAT 32 кореневий каталог може бути необов'язково після таблиці FAT. Початковий кластер кореневого каталогу для FAT32 зберігається починаючи з адреси +2Ch в BOOT секторі і займає 4 байти.

6.2.2.2. Таблиця розміщення файлів (FAT).

FAT розташовується безпосередньо після завантажувального запису, починаючи з сектора 2 на доріжці 0 сторони 0 і містить код формату та повну карту приналежності секторів файлів. ОС використовує FAT для зберігання інформації про розміщення файлів. Кожен елемент таблиці відповідає ділянці дискового простору та містить код стану цієї ділянки: зайнятий, вільний або містить дефект поверхні і тому не використовується. Збереження таблиці розміщення файлів має критичне значення для збереження всього диску, тому на диску зберігаються дві ідентичні копії цієї таблиці.

6.2.2.3. Кореневий каталог.

Каталог являє собою таблицю, в якій кожен елемент відповідає файлу або іншому каталогу і містить інформацію про ім'я файлу, його розмір та ін.

6.2.2.4. Простір даних.

Простір даних використовується для зберігання власне даних і займає більшу частину диска (понад 99% і зі зростанням ємності дисків цей відсоток збільшується).

6.2.3. Логічна структура жорсткого диска.

Логічна структура жіночого диска включає всі елементи структури гнучкого диска, описані в п. 6.2.2 і вони мають те ж саме призначення. Однак, крім цього, жорсткий диск містить у першому секторі (циліндр (доріжка) 0, головка (сторона) 0, сектор 1) програму, яка називається головним кореневим записом. Остання частина цього сектора містить таблицю розділів – 4-елементну таблицю із 16-байтовими елементами. Цією таблицею маніпулює програма розбиття дисків на розділи . Кожен розділ на жорсткому диску має логічне ім'я, наприклад, 1-й розділ "C:", 2-й розділ - "D:" і т.д.

Під час завантаження ROM-BIOS завантажує головний кореневий запис у пам'ять і передає йому керування. Ця програма зчитує таблицю розділів, щоб визначити розділ, позначений активним. Потім у пам'ять зчитується кореневий сектор активного розділу та виконується. У табл. 6.2 наведено структуру таблиці розділів.

Таблиця 6.2 - Структура головного кореневого запису.

Зсув	Довжина	Вміст
+0	1BDh	програма завантаження та виконання завантажувального сектора активного розділу
+1BEh	10h	описувач розділу 1 (див. табл. 6.3)
+1CEh	10h	описувач розділу 2
+1DEh	10 h	описувач розділу 3
+1EEh	10 h	описувач розділу 4
+1FEh	2	55 AA – підпис (сигнатура) таблиці розділів (AA 55 h)

Таблиця 6.3 - Структура описувача розділу

Зсув	Довжина	Вміст
+0	1	Boot Flag - прапор завантаження: 0 = не активний, 80 h = активний.
+1	1	Begin Hd – початок розділу: номер головки (сторони).
+2	2	Begin Sec / Cyl – початок розділу: сектор/циліндр.
+4	1	System Code – код файлової системи (див. табл. 6.4).
+5	1	Ending Hd – кінець розділу: номер головки.
+6	2	Ending Sec / Cyl – кінець розділу: сектор/циліндр останнього сектора.
+8	4	Starting Sector – логічний номер початкового сектора розділу (адреса в системі LBA).
+0 Ch	4	Num Sectors – розмір розділу (кількість секторів)

Зауваження:

Значення циліндра та сектора займають 10 і 6 біт відповідно. Формат слова, що задає значення циліндра та сектора наведено на рис. 6.1 (ц – розряди значення цилінду, Ц – старший розряд; с – розряди значення сектора, С – старший розряд).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ц	ц	ц	ц	ц	ц	ц	ц	Ц	ц	З	з	з	з	з	з

Рис. 6.1 – Формат упакованого сектора/циліндра

Відносний сектор 0 збігається з циліндром 0, головкою 0, сектором 1. Відносний номер сектора збільшується спочатку по кожному сектору на головці, потім по кожній головці і нарешті по кожному циліндру.

Таблиця 6.4 - Значення кодів файлових систем

Код	*Тип розділу (файлова система, максимальний об'єм)*
00 h	Невідомий (неформатований) розділ
01h	DOS FAT 1 2; до 15 Мбайт
02h	XENIX root
03h	XENIX usr
04h	DOS FAT 16; до 32 Мбайт
05h	(DOS) Extended; до 2 Гбайт
06h	DOS FAT16 (Big DOS); до 2 Гбайт
07h	OS/2 HPFS або Windows NT NTFS
08h	AIX
09h	AIX bootable
0 A h	OS/2 Boot Manage
0Bh	Win95 FAT32; 512 Мбайт -2 Тбайт
0 3 h	Win95 FAT32 (LBA); 512 Мбайт -2 Тбайт
0 E h	Win95 FAT16 (LBA); 32 Мбайт -2 Гбайт
0Fh	(Win95) Extended (LBA)
40h	Venix 80286
51h	Novell
52h	Microport
63h	GNUHURD
64h	Novell Netware 2
65h	Novell Netware 3
80h	Old MINIX
81h	Linux/MINIX
82h	LinuxSwap
83h	Linux
85h	Linux extended
93h	Amoeba
94h	Amoeba BBT
A5 h	BSD /386
B7 h	BSDI fs
B8 h	BSDI swap
3 7h	Syrinx
DBh	CP/M
E 1h	DOS access
E3h	DOS R/0
F2h	DOS secondary
FFh	BBT

6.2.4. Читання сектора диска в пам’ять.

Для читання вмісту необхідного сектора у вказану область пам'яті використовується функція 2 переривання 13h. При цьому регістри мають таке призначення:

- ah – номер функції;
- dl – номер дисковода: 0 – "A", 1 – "B", 80h – жорсткий диск;
- dh – номер головки (сторони);
- ch – номер доріжки (циліндра);
- cl – номер сектора;
- al - число секторів;
- es – значення сегмента адреси області пам'яті;
- bx – значення зсуву адреси області пам'яті.

Якщо функція переривання виконана без помилок, то прапор переносу CF (молодший розряд регістра прапорів Flags) встановлений в 0 і регістр ah містить 0. Якщо прапор переносу встановлено в 1, то була помилка і регістр ah містить код помилки (байт стану).

6.3. Порядок виконання

6.3.1. За допомогою функції 2 переривання 13h прочитати в пам’ять вміст першого сектора жорсткого диска (див. п. 6.2.4).

6.3.2. Перевірити правильність виконаної операції.

6.3.3. Використовуючи таблицю розділів жорсткого диска (табл. 6.2), визначити кількість розділів жорсткого диска ПК.

6.3.4. Використовуючи табл. 6.3 визначити початок розділу (номери головки, сектора і циліндра), тобто адресу кореневого BOOT сектора. Вивести на екран вміст таблиці розділів.

6.3.5. Використовуючи таблицю 6.3 та 6.4, Визначити тип файлової системи.

6.3.6. За допомогою функції 2 переривання 13h прочитати в пам'ять вміст кореневого BOOT сектора обраного розділу і перевірити правильність операції. Вивести на екран вміст BOOT сектора відповідно до табл. 6.1.

6.3.7. Використовуючи табл. 6.1 визначити адресу (номери головки, циліндра та сектора) області FAT та кореневого каталогу.

6.4. Особливості програмування

6.4.1. Мовою Паскаль.

6.4.1.1. При використанні програмного переривання необхідно:

– підключити модуль Dos, в якому описано процедуру Intr та тип змінної

Registers;
– оголосити змінну цього типу, наприклад, reg:Registers;
– до регістрів мікропроцесора звертатися, як reg.ah, reg.al тощо;
– процедуру переривання 13h викликати так:
Intr($13,reg).

6.4.1.2. Область пам'яті, куди зчитується вміст сектора, зручно оголосити як масив байтів:

```
var
    sect:array[0..511] of byte;
```

Тоді необхідні значення сегмента та зсув адреси області пам'яті можна задати, використовуючи функції Seg і Ofs:

```
reg.es: = Seg (sect);
reg.bx:=Ofs(sect);
```

6.4.2. Мовою С.

6.4.2.1. При використанні програмного переривання необхідно:
– підключити бібліотеку Dos, в якій описано процедуру Int86x та тип суміші REGS директивою:

```
#include <dos.h>
```

– оголосити змінні суміші:
union REGS in, out, sr;

– до регістрів мікропроцесора звертатися, як in.h.ah, in.x.ax;
– до сегментних регістрів звертатися як sr.es;
– процедуру переривання 13 h викликати так:

```
int86x(0x13,&in,&out,&sr).
```

6.4.2.2. Область пам'яті, куди зчитується вміст сектора, зручно оголосити як масив байтів:

```
unsigned char buf[512];
```

Тоді необхідні значення сегмента та зсуву адреси області пам'яті можна задати, використовуючи функції FP_SEG() та FP_OFF():

```
sr.es=FP_SEG(buf);
in.x.bx=FP_OFF(buf);
```

6.5. Індивідуальні завдання
Індивідуальні завдання у цій лабораторній роботі визначено у п. 6.3.

6.6. Зміст звіту

6.6.1. Тема лабораторної роботи
6.6.2. Мета роботи.
6.6.3. Індивідуальне завдання.
6.6.4. Текст програми.
6.6.5. Результати роботи програми.
6.6.6. Висновки.

Лабораторна робота 7

Тема: Структура кореневого каталогу

Мета роботи: Вивчення структури кореневого каталогу та набуття практичних навичок роботи з його окремими елементами.

7.1. Теми для попереднього опрацювання

7.1.1. Організація кореневого каталогу.

7.2. Опис роботи

7.2.1. Структура кореневого каталогу.

Кожен диск з FAT має один кореневий каталог. Розташування кореневого каталогу на диску фіксовано та визначається типом та форматом диска. Адресу каталогу можна визначити за допомогою таблиці розділів (тільки для жорстких дисків) та таблиці BOOT сектора (див. лаб. раб. 6. Структура дисків).

Кореневий каталог використовується для зберігання інформації про файли, включаючи ім'я файлу, його розмір, початковий елемент таблиці розміщення, що стосується даного файлу, дату та час його створення або модифікації та атрибути файлу (див. табл. 7.1).

Каталог є таблицею, в якій кожному файлу на диску (включаючи підкаталоги та мітку тома) відповідає один запис. Цей запис має розмір 32 байти, отже на одному секторі розміром 512 байтів міститься 16 записів (елементів) каталогу.

Таблиця 7. 1 – Поля запису каталогу

Поле	Зміщ.	Опис	Розмір	Формат
1	0	Ім'я файлу	8	Символи ASCII
2	8	Розширення файлу	3	Символи ASCII
3	11	Атрибути	1	Біти байта
4	12	Службове поле	10	Не використовується
5	22	Час створення	2	Слово
6	24	Дата створення	2	Слово
7	26	Початковий кластер	2	Слово
8	28	Розмір файлу	4	Довге ціле

7.2.2. Поля запису каталогу.

7.2.2.1. Ім'я файлу.

Перші вісім байт запису каталогу містять ім'я файлу у форматі ASCII. Якщо ім'я файлу коротше восьми символів, воно доповнюється праворуч пробілами (ASCII код – 32). Літери в імені обов'язково мають бути великими.

Існує три особливі ситуації, що відзначаються спеціальними значеннями першого байта в імені файлу.

Елементи каталогу, що не використовуються, мають перший байт рівний 0. DOS припиняє перегляд каталогу, як тільки зустріне такий елемент. Ту саму ознаку може використовувати програміст під час роботи з каталогом як ознаку закінчення елементів каталога.

Якщо перший байт імені файлу дорівнює E5h, це означає, що цей файл знищений.

Знищення файлу на диску, власне, означає дві речі:

- перший байт імені встановлюється рівним E5h;

– всі кластери, які стосуються цього файлу, позначаються у таблиці розміщення файлів як вільні.

Вся інша інформація про цей файл, включаючи залишок імені, розмір файлу , його початковий кластер, залишається в каталозі.

Третій особливий випадок - коли в якості першого байту імені використовується символ "." (ASCII код – 2h). Цей символ застосовується для вказівки підкаталогу. Якщо і другий символ імені - точка, тоді цей елемент каталогу відповідає катологу-батькові поточного, а поле початкового кластера вказує розташування каталогу-батька.

7.2.2.2. Розширення імені файлу.

Безпосередньо за іменем файлу слідує розширення імені файлу, записане у форматі ASCII. Довжина його становить три байти і подібно до імені файлу, якщо розширення імені складається менше ніж з трьох символів, воно доповнюється пробілами. Однак, якщо ім'я файлу має складатися хоча б з одного символу, розширення може мати лише пробіли.

7.2.2.3. Атрибут файлу.

Формат байта атрибутів файлу наведено в табл. 7.2.

Таблиця 7.2 - Формат байта атрибутів файлу

Біти	Значення, коли біт = 1
0	Дозволено лише читання
1	Прихований
2	Системний
3	Мітка тому
4	Підкаталог
5	Архів
6	Не використовується
7	Не використовується

Якщо дозволено лише читання файлу, то в цьому випадку файл захищений від будь-якої зміни або видалення за допомогою будь-яких дій операційної системи.

Біти 1 і 2 маркують файл як прихований чи системний. Файли позначені як приховані чи системні, або й ті й інші, не можна побачити за допомогою простої операції DOS на кшталт DIR (щоб побачити такий файл необхідно змінити його атрибут). Атрибут "системний" не має якогось особливого значення, він залишився у спадок від операційної системи CP/M і в DOS абсолютно нічого не робить.

Біт 3 маркує елемент каталогу як мітку, що означає, що цей елемент містить мітку тома диска, що ідентифікує. Мітка записується в полях імені та розширення імені файлу, які в даному випадку розглядаються як об'єднане поле. Поля розміру та початкового кластера не використовуються, хоча поля дати та часу використовуються.

Біт 4 маркує елемент каталогу як підкаталог. Підкаталоги записуються на диску подібно до звичайних файлів. Для елемента каталогу такого типу використовуються всі поля, за винятком поля розміру файлу, що дорівнює нулю. Справжній розмір підкаталогу визначається переглядом його FAT.

Біт 5, атрибут архіву був створений для того, щоб допомогти скопіюванню великої кількості файлів, що зберігаються на жорсткому диску. Цей розряд скинутий для всіх файлів, які змінилися після останнього копіювання. Для дискет атрибут архіву практично марний.

7.2.2.4. Службове поле.

FAT32 використовується для зберігання символів «довгого імені файлу». Усі 10 байтів поля зазвичай дорівнюють 0 (для файлів у форматі «8.3»).

7.2.2.5. Час створення файлу чи його останньої модифікації.

Поле 5 містить 2-байтове значення, що вказує час створення або останньої зміни файлу. Разом із полем дати ці два поля можуть розглядатися

як одне 4-байтове беззнакове число.
Це число може порівнюватися з тими самими числами в інших елементах каталогу на більше, менше або рівне. Формат поля часу та вирази для виділення годин, хвилин та 2-секундних одиниць (для зберігання повних секунд не вистачило одного біта) представлені на рис. 7.1.

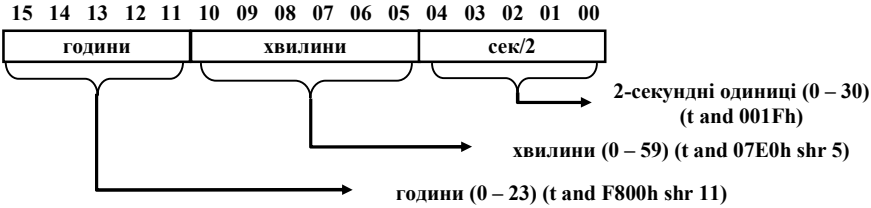


Рис. 7.1 – Формат поля часу

7.2.2.6. Дата створення файлу чи його останньої модифікації.

Поле 6 містить 2-байтове значення, що вказує на дату створення або зміни файлу. Разом із полем часу ці два поля можуть розглядатися як одне 4-байтове беззнакове число. Це число може порівнюватися з тими самими числами в інших елементах каталогу на більше, менше або рівне. Формат поля дати та виразу для виділення дня, місяця та року представлені на рис. 7.2.

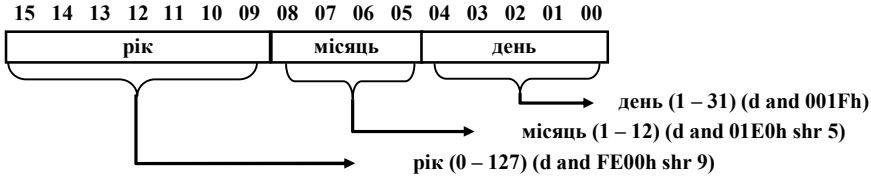


Рис.7.2 – Формат поля дати

Для отримання істинного значення року треба до значення року, виділеного з дати додати 1980 року.

7.2.2.7. Номер початкового кластера

Кластер – це група суміжних секторів. Величина кластера, яка для різних типів дисків і форматів може бути різною, вказується в таблиці кореневого сектора.

Поле 7 - це 2-байтове значення, яке дає номер початкового кластера для

області даних файлу. Воно діє як точка входу в ланцюжок зайнятого простору файлу, що у FAT. Для файлів, які не займають області даних, а також для міток томів номер початкового кластера дорівнює нулю.

7.2.2.8. Розмір файла.

Останнє поле елемента файлу дає розмір файлу в байтах. Зазначимо, що коли DOS читає файл, вона вказує на кінець файлу, коли перевищує розмір файлу або коли досягає кінця ланцюжка зайнятого простору FAT в залежності від того, що першим буде виявлено.

7.2.3. Читання сектора диска в пам'ять.

Для читання вмісту необхідного сектора у вказану область пам'яті використовується функція 2 переривання 13h. При цьому регістри мають таке призначення:

ah – номер функції;
dl – номер дисковода: 0 – "A", 1 – "B", 80h – жорсткий диск;
dh – номер головки (сторони);
ch – номер доріжки (циліндра);
cl – номер сектора;
al - число секторів;
es – значення сегмента адреси області пам'яті;
bx – значення усунення адреси області пам'яті.

Якщо функція переривання виконана без помилок, то прапор переносу CF (молодший розряд регістра прапорів Flags) буде встановлений в 0 і регістр ah містить 0. Якщо прапор переносу встановлено в 1, то була помилка і регістр ah містить код помилки (байт стану).

7.3. Порядок виконання

7. .1. За допомогою функції 2 переривання 13h прочитати в пам'ять вміст першого сектора кореневого каталогу (адресу кореневого каталогу було визначено в лаб. раб. 6 Структура дисків).

7.3.2. Перевірити правильність виконаної операції.

7.3.3. Для всіх файлів цього сектора визначити та вивести на екран такі дані файлів:

- Ім'я файлу та його розширення;
– Атрибути;
- Номер початкового кластера;

– Час та дату створення;
- Розмір.

7.3.4. Порівняти отриману інформацію про файли з інформацією операційної системи.

7.4. Особливості програмування

7.4.1. Мовою Паскаль.

7.4.1.1. При використанні програмного переривання необхідно:
– підключити модуль Dos, в якій описано процедуру Intr та тип змінної Registers;
– оголосити змінну цього типу, наприклад, reg:Registers;
– до регістрів мікропроцесора звертатися, як reg.ah, reg.al тощо;
– процедуру переривання 13h викликати так:

Intr($13,reg).

7.4.1.2. Область пам'яті, куди зчитується вміст сектора каталогу, зручно оголосити як масив записів (див. табл. 7.1):

```
var
sec_cat:array[1..16] of record
name :array[1..8] of char;
name_e :array[1..3] of char;
atr: byte;
rez :array[1..10] of byte;
time,
date,
n_clast: Word;
 size :longint
end;
```

Необхідні значення сегмента та зсуву адреси області пам'яті можна задати, використовуючи функції Seg і Ofs:

```
reg.es:=Seg(sec_cat);
reg.bx:=Ofs(sec_cat);
```

7.4.1.3. Для виділення полів часу та дати використовувати вирази, наведені на рис. 7.1 та рис. 7.2. Символу & в Паскалі відповідає операція AND,

а символу >> - Shr (зсув праворуч).

7.4.2. Мовою Турбо-С.

7.4.2.1. При використанні програмного переривання необхідно:
– підключити бібліотеку Dos, в якій описано процедуру int86x та тип суміші REGS директивою:

#include <dos.h>

– оголосити змінні суміші:

union REGS in, out, sr;

– до регістрів мікропроцесора звертатися, як in.h.ah, in.x.ax;
– до сегментних регістрів звертатися як sr.es;
– процедуру переривання 13 h викликати так:

int86x(0x13,&in,&out,&sr).

7.4.2.2. Область пам'яті, куди зчитується вміст сектора каталогу, зручно оголосити як шаблон структури (див. табл. 7.1):

```
struct sec_cat {
unsigned char name[8];
unsigned char name_e[3];
unsigned char atr;
unsigned char rez[10];
unsigned int time;
unsigned int date;
unsigned int n_clast;
unsigned long size;
 };
```

та визначити масив структур

struct sec_cat sec [16];

Тоді необхідні значення сегмента та зсуву адреси області пам'яті можна встановити, використовуючи функції FP_SEG() і FP_OFF():

sr.es=FP_SEG(sec);

in.x.bx=FP_OFF(sec);

7.4.2.3. Для виділення полів часу та дати використовувати вирази, наведені на рис. 1 та рис. 2.

7.5. Індивідуальні завдання

Індивідуальні завдання у цій лабораторній роботі визначено у п.7.3.

7.6. Зміст звіту

7.6.1. Тема лабораторної роботи
7.6.2. Мета роботи.
7.6.3. Індивідуальне завдання.
7.6.4. Текст програми.
7.6.5. Результати роботи програми.
7.6.6. Висновки.

Лабораторна робота 8

Тема: Структура таблиці розміщення файлів FAT

Мета роботи: Вивчення структури таблиці розміщення файлів та набуття практичних навичок визначення послідовностей кластерів, що належать одному файлу.

8.1. Теми для попереднього опрацювання

8.1.1.Організація таблиці розміщення файлів FAT.

8.2. Опис роботи

8.2.1. Призначення FAT.

Диск використовує таблицю розміщення файлів (FAT) для відведення дискового простору файлам та зберігання інформації про вільні сектори. З міркувань безпеки на всіх дисках зберігаються дві копії FAT. Вони зберігаються послідовно з першого сектора FAT. Для дискети першим сектором FAT буде сектор, що прямує за кореневим сектором, тобто, сектор з координатами: доріжка 0, сторона 0, сектор 2. Для жорсткого диска перший сектор FAT визначається за допомогою таблиці розділів та таблиці кореневого сектора (див. лаб. роб. 6 Структура дисків). Число секторів, що займає FAT, визначається типом і форматом диска і вказується в таблиці BOOT сектора.

Таблиця розміщення файлів зберігає інформацію про кожен кластер файлу на диску. Кластер – це група суміжних секторів. Величина кластера, яка може бути різною для різних типів дисків і форматів, вказується в таблиці BOOT сектора. Кластери були введені для зменшення розміру FAT.

8.2.2. Організація FAT.

Кожна позиція таблиці розміщення файлів відповідає певної позиції кластера на диску. Зазвичай файл займає кілька кластерів і запис у каталозі файлів містить номер стартового кластера, де знаходиться початок файла. Подивившись позицію FAT, що відповідає першому кластеру, DOS знаходить номер кластера, у якому зберігається наступна порція файлу. Цьому кластеру відповідає свій запис у FAT, який у свою чергу містить номер наступного кластера в ланцюжку. Для останнього кластера, зайнятого файлом, FAT містить значення від FFF8h до FFFFh (FAT 16). Кластерам, що не використовуються (або звільненим), відповідає значення 0, а дефектним секторам – FFF7h. Нарешті значення від FFF0h до FFF7h приписуються резервним кластерам.

На рис. 8.1 наведено елемент кореневого каталогу, що описує файл з ім'ям MYFILE.TXT, та фрагмент таблиці розміщення файлів, що визначає розміщення цього файлу на диску:

Рис. 8.1 – Основні концепції FAT

Цей рисунок ілюструє основні концепції FAT.

З нього видно, що

– Файл MYFILE.TXT займає 10 кластерів. Перший кластер – це кластер 08, останній кластер – 1Bh. Ланцюжок кластерів – 8, 9, 0A, 0B, 15, 16, 17, 19, 1A, 1B. Кожен елемент вказує наступний елемент ланцюжка, а останній елемент містить спеціальний код.

– Кластер 18h позначений як поганий і не входить до ланцюжка розподілу.

– Кластери 6,7, 0Ch-14h та 1Ch-1Fh порожні та доступні для розподілу.

- Ще один ланцюжок починається з кластера 2 і закінчується кластером 5.

Щоб дізнатися ім'я файлу, необхідно знайти елемент кореневого каталогу з початковим номером кластера 02.

Для всіх гнучких дисків розмір комірки таблиці розміщення файлів дорівнює 12 бітів, що дозволяє звертатися до 4096 кластерів. Тип використовуваної FAT вказаний у таблиці розділів.

8.2.3. Читання сектора диска у пам'ять.

Для читання вмісту необхідного сектора у вказану область пам'яті використовується функція 2 переривання 13h.

Якщо функція переривання виконана без помилок, то прапор переносу CF (молодший розряд регістра прапорів Flags) встановлений в 0 і регістр AH

містить 0. Якщо прапор переносу встановлено в 1, то була помилка і регістр AH містить код помилки (байт стану).

8.3. Порядок виконання

8.3.1. За допомогою функції 2 переривання 13h прочитати у пам'ять вміст першого сектора таблиці розміщення файлів (адресу FAT було визначено в лаб. роб. 6 Структура магнітних дисків).

8.3.2. Перевірити правильність виконаної операції.

8.3.3. Використовуючи номери початкових кластерів файлів (номер файлу відповідає номеру за списком у журналі групи), отриманих у лаб. роб. 7 Структура кореневого каталогу, за таблицею розміщення файлів побудувати повні ланцюжки кластерів для цих файлів.

8.4. Особливості програмування.

8.4.1. Мовою Паскаль.

8.4.1.1. При використанні програмного переривання необхідно:

– підключити модуль DOS, в якому описано процедуру Intr та тип змінної Registers;

– оголосити змінну цього типу, наприклад, reg:Registers;

– до регістрів мікропроцесора звертатися, як reg.ah, reg.al тощо;

– процедуру переривання 13h викликати так:

Intr($13,reg).

8.4.1.2. Область пам'яті, куди зчитується вміст сектора, зручно оголосити як масив байтів:

```
 var
fat:array[0..511] of byte;
```

Необхідні значення сегмента та зсуву адреси області пам'яті можна задати, використовуючи функції Seg і Ofs:

```
reg.es: = Seg (sect);
reg.bx:=Ofs(sect);
```

8.4.2. Мовою С.

8.4.2.1.При використанні програмного переривання необхідно:
– підключити бібліотеку DOS, в якій описано процедуру int86x та тип суміші REGS директивою:

#include <dos.h>

– оголосити змінні суміші:

union REGS in, out, sr;

– до регістрів мікропроцесора звертатися, як in.h.ah, in.x.ax;
– до сегментних регістрів звертатися як sr.es;
– процедуру переривання 13h викликати так:
int86x(0x13,&in,&out,&sr).

8.4.2.2. Область пам'яті, куди зчитується вміст сектора, зручно оголосити як масив байтів:
unsigned char buf[512];
Тоді необхідні значення сегмента та зсуву адреси області пам'яті можна задати, використовуючи функції FP_SEG() та FP_OFF():

sr.es=FP_SEG(buf);
in.x.bx=FP_OFF(buf);

8.5. Індивідуальні завдання

Індивідуальні завдання у цій лабораторній роботі визначено у п.8.3.

8.6. Зміст звіту

8.6.1. Тема лабораторної роботи
8.6.2. Мета роботи.
8.6.3. Індивідуальне завдання.
8.6.4. Текст програми.
8.6.5. Результати роботи програми.
8.6.6. Висновки.

Лабораторна робота №9

Системний таймер. Генерація звуку

Мета роботи: Вивчення функцій системного таймера та набуття практичних навичок роботи з таймером та динаміком при генерації звуку.

9.1. Теми попереднього опрацювання

9.1.1. Структура та призначення портів мікросхеми конфігурації та системного таймера.

9.2. Опис роботи

9.2.1. Системний таймер

Усі комп'ютери IBM містять пристрій, який називається системним таймером. Цей пристрій підключено до лінії запиту на переривання IRQ0 і виробляє переривання INT 8h приблизно 18,2 разів на секунду (точне значення – 1193180/65535 разів на секунду).

При ініціалізації BIOS встановлює свій обробник для переривання таймера. Ця програма щоразу збільшує на 1 поточне значення 4-байтової змінної, що знаходиться в області даних BIOS за адресою 0040:006Ch - лічильника тиків таймера. Якщо цей лічильник переповнюється (тобто минуло понад 24 години з моменту запуску таймера), в комірку 0040:0070h заноситься 1.

Стандартний обробник переривання таймера здійснює контроль за роботою двигунів НГМД. Якщо після останнього звернення до НГМД пройшло більше 2 секунд, обробник переривання вимикає двигун (комірка з адресою 0040:0040h містить час, що залишився до вимкнення двигуна).

Остання дія, яку виконує обробник переривання таймера, це виклик переривання INT 1Ch. Після ініціалізації системи вектор INT 1Ch свідчить про команду IRET, тобто, нічого не виконується. Програма користувача може встановити власний обробник цього переривання для того, щоб виконати будь-які періодичні дії.

Необхідно відзначити, що переривання INT 1Ch викликається до скидання контролера переривань, тому під час виконання переривання INT 1Ch всі апаратні переривання заборонені. Для скидання контролера переривань треба до порту 20 h записати значення 20 h .

На рис. 9.1 показано механізм обробки переривання таймера.

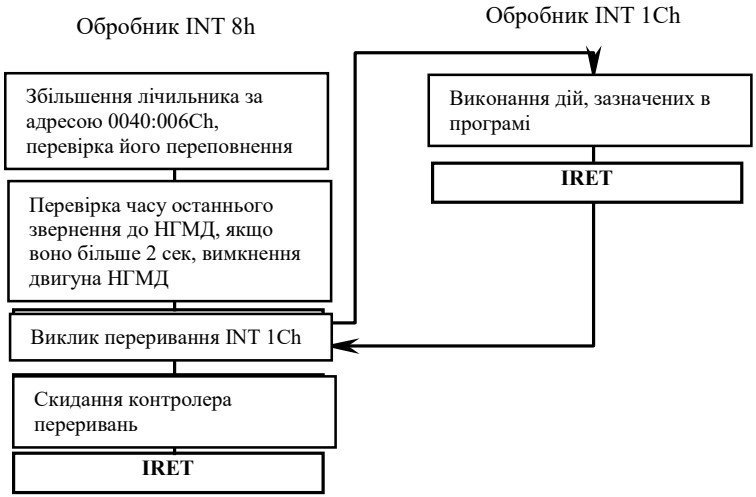


Рис. 9.1 - Алгоритм обробника переривання від таймера

Таймер зазвичай реалізується на мікросхемі Intel 8253 (для IBM PC та IBM XT) або 8254 (для IBM AT та IBM PS/2)

9.2.2. Склад та режими роботи мікросхеми таймера.

Таймери 8253 та 8254 складаються з трьох незалежних каналів. Кожен канал містить регістри:

– стан каналу RS (8 розрядів);
– керуюче слово PSW (8 розрядів);
- буферний регістр OL (16 розрядів);
– регістр лічильника CE (16 розрядів);
– регістр констант перерахунку CR (16 розрядів).

Канали таймера підключаються до зовнішніх пристроїв за допомогою трьох ліній:

GATE - керуючий вхід;
CLOCK – вхід тактової частоти;
OUT – вихід таймера.

Регістр лічильника CE працює у режимі віднімання. Його вміст зменшується по задньому фронту сигналу CLOCK за умови, що на вході

GATE встановлено рівень логічної 1. Залежно від режиму роботи таймера при досягненні лічильником CE нуля тим чи іншим способом змінюється вихідний сигнал OUT.

Буферний регістр OL призначений для запам'ятовування поточного значення регістру лічильника CE без зупинки процесу рахунку. Після запам'ятовування буферний регістр доступний програмі читання.

Регістр констант перерахунку CR може завантажуватись у регістр лічильника, якщо це потрібно в поточному режимі роботи таймера.

Можливі 6 режимів роботи таймера. Вони поділяються на три типи:

- режими 0 і 4-одноразове виконання функцій;
- режими 1 та 5 - робота з перезапуском;
- режими 2 та 3 - робота з автозавантаженням.

У режимі одноразового виконання функцій перед початком рахунку вміст регістру констант CR переписується в регістр лічильника CE за сигналом CLOCK, якщо сигнал GATE встановлений в 1. Надалі вміст регістра CE зменшується в міру надходження сигналів CLOCK.

Процес рахунка можна призупинити, якщо подати на вхід GATE рівень логічного 0. Якщо потім на вхід GATE подати 1, рахунок буде продовжено далі. Для повторення виконання функції потрібно нове завантаження регістра CR, тобто, повторне програмування таймера.

При перезапуску не потрібно повторного програмування таймера для виконання тієї ж функції. По фронту сигналу GATE значення константи з регістра CR знову переписується в регістр CE, навіть якщо поточна операція не була завершена.

У режимі автозавантаження регістр CR автоматично переписується в регістр CE після завершення рахунку. Сигнал на виході OUT з'являється лише за наявності на вході GATE рівня логічного 1. Цей режим використовується для створення програмованих імпульсних генераторів і прямокутних генераторів імпульсів (меандрів).

У комп'ютерах IBM PC/XT/AT/PS2 задіяні всі три канали таймера.

Канал 0 використовується операційною системою для формування часових інтервалів. Цей канал працює в режимі 3 і використовується як генератор імпульсів із частотою приблизно 18,2 Гц. Саме ці імпульси викликають апаратне переривання INT 8 h .

Канал 1 використовується для регенерації вмісту динамічної пам'яті комп'ютера, тому краще не чіпати його. Вихідна лінія каналу OUT пов'язана з мікросхемою прямого доступу до пам'яті (DMA) і її імпульс змушує DMA регенерувати пам'ять.

Канал 2 пов'язаний з динаміком комп'ютера і може бути використаний для генерації різних звуків або музики або як генератор випадкових чисел. Канал використовує режим 3 таймери.

9.2.3. Програмування таймера.

Таймеру відповідають 4 порти вводу/виводу з наступними адресами: 40 h - канал 0; 41 h - канал 1; 42 h - канал 2; 43 h - керуючий регістр. Формат байта керуючого регістру (керуючого слова) наведено у табл. 9.1.

Таблиця 9.1 - Формат керуючого регістру

Біти	Значення
0	0 – двійкові дані; 1 - дані у формі BCD (двійково-десяткові).
3-1	номер режиму: 000 - режим 0 (переривання від таймера); 001 - режим 1 (програмований мультивібратор); X01 - режим 2 (програмований генератор імпульсів); X11 – режим 3 (генератор меандру); 100 - режим 4 (програмно-запускний одновібратор); 101 – режим 5 (апаратно-запускний одновібратор).
5-4	тип операції: 00 - передати значення лічильника в буфер (читання на льоту); 01 – читати/писати лише старший байт; 10 – читати/писати лише молодший байт; 11 – читати/писати спочатку молодший байт, потім – старший.
7-6	номер каналу: 00 – канал 0; 00 - канал 0; 01 - канал 1; 10 - канал 2; 11 – код команди RBC (зворотне читання).

Молодший розряд байта визначає формат константи, що використовується для рахунку. У двійково-десятковому режимі константа задається в діапазоні 1-9999.

Для програмування каналу таймера необхідно виконати таку послідовність дій:

- записати в керуючий регістр за адресою 43 h керуюче слово;
- необхідне значення лічильника надсилається до порту каналу (адреси 40 h – 42 h), причому спочатку посилається молодший байт, а потім старший байт значення лічильника.

Відразу після цього канал таймера починає виконувати потрібну функцію.

9.2.4. Генерація звуків та музики.

Звуки та музику на ПЕОМ можна відтворювати двома способами:

- із використанням таймера;
- без використання таймера.

У будь-якому випадку при надходженні на динамік електричного сигналу деякого рівня відбувається втягування мембрани динаміка, а при зміні рівня електричного сигналу виштовхування. Отже періодичний електричний сигнал викликає вібрацію мембрани динаміка, тобто, генерацію звуку тієї самої частоти.

Зазначимо, що у всіх моделях IBM, за винятком IBM PCjr, яка має (крім динаміка) спеціальний звуковий генератор, гучність звуку не змінюється.

9.2.4.1. Керування динаміком за допомогою таймера.

Одне з найпоширеніших застосувань таймера – генерація звукових сигналів та відтворення музики. Таймер дозволяє відтворювати музику у фоновому режимі, тобто, під час роботи програми може звучати музика. Як зазначалося вище, канал 2 мікросхеми 8254 пов'язаний з динаміком. Однак динамік не просто з'єднаний із виходом OUT каналу 2. Порт 61 h також використовується для керування динаміком (рис. 9.2).

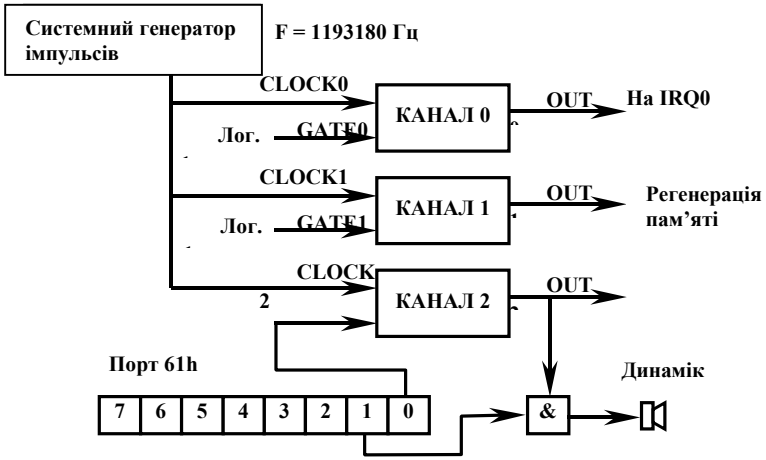


Рис. 9.2 - Схема таймера

Молодший біт цього порту при встановленні 1 дозволяє роботу каналу, тобто, генерацію імпульсів динаміка. Додатково для керування динаміком

використовується біт 1 порту 61 h . Якщо цей біт встановлений 1, імпульси від каналу 2 таймера зможуть проходити на динамік.

Системний генератор імпульсів (СДІ) незалежно від типу та продуктивності комп'ютера IBM виробляє імпульси однієї і тієї ж частоти - 1193180 Гц.

Таким чином, для включення звуку треба виконати такі дії:

- запрограмувати канал 2 таймера на потрібну частоту (тобто, завантажити в регістр лічильника каналу значення, що дорівнює відношенню частоти СГІ та необхідної частоти звуку);
- для включення звуку встановити в 1 два молодші розряди порту 61 h.

Оскільки інші розряди порту 61 h використовуються для інших цілей, вони повинні залишатися без зміни.

Для вимкнення звуку треба скинути два молодших розряди порту 61 h (не змінюючи знову ж таки стану інших розрядів).

9.2.4.2. Управління динаміком без таймера.

Для генерації звуку без таймера треба скинути молодший біт порту 61h (заборонивши цим роботу каналу 2 таймера) і, керуючи бітом 1 цього порту, тобто, встановлюючи цей біт то в 1, то в 0, формувати імпульси для динаміка. Висота звуку, що генерується, буде відповідати періоду імпульсів. Зазначимо, що цим способом можна генерувати імпульси будь-якої шпаруватості, що дає більше можливостей для створення різних звукових ефектів.

9.2.4.3. Відтворення музики.

Мелодія, як відомо, складається з нот, розділених або не розділених паузами. При програванні мелодії необхідно для кожної ноти програмувати відповідним чином канал 2 таймера та включати динамік (за допомогою порту 61h) на певний час, що дорівнює тривалості ноти. Потім програма повинна вимкнути динамік і витримати паузу, якщо потрібно, перед програванням наступної ноти. У табл. 9.2 наведено частоти для нот 2-ї октави. Для інших октав при зниженні чи підвищенні тону треба значення частот ділити чи множити на два.

Таблиця 9.2. Частоти нот другої октави

Нота	Частота, Гц	Нота	Частота, Гц
До	267,7	Фа-дієз	370,0
До-дієз	277,2	Соль	392,0
Ре	293,7	Соль-дієз	415,3
Ре-дієз	311,1	Ля	440,0
Ми	329,6	Ля-дієз	466,2
Фа	349,2	Сі	493,9

Плавні переходи тонів виробляються за допомогою безперервної зміни частоти. Такий звуковий ефект можна зробити більш виразним, якщо зменшити тривалість кожного тону при підвищенні звуку або злегка збільшувати тривалість при зниженні.

Ви можете використовувати можливість керування динаміком для створення різних ефектів, які, зокрема, роблять відеоігри дуже цікавими та привабливими. В основі всіх звукових ефектів лежить варіювання частоти звуку - часто незвичайним чином.

Наприклад, для створення ефекту звучання сирени необхідно варіювати частоту звуку між двома кінцевими точками. Висота звуку повинна змінюватися від меншої до більшої, а потім зменшуватись від більшої до меншої.

Для імітації звуку вибуху, який використовується у багатьох відеоіграх, можна модифікувати звучання сирени таким чином, щоб вона дозволяла генерувати звук лише збільшуючи частоти.

9.3. Порядок виконання

9.3.1. Відтворити "мелодію" згідно з індивідуальним завданням.

Для цього необхідно:

- програмувати канал 2 таймера в режимі 3 на потрібну частоту (що визначається нотою, див. табл. 9.2);
- за допомогою порту 61 h задавати необхідну тривалість звучання ноти та тривалість паузи (тривалість ноти у секундах задана в індивідуальному завданні після найменування ноти).

9.3.2. Відтворити звуковий ефект без використання таймера.

9.4. Особливості програмування

9.4.1. Мовою Турбо-Паскаль.

9.4.1.1. Для виділення молодшого та старшого байта деякого слова зручно використовувати функції Lo та Hi:

Lo(w) - виділяє молодший байт слова w;

Hi(w) – виділяє старший байт слова w, де w:word.

9.4.1.2. Для запису даних у порт використовуйте попередньо визначений масив Port. Наприклад, для запису молодшого байта змінної kd в регістр лічильника каналу 2 можна застосувати вираз Port[$42]:=Lo(kd), де kd – змінна типу word, що задає коефіцієнт розподілу частоти системного генератора на потрібну частоту.

9.4.1.3. Для встановлення деяких бітів слова або байта без зміни інших бітів використовуйте операцію OR з маскою, що містить 1 в потрібних бітах і 0 в інших; для скидання – операцію AND із маскою, що містить 0 у необхідних бітах і 1 в інших.

9.4.1.4. Для виконання програмної затримки використовуйте процедуру Delay, описану у модулі Crt. Аргумент процедури визначається в мс, наприклад, Delay(1000) означає затримку в 1 секунду.

9.4.2. Мовою Турбо-С.

9.4.2.1. Для запису даних у порт застосовуються функції запису у порт, які зберігаються у бібліотеці <dos.h>. Бібліотека підключається директивою

```
#include <dos.h>
```

після чого, наприклад, для запису молодшого байта змінної kd в регістр лічильника каналу 2 можна застосувати вираз

```
outportb(0x42,kd);
```

де kd – змінна типу int, що задає коефіцієнт розподілу частоти системного генератора на потрібну частоту.

9.4.2.2. Для встановлення деяких бітів слова або байта без зміни інших бітів використовуйте операцію | з маскою, що містить 1 у потрібних бітах і 0 в інших; для скидання - операцію & з маскою, що містить 0 у необхідних бітах та 1 в інших.

9.4.2.3. Для виконання програмної затримки використовуйте функцію Delay(), описану в stdlib.h бібліотеці, яка підключається директивою

```
#include <stdlib.h>
```

9.5. Індивідуальні завдання

Виконати п. 9.3. Варіанти мелодій наведені нижче. Після ноти вказано її тривалість за секунди.

1. До(1), ре(2), мі(3).
2. Сі(3), ля(2), соль(1).
3. До-діез(5), соль(1), ре(3).
4. Мі(2), до(3), ля-діез(1)
5. Фа(3), соль-діез(1), фа(3).
6. Фа-діез(2), ре(4), сі(2).
7. Соль(1), ля-діез(1), ля(2).
8. Ля(3), до-діез(1), ля-діез(3).
9. Соль-діез(2), сі(3), мі(1).
10. Ля-діез(1,5), фа(1), до(2).
11. Сі(0,5), до-діез(3), ля-діез(0,5).
12. До(2,5), фа(1), соль(2).
13. Фа(2), фа-діез(2), соль(2).
14. Ре-діез(2,5), ре(2,5), до-діез(2,5).
15. Мі(1), ре(2), до(1).
16. Ре(1,5), ля-діез(2), ре-діез(1).
17. Соль(2), ля(1), до-діез(0,5).
18. Ля(4), ре-діез(1), сі(0,5).
19. Ре(3), мі(0,5), фа(0,9).
20. Сі(0,5), до(1,3), ля(1,5).

9.6. Зміст звіту

9.6.1. Тема лабораторної роботи
9.6.2. Мета роботи.
9.6.3. Індивідуальне завдання.
9.6.4. Текст програми.
9.6.5. Результати роботи програми.
9.6.6. Висновки.

Лабораторна робота 10

Тема: Робота відеосистеми ПЕОМ у текстовому режимі

Мета роботи: Вивчення особливостей функціонування відеосистеми у текстовому режимі та набуття практичних навичок роботи з відеомонітором у цьому режимі.

10.1. Теми для попереднього опрацювання

10.1.1. Організація відеопам'яті у текстовому режимі.
10.1.2. Структура відеоадаптера.

10.2. Опис роботи

Всі відеосистеми використовують буфер, що є двопортовою пам'ятью на платі відеоконтролера (адаптера), в який мікропроцесор записує дані для зображення на екрані. Екран періодично оновлюється скануванням цих даних з боку дисплея. Розмір та розташування буфера в адресному просторі залежить від типу відеоконтролера та режиму екрану. Коли в буфері зберігається кілька зображень екрана, кожне окреме зображення екрана називають сторінкою дисплея.

10.2.1. Монохромний адаптер.

Має 4К байт пам'яті на платі, починаючи з адреси B0000h (тобто B000:0000h). Цієї пам'яті вистачає для зберігання однієї 80-символьної сторінки тексту.

10.2.2. Кольоровий графічний адаптер.

CGA має 16К байт пам'яті на платі, починаючи з адреси пам'яті B8000h. Цього достатньо для відображення одного графічного екрана або для відображення від чотирьох до восьми екранів тексту в залежності від числа символів у рядку – 40 або 80.

10.2.3. Графічні адаптери (EGA, VGA та SVGA) .

Адаптери EGA VGA та SVGA працюють у текстовому режимі аналогічним чином. Пам'ять адаптерів, крім використання її як відеобуфера, зберігає також описи форми символів. Стартова адреса відеобуфера програмована, тому буфер з адресами A000h – BFFFh використовується для

покращених графічних режимів, а починаючи з адреси B000 h та B800 h – для режимів, сумісних відповідно до монохромного та кольорового текстових режимів.

Доступна кількість сторінок залежить як від режиму екрана, так і кількості наявної пам'яті. Внаслідок своєї складності EGA, VGA та SVGA мають ПЗП на 16К байт, який замінює та розширює процедури BIOS роботи з терміналом. Початок області ПЗП – адреса C000:0000 h .

10.2.4. Вміст буфера в текстовому режимі.

У текстових режимах встановлюється така відповідність між пам'яттю відеоконтролера та зображенням на екрані. На початку пам'яті записуються дані про символ, що знаходиться на першому рядку в лівому куті, потім дані про інші символи першого рядка, потім дані про символи другого рядка починаючи зліва і т.д.

Під час виведення тексту різні відеосистеми працюють однаково. Для екрану приділяється 4000 байт, так що на кожну з 2000 позицій екрана (25 рядків ×80 символів) припадає 2 байти. Перший байт містить код символу ASCII. Апаратура дисплея перетворює номер коду ASCII у пов'язаний із ним символ і надсилає його зображення на екран. Другий байт (байт атрибутів) містить інформацію про те, як має бути виведено цей символ. Для монохромного дисплея він встановлює, чи буде цей символ підкреслено, виділено яскравістю чи негативом, або застосовується комбінація цих атрибутів. У кольорових системах байт атрибутів встановлює основний та фоновий кольори символу, а також ознаку його миготіння.

При записі даних у буфер монітора значно підвищується швидкість виведення на екран.

10.2.5. Відеоатрибути символів.

Коли дисплей встановлений у текстовому режимі в будь-якій відеосистемі, кожній позиції символу на екрані відводиться два байти пам'яті. Перший байт містить код ASCII символу, а другий відеоатрибут символу. Кольоровий адаптер може виводити у кольорі як символ, так і всю область, відведену даному символу (фоновий колір).

Монохромний адаптер обмежений лише чорним та білим кольором, але він може генерувати підкреслені символи, що не може робити кольоровий адаптер.

EGA може робити все, що й інші системи, а також багато іншого. Зокрема, на покращеному дисплеї він виводить підкреслені кольорові символи, оскільки матриця зображення 8 ×14 символів надає таку можливість.

На рис. 10.1 наведено формат байта відеоатрибуту

7	6	5	4	3	2	1	0
M	**R**	**G**	**B**	**I**	**R**	**G**	**B**
*	**Колір фону**			**Колір символу**			

Рис. 10.1 - Байт відеоатрибуту

R - входження червоного кольору: 0 = не входить; 1=входить

G – входження зеленого кольору: 0=не входить; 1=входить

B – входження синього кольору: 0=не входить; 1=входить

I - інтенсивність: 0 = символ неяскравий; 1 = символ яскравий;

M - мерехтіння: 0 = символ не мерехтить; 1=символ мерехтить.

* – можливий режим, коли цей біт визначатиме не мерехтіння символу, а інтенсивність кольору фону.

Як колір символу, так і колір фону визначаються як сума трьох основних кольорів – червоного, зеленого і синього. Усі можливі кольори наведено у табл. 10.1 .

Таблиця 10.1 – Кольори символів та фону

	IRGB	Колір	
Колір символу	**0 0 0 0**	**чорний**	**Колір фону**
	0 0 0 1	**синій**	
	0 0 1 0	**зелений**	
	0 0 1 1	**блакитний (ціан)**	
	0 1 0 0	**червоний**	
	0 1 0 1	**рожевий (мажента)**	
	0 1 1 0	**коричневий**	
	0 1 1 1	**сірий**	
	1 0 0 0	**темно-сірий**	
	1 0 0 1	**яскраво-синій**	
	1 0 1 0	**світло зелений**	
	1 0 1 1	**світло блакитний**	
	1 1 0 0	**світло-червоний**	
	1 1 0 1	**світло рожевий**	
	1 1 1 0	**жовтий**	
	1 1 1 1	**білий**	

Монохромні символи використовують байт атрибутів дещо інакше. Як і з атрибутами кольору, біти 0-2 встановлюють колір символу, а біти 4-6 – фону. Ці кольори можуть бути лише білим та чорним.

Нормальний режим – білий на чорному, коли біти 0-2 встановлені в 111, а біти 4-6 встановлені в 000. Негативне зображення створюється зворотними значеннями бітів. Символи виводяться з підвищеною яскравістю, коли біт 3

встановлений в 1. У всіх випадках установка в 1 7-го біта дає мерехтіння символів.

10.2.5. Реєстри відеоконтролера.

У всіх відеосистемах контролер відеотерміналу CRTC виконаний на базі мікросхеми Motorola 6845 (EGA VGA та SVGA використовують рекомендовані мікросхеми, засновані на 6845).

Мікросхема 6845 має 25 управляючих регістрів, пронумерованих від 0 до 18h. Перші 10 регістрів фіксують горизонтальні та вертикальні параметри дисплея. Ці регістри автоматично встановлюються BIOS під час зміни режиму екрана. Не радимо експериментувати з цими регістрами, оскільки можна зіпсувати термінал. Регістри мають розмір 8 біт, але деякі пов'язані в парі, щоб зберігати 16-бітові величини. Пари регістрів Ah-Bh і Eh-Fh встановлюють форму та місцезнаходження курсору. Пара Ch–Dh керує початковою адресою виведення на дисплей.

Регістр 14h визначає, яка лінія сканування у рядку символу використовується для підкреслення.

Доступ до всіх 25 регістрів здійснюється через той самий порт, адреса якого для монохромного адаптера – 3B5 h, для кольорового адаптера та PCjr – 3D5 h. EGA VGA і SVGA використовують одну з цих двох адрес залежно від того, приєднаний до нього кольоровий або монохромний монітор. Для запису в регістр треба спочатку в регістр адреси, розташований у порту 3B4 h (3D4 h для кольорового), надіслати номер необхідного регістра. Тоді наступний байт, надісланий до порту з адресою 3B5 h (3D5 h), буде записаний у цей регістр.

CGA і MDA мають ще три регістри, які важливі для програмістів. Вони мають адреси 3B8 h, 3B9 h і 3BA h для монохромного і 3D8 h, 3D9 h і 3DA h - для кольорового адаптера. Перший встановлює режим екрану, другий пов'язаний в основному із встановленням кольорів екрану, а третій повідомляє корисну інформацію про статус дисплея.

EGA VGA і SVGA розподіляють ці функції між мікросхемою контролера атрибутів (адреса порту 3C0 h) і двома мікросхемами контролера графіки (адреси портів 3CC h -3CF h). Контролер атрибутів містить 16 регістрів палітри EGA, пронумерованих від 00 до 0F h . Ці регістри можуть містити 6-бітові коди кольорів, коли підключено покращений кольоровий дисплей, тому можуть бути використані будь-які 16 кольорів з набору 64.

10.2.6. Режими дисплея.

Можливі режими наведено у табл. 10.2.

Таблиця 10.2 – Режими графічного адаптера

Режим	Тип	Формат	Розмір символів	Кількість кольорів	Тип відеопам'яті
0	TXT	40x25	8x8*	16/8 (Shades)	4-х шарова
1	TXT	40x25	8x8*	16/8	4-х шарова
2	TXT	80x25	8x8*	16/8 (Shades)	4-х шарова
3	TXT	80x25	8x8*	16/8	4-х шарова
4	Gr	320x200	8x8	4	Лінійна
5	Gr	320x200	8x8	4 (Shades)	Лінійна
6	Gr	640 x 200	8x8	2	Лінійна
7	TXT	80x25	9 x 14*	3 (B/W/Bold)	Лінійна
08h- 0 h	резерв				
0Dh	Gr	320x200	8x8	16	4-х шарова
0 Eh	Gr	640x200	8x8	16	4-х шарова
0 Fh	Gr	640x350	8x14	3 (B/W/Bold)	Лінійна
10 h	Gr	640x350	8x14	4 або 16	4-х шарова
llh	Gr	640x480	8x16	2	Лінійна
12 h	Gr	640x480	8x16	16	4-х шарова
13 h	Gr	320x200	8x8	256	Лінійна
14 h -7 Fh	Режими нестандартних розширень BIOS для VGA та SVGA				

Функція 0 переривання 10 h встановлює режим дисплея. В AL повинен бути номер режиму згідно з табл. 10.2.

10.3. Порядок виконання роботи

10.3.1.2. Встановити текстовий (80×25) режим роботи дисплея.

10.3.1.2. Відповідно до індивідуального завдання вивести у задане місце екрана слова з необхідними відеоатрибутами.

10.4. Особливості програмування

10.4.1. Мовою Турбо-Паскаль.

10.4.1.1. При використанні програмного переривання необхідно:

- підключити модуль Dos, в якому описано процедуру Intr і тип змінної Registers;
- оголосити змінну цього типу, наприклад, reg:Registers;
- до регістрів мікропроцесора звертатися, як reg.al;
- процедуру переривання 10h викликати так:

Intr($10,reg).

10.4.1.2.Буфер відеоконтролера в текстовому режимі можна подати у вигляді масиву:

buff:array[0..3999] of byte absolute $B800:$0000.

Тоді для виведення на екран символу в i-й рядок і в j-й стовпець необхідно за адресою buff[i*160+j*2] записати ASCII код символу, а за наступною адресою – байт відеоатрибуту.

10.4.1.3. Якщо слова задані у вигляді рядка символів типу string, то для визначення ASCII коду k-ї літери рядка можна скористатися виразом ord(s[k]).

10.4.2. Мовою Турбо-С.

10.4.2.1. При використанні програмного переривання необхідно:

- підключити бібліотеку dos, в якій описано процедуру int86 та тип суміші REGS директивою

#include <dos.h>

- оголосити суміш REGS оператором

union REGS in,out

де in – ім'я структури вхідних регістрів
out – ім'я структури вихідних регістрів
до 8-розрядних регістрів адресуються як in.h.al або out.h.ah,
до 16-розрядних регістрів адресуються як in.x.ax або out.x.ax

- процедуру переривання 10h викликати так:

int86(0x10,&in,&out);

10.4.2.2. Звернення до буферу відеоконтролера в текстовому режимі здійснюється аналогічно зверненню до коміркам ОЗП та ПЗП за допомогою далеких покажчиків, оголошених як

char far*uk;

Тоді для занесення початкової адреси відеобуфера необхідно записати:
uk = (char far *) 0xB8000000;
а для виведення на екран символу в i-му рядку та в j-му стовпці записати

```
*(uk + i*160+j*2)=kod;
*(uk + i*160+j*2+1)=attr;
```

де kod і attr -змінні типу char, що описують ASCII код та атрибут символу відповідно.

10.5. Індивідуальні завдання

1. Перші два слова червоним на білому фоні; третє слово синім високої інтенсивності; четверте слово зелене миготить на червоному фоні.
2. Перші три слова синім кольором на червоному фоні; друге слово високої інтенсивності.
3. Друге слово магента на чорному фоні; п'яте слово мерегтить; сьоме слово біле низької інтенсивності.
4. Перші три слова синім на білому фоні; четверте слово червоним високої інтенсивності; шосте слово зелене миготить.
5. Сьоме слово біле на чорному фоні; п'яте слово червоним кольором високої інтенсивності; перші три слова синім кольором блимають.
6. Два слова кольору ціан на червоному фоні; третє слово блимає; перше слово коричневе високої інтенсивності.
7. Перші два слова сині високої інтенсивності; третє слово блимає; четверте слово зелене на червоному фоні.
8. Перше слово чорне на білому фоні; четверте слово червоне на зеленому фоні блимає; шосте слово коричневе високої інтенсивності.
9. Друге слово рожеве високої інтенсивності; третє слово ціан на зеленому фоні; п'яте слово блимає.
10. Перші два слова синім низької інтенсивності; третє слово блимає; четверте слово червоне на білому фоні.
11. Парні слова блакитним по рожевому; непарні – червоним по чорному; останнє п'яте слово блимає.
12. Перше слово чорне на білому фоні; четверте слово зелене на червоному фоні; решта – блакитним на чорному фоні.

Зазначені слова вивести на екран у рядок і стовпець, номери яких збігаються з номером студента в журналі.

10.6. Зміст звіту

10.6.1. Тема лабораторної роботи
10.6.2. Мета роботи.

10.6.3. Індивідуальне завдання.
10.6.4. Текст програми.
10.6.5. Результати роботи програми.
10.6.6. Висновки.

Лабораторна робота 11

Тема: Управління курсором, кольором бордюру, регістрами палітри, створення спеціальних символів

Мета роботи: Отримання практичних навичок керування курсором, кольором бордюру та регістрами палітри шляхом програмування регістрів адаптера, створення спеціальних символів.

11.1. Теми для попереднього опрацювання

11.1.1. Призначення регістрів відеоадаптера.
11.1.2. Опис символів у текстовому режимі.

11.2. Опис роботи

11.2.1. Управління курсором.

Курсор служить двом цілям. По-перше, він служить вказівником місця на екрані, в яке оператор програми буде щось виводити. По-друге, він забезпечує видиму точку відліку на екрані для користувача програми. Тільки для другого застосування курсор має бути видимим. Коли курсор невидимий, він все одно вказує на позицію екрана. Це важливо, оскільки будь-яке виведення на екран, яке підтримує операційна система, починається з поточної позиції курсору.

Курсор генерується мікросхемою контролера дисплея 6845. Ця мікросхема має регістри, що встановлюють розмір та положення курсору. Мікросхема 6845 забезпечує тільки мерехтливий курсор, хоча є програмні способи створення немерехтливого курсору. Частота мерехтіння курсору не може бути змінена. У графічних режимах курсор не виводиться, хоча символи позиціонуються на екрані тими самими процедурами встановлення курсора, що у текстових режимах.

Коли відеосистема працює в режимі, що допускає кілька дисплейних сторінок, кожна сторінка має свій власний курсор і при перемиканні між сторінками відновлюється позиція курсору, яку він займав, коли було останнє звернення до сторінки, що відновлюється. Деякі режими дисплея дозволяють мати до 8 дисплейних сторінок і відповідні їм позиції курсору зберігаються у наборі восьми 2-байтових змінних в області даних BIOS, починаючи з адреси 0040:0050h. У кожній змінній молодший байт містить номер стовпця, відраховуючи від 0, а старший байт містить номер рядка, також відраховуючи від 0. Коли число сторінок менше 8, використовуються змінні, розташовані в молодших адресах пам'яті.

11.2.1.1. Управління координатами курсору (позиціонування).

Для курсора можуть бути встановлені абсолютні координати або координати щодо поточної позиції. Абсолютні координати можуть змінюватися в межах 25 рядків та 80 (іноді 40) стовпців.

Регістри 0Eh та 0Fh мікросхеми 6845 зберігають положення курсору. Адресний регістр має порт 3D4 h (дані заносяться до порту 3D5 h). Старший байт зберігається у регістрі 0Eh, молодший – у регістрі 0Fh.

Ви можете змінити їх значення і курсор пересунеться у відповідну позицію екрана, але переривання виведення на екран DOS і BIOS ігноруватиме вашу установку і повернуть курсор у старе положення. Це відбувається тому, що кожного разу при виклику цих переривань вони відновлюють регістри курсору, використовуючи 2-байтове значення, що зберігається в області даних BIOS. У цій області, починаючи з адреси 0040:0050h, можуть бути до восьми таких значень, що дають поточне положення курсору для кожної зі сторінок дисплея. Перша позиція відповідає сторінці 0, друга сторінці 1 і т.д. Молодший байт кожної змінної містить номер стовбця, а старший номер рядка. Як стовпці, так і рядки нумеруються з нуля. Процедура низького рівня повинна модифікувати ці значення, щоб змінити положення курсора повністю.

Позиція курсору зберігається в регістрах 0Eh і 0Fh як число від 0 до 1999, що відповідає 2000 (25×80) позиціям екрану. Не сплутайте цю систему нумерації з позиціями відеобуфера від 0 до 3999, де кожен символ супроводжується ще байтом атрибутів (для отримання еквівалентного покажчика на позицію курсору треба зрушити покажчик відеобуфера на 1 біт праворуч). Звертаємо також вашу увагу на те, що не треба міняти місцями старший та молодший байти: у регістрі 0Eh – старший, а 0Fh – молодший.

Курсор функціонує незалежно від відеопам'яті. Це означає, що при прямій адресації в пам'ять дисплея програмне забезпечення має координувати переміщення курсору із вставкою нового символу у буфер.

Програми іноді читають та зберігають поточне положення курсору, щоб можна було тимчасово перевести курсор у командний рядок, а потім повернути його у вихідну позицію. Процедура читання позиції курсора (з регістрів та області даних BIOS) аналогічна до процедури встановлення курсора.

11.2.1.2. Управління формою курсору.

Курсор може змінюватися по товщині від тонкої лінії до максимального розміру, яке відводиться під символ. Він будується з коротких горизонтальних відрізків, верхній з яких називається початковим рядком курсору, а нижній – кінцевим рядком. Для монохромного дисплея під кожен символ відводиться 14 рядків, пронумерованих від 0 до 13 починаючи згори. Проміжки між

символами забезпечуються двома верхніми рядками та трьома нижніми. Більшість символів розташовується в рядках 2-10, хоча хвостики деяких символів досягають ліній 12 і 13, тоді як підкреслення займає один дванадцятий рядок. На 200-рядковому кольоровому дисплеї для кожного символу відводиться лише 8 рядків, а символ малюється у верхніх семи рядках. Ці 8 рядків пронумеровані від 0 до 7 починаючи згори, і нормальний курсор формується одним рядком 7. (Зазначимо, що на кольоровому дисплеї немає підкреслення, оскільки використання для цього рядка 7 призвело б до того, що символи зливалися б з розташованими під ними.) Кольоровий дисплей високої роздільної здатності використовує 14-рядковий монохромний варіант, коли він працює в режимі роздільної здатності, і 8-рядковий варіант при роботі в одному з кольорових графічних режимів.

Курсор може бути сформований будь-якою комбінацією прилеглих відрізків. Для монохромного дисплея він займає все відведене під символ місце, коли початковий рядок дорівнює 0, а кінцевий рядок дорівнює 13 (для графічного дисплея значення кінцевого рядка має дорівнювати 7). Якщо значення початкового та кінцевого рядків збігаються, виникає однорядковий курсор. Якщо номер кінцевого рядка менший за номер початкового, то курсор стає невидимим.

BIOS зберігає 2-байтову змінну за адресою 0040:0060h, яка містить поточні значення початкового та кінцевого рядків. Перший байт містить значення кінцевого рядка, а другий – початкового.

11.2.1.3. Створення альтернативних типів курсору.

Усі переривання операційної системи, пов'язані з виведенням на екран, використовують курсор. Ви можете змінити форму курсору або зробити курсор невидимим шляхом завдання адреси більше 2000 або завдання номера початкового рядка більше за номер кінцевого рядка. Можливі альтернативні типи курсору, коли виведення на екран здійснюється за допомогою методів прямого відображення пам'яті. При цьому "справжній" курсор вимикається, оскільки він не адресуватиме символи у певну позицію відеобуфера. Натомість створюється "фальшивий" курсор за допомогою байта атрибутів.

Найбільш ефективним методом є встановлення атрибута виведення в негативі для символа, на який вказує курсор. Для чорно-білого екрана в якості цього атрибуту слід застосовувати код ASCII 112. Інший спосіб - змусити символ, на який вказує курсор, блимати. У цьому випадку потрібно просто додати 128 до поточного значення атрибута, щоб символ почав блимати, і відняти 128, щоб припинити миготіння. Третій спосіб – встановити для символу режим підкреслення (код ASCII 1). І нарешті, у програмах, що використовують командний рядок, можна розглянути застосування спеціального графічного символу, який слідує за останнім символом

командного рядка, такого, як стрілки, що виводяться кодами ASCII 17 або 27. Зазначимо, що коли програма отримує введення в кількох режимах, ви можете допомогти ідентифікувати поточний режим за рахунок спеціального курсору.

11.2.2. Керування кольором межі екрану (бордюру).

Межа символьного екрана (за межами адресної області) може мати колір, відмінний від фонового кольору центральної частини екрана. Може бути використаний будь-який із 16 кольорів для CGA або будь-який з 64 кольорів для EGA. Для кольорового графічного адаптера CGA біти 0-3 порту 3D9h (реєстр вибору кольору) встановлюють колір межи, коли екран знаходиться в текстовому режимі.

Для адаптера EGA колір бордюру визначається регістром 11h блоку атрибутів адаптера. При підключенні покращеного графічного дисплея можливе завдання будь-якого з 64 кольорів. Для завдання кольору бордюру в EGA необхідно виконати такі дії:

- прочитати порт 3DAh для ініціалізації адресного регістру;
- встановити номер регістра 11h шляхом його запису в порт 3C0h;
- задати колір засилкою байта даних у порт 3C0h (аналізуються 6 біт);
- дозволити виведення на екран шляхом встановлення 5-го біта адресного регістру блоку в 1 (засилання коду 20h порт 3C0h). Зміна кольору бордюру використовується в драйверах клавіатури для індикації перемикання між російською та латинською абеткою.

11.2.3. Створення спеціальних символів.

Тільки монохромний адаптер неспроможний виводити символи виду, заданого самим програмістом. Кольоровий адаптер підтримує 128 символів, визначених користувачем, PCjr – 256, а EGA – 1024, з яких одночасно доступно 512. Для кольорового адаптера ROM-BIOS містить дані для виведення лише перших 128 символів набору ASCII (з номерами від 0 до 127). Останні 128 символів недоступні для вас, доки ви не створите їх, використовуючи описану тут техніку.

11.2.3.1. Адаптер CGA.

Символи для графічного адаптера та PCjr описуються за допомогою матриці 8*8 пікселів. Дані кожного символу містяться у восьми байтах. Кожен байт містить значення для точок одного ряду, починаючи з верхнього, причому старший біт (номер 7) відповідає лівій точці в ряду. Коли відповідний біт дорівнює 1 точка висвічується. Для опису символу необхідно визначити правильні послідовності бітів для восьми байтів і помістити їх у послідовні

комірки пам'яті.

Всі 128 символів разом вимагають 1024 байти, хоча зовсім не обов'язково, щоб були описані всі символи. Спеціальний вектор переривання (постійний покажчик у молодших адресах пам'яті) показує адресу першого байта першого символу розширеного набору, тобто, символ номер 128. Коли в позицію символу у відеобуфері посилається код 128, переглядаються і виводяться перші вісім байт. Якщо номер символу 129, то виводяться байти з дев'ятого до шістнадцятого тощо. Номер цього вектора переривання 1Fh і розташований він за адресою 0000:007Ch. Помістіть значення зсуву до молодшого слова (спочатку молодший байт), а адресу сегмента – до старшого слова (спочатку молодший байт) і змініть 8 байт опису будь-якого символу. Зазначимо, що можна визначати символи з великими номерами кодів без виділення пам'яті для символів з меншими номерами; треба просто, щоб вектор вказував на деяку адресу, яка менша, ніж адреса початку блоку, що містить дані для опису символів.

11.2.3.2. Адаптери EGA, VGA, SVGA.

Для EGA картина трохи складніша, хоч і гнучкіша. При ініціалізації текстового режиму один із двох наборів символів (8×8 або 8×14) залежно від роздільної здатності дисплея копіюється з ПЗП EGA в бітову площину 2 відеобуфера. Ця частина буфера розбита на блоки, причому стандартний набір символів міститься в блоці 0. У EGA визначено чотири блоки для опису символів, VGA, SVGA – 8 блоків. Кожен блок містить опис 256 символів (ASCII коди 0-255). Незалежно від розміру матриці опису символу під кожен символ відводиться 32 байти, тому кожен блок символів має розмір 8 Кбайт (32*256). Коли є більше одного блоку символів, біт 3 байту атрибутів визначає, з якого блоку будуть братися дані для опису символу.

Який із блоків буде необхідний - залежить від установки бітів регістру вибору набору символів, адреса порту якого 3C5h.

Попередньо треба записати 3 у порт 3C4h, щоб вказати необхідний регістр, після чого в 3C5h записати байт даних. Для EGA біти 1,0 дають номер блоку символів, який береться, коли біт 3 дорівнює 0, а біти 3,2 дають номер блоку символів, коли біт 3 байту атрибутів дорівнює 1; для VGA SVGA відповідно біти 1,0,4 визначають номер блоку, коли біт 3 дорівнює 0, а біти 3,2,5 – коли біт 3 дорівнює 1. У разі рівності номерів блоків можливість використання двох наборів символів відсутня, але можна задати будь-який набір. У стандартних налаштуваннях BIOS всі ці біти дорівнюють 0, тому використовується тільки блок 0. Однак ніщо не може завадити вам помістити свої символи в будь-яку потрібну вам позицію в будь-якому блоці. Для цього необхідно виконати такі дії:

- записати 6 в порт 3CEh (адреса багатоцільового регістра графічного контролера);
- записати 1 в порт 3CFh (переведення відеосистеми в графічний режим);
- записати 2 до порту 3C4h (адреса регістра маски бітової площини);
- записати 4 (0100b) в порт 3C5h (дозволити доступ до 2-ї бітової площини);
- записати 4 у порт 3C4h (адреса регістру режиму використання відеопам'яті);
- записати 6 у порт 3C5h (скасування режиму парне/непарне, який зчіплював бітові площини 0–1 і 2–3 у текстовому режимі);

Після дозволу запису з урахуванням номера блоку (Blok), ASCII коду символу (Kod) та числа байт в описі символу змінюється *N* байт, починаючи з адреси

Adr = A000h : Blok * 2000 h + kod * 32 .

Після цього необхідно відновити регістри та перевести відеосистему в текстовий режим:

- записати 4 у порт 3C4h (адреса регістру режиму);
- записати 2 в порт 3C5h (установка режиму парне/непарне);
- записати 2 до порту 3C4h (адреса регістра маски бітової площини);
- записати 3 (0011b) в порт 3C5h (дозволити доступ до 0 і 1-ї бітових площин);
- записати 6 в порт 3CEh (адреса багатоцільового регістру);
- записати Eh в порт 3CFh (переведення в текстовий режим) .

Вміст N байт визначається потрібним зображенням символу.

На рис. 11.1 наведено рукописне зображення літери "И" для матриці (8 ×14).

Зверніть увагу на те, що зверху, знизу та праворуч (або зліва) від зображення символу потрібно залишити порожні рядки для розділення символів у рядку та між рядками. Ця буква описується наступними 14 байтами: 00, 00, 88, 88, 88, 88, 88, 88, 98, AA, 44, 00, 00, 00.

Якщо ви змінили стандартний набір символів (Blok = 0), можна в будь-який момент відновити його з ПЗУ.

Для кольорового адаптера та PCjr для зміни вектора переривання 1Fh застосовуйте функцію 25h переривання 21h. При виклику переривання DS:DX повинні вказувати перший байт блоку даних.

Для адаптера EGA характерна функція 11h переривання 10h, яка маніпулює набором символів. Ця функція може бути дуже складною, коли вона застосовується для створення спеціальних режимів екрана, але її основне призначення є досить простим. Існують чотири підфункції.

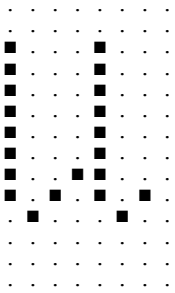


Рис. 11.1 – Зображення літери И

Коли AL дорівнює 0, дані, які визначає користувач, переносяться з пам'яті в блок опису символів 2-ї бітової площини.

Коли AL дорівнює 1 або 2, набори даних для символів 8×14 і 8×8 копіюються відповідно з ПЗП в блок опису символів.

Для VGA коли AL дорівнює 4, набір символів 8×16 копіюється з ПЗП у блок символів. Коли AL дорівнює 3, функція встановлює призначення блоку в регістрі вибору набору символів, як описано вище. В останньому випадку треба просто помістити відповідні дані у BL та викликати функцію.

Щоб завантажити дані з ПЗП, помістіть номер блоку в BL і виконайте функцію. Для завантаження своїх даних треба, щоб ES:BP вказували на них, число символів, що передаються, повинно бути в CX, зсув (номер символу) в блоці має бути в DX, число байтів на символ - в BH, а номер блоку - в BL. Після цього викликайте переривання 10h.

Ви можете отримати інформацію про статус наявного набору символів за допомогою підфункції 30h (реєстр AL) функції 11h (реєстр AH) переривання 10h. Ця підфункція не має вхідних регістрів, а при поверненні в CX міститься кількість байтів, відведених для кожного символу, а в DX – скільки рядів точок символ займає на екрані (обидва ці параметри залежать як від розміру символу, так і від вертикальної роздільної здатності екрана).

11.3. Порядок виконання.

11.3.1. Визначити ASCII коди символів, що становлять прізвище та ім'я студента.

11.3.2. Задати байтові описи вказаних символів при рукописному шрифті та опис особистого ієрогліфа студента.

11.3.3. Скласти програму, яка виконує такі дії.

11.3.3.1. Встановлює колір межі екрана відповідно до індивідуального

завдання.

11.3.3.2. Переміщує курсор у позицію, номер рядка якої відповідає місяцю народження студента, а номер стовпця відповідає номеру студента у журналі.

11.3.3.3. Змінює форму курсора шляхом зміни номера початкового та кінцевого рядка.

11.3.3.4. Визначає кількість байт на символ для опису символу.

11.3.3.5. Змінює байтові описи символів прізвища та імені студента.

11.3.3.6. Виводить у задану позицію курсора прізвище та ім'я студента рукописними літерами, завершуючи рядок особистими ієрогліфом.

11.3.3.7. Змінює колір фона, програмуючи 0-й регістр палітри.

11.4. Особливості програмування.

11.4.1. Мовою Турбо-Паскаль.

11.4.1.1. Для звернення до портів ПЕОМ використовуються попередньо визначені масиви Port і PortW. Наприклад, для запису адреси регістра бітової площини в порт 3C4h використовуємо вираз Port[$3C4]:=2.

11.4.1.2. При використанні програмного переривання необхідно:

- підключити модуль Dos, в якому описано процедуру Intr і тип змінної Registers;
- оголосити змінну цього типу, наприклад, reg:Registers;
- до регістрів мікропроцесора звертатися, як reg.al;
- процедуру переривання 10h викликати так:

Intr($10,reg).

11.4.1.3. Опис зображення символу зручно подати за допомогою типізованої константи. Наприклад, опис символу, зображеного на рис. 11.1, має такий вигляд:

```
const
ms:array[1..14] of byte=($00,$00,$88,$88,$88,$88,$88,
$88,$98,$AA,$44,$00,$00,$00);
```

11.4.1.4. Якщо слова задані у вигляді рядка символів типу string, то для визначення ASCII коду k-ї літери рядка можна скористатися виразом ord(s[k]).

11.4.1.5. Блок буфера відеоконтролера можна подати у вигляді масиву:

```
block:array[0.. 32 *256] of byte absolute $A000:$0000.
```

11.4.2. Мовою Турбо-С.

11.4.2.1. Для звернення до портів ПК застосовуються функції читання та запису у порт, які зберігаються у бібліотеці <dos.h>. Бібліотека підключається директивою

#include <dos.h>

після чого, наприклад, для запису значення 0Eh в порт 3D5h використовується вираз:

outportb(0x3D5,0x0E);

11.4.2.2. При використанні програмного переривання необхідно:

- підключити бібліотеку Dos, в якій написано процедуру int86x і тип суміші REGS директивою:

#include <dos.h>

- оголосити змінні суміші:

union REGS in, out, sr;

до регістрів мікропроцесора звертатися, як in.h.ah, in.x.ax;
до сегментних регістрів звертатися, як sr.es;
Для доступу до регістру BP необхідно оголосити структуру

struct REGPACK inb;

після чого до регістру звертатись, як inb.r_bp;
процедуру переривання 10h викликати так:

int86x(0x10,&in,&out,&sr).

11.4.2.3. Опис зображення символу зручно задавати під час ініціалізації масиву. Наприклад, опис символу, зображеного на рис. 11.1, має такий вигляд:

char ms[14]={0x00,0x00,0x88,0x88,0x88,0x88,0x88,
0x88,0x98,0xAA,0x44,0x00,0x00,0x00};

4.2.4. Звернення до буфера відеоконтролера аналогічне зверненню до комірок ОЗП за допомогою далеких покажчиків, оголошених

char far*uk;

Тоді для занесення початкової адреси відеобуфера необхідно записати:

uk = (char far *) 0xA0000000;

11.5. Індивідуальні завдання

За другою цифрою номера студента за журналом вибрати параметри зображення, що формується.

Таблиця 11.1 – Варіанти індивідуальних завдань

Параметри	Номер у журналі									
	0	1	2	3	4	5	6	7	8	9
n_str	0	1	2	3	4	5	6	7	8	9
k_str	13	12	11	10	8	9	10	11	12	13
col_fon	30	31	32	33	34	35	36	37	38	39
col_bor	10	9	8	7	6	5	4	3	2	1

де: n_str і k_str - початковий і кінцевий рядок позиції курсору в знайомому.

col_fon – колір фону (0 регістр палітри)

col_bor – колір межі екрана.

11.6. Зміст звіту

11.6.1. Тема лабораторної роботи

11.6.2. Мета роботи.

11.6.3. Індивідуальне завдання.

11.6.4. Текст програми.

11.6.5. Результати роботи програми.

11.6.6. Висновки.

Лабораторна робота 12

Тема: Робота зі сторінками

Ціль роботи: Отримання практичних навичок маніпуляції з екраном, використовуючи перемикання сторінок.

12.1. Теми для попереднього опрацювання

12.1.1. Організація відеопам'яті у текстовому режимі.
12.1.2. Призначення регістрів відеоадаптера.

12.2. Опис роботи

Зсув екрана та розбиття на сторінки – це два способи перенесення блоку інформації з пам'яті на екран. При зсуві одна з меж екрана зсувається всередину екрана, стираючи інформацію на протилежному боці. Потім звільнена область заповнюється з відеопам'яті. Повторення цієї дії рядок за рядком створює ілюзію зсуву екрана.

З іншого боку, розбиття на сторінки ґрунтується на одночасному зберіганні декількох екранів інформації у відеобуфері та перемиканні виведення з однієї сторінки на іншу. Використання дисплейних сторінок неможливе на монохромному адаптері, оскільки його пам'яті вистачає лише для одного символьного екрану. Інші відеосистеми у більшості екранних режимів можуть працювати з кількома сторінками. Використання сторінок дисплея особливо корисне при побудові складних картин "за лаштунками", після того, як ця робота завершена, новий екран виводиться моментально.

12.2.1. Ззув екрана шляхом зміни даних у відеобуфері.

12.2.1.1. Вертикальний зсув всього екрану – тривіальне завдання, оскільки права межа одного рядка в пам'яті продовжується лівою межею наступного рядка. Зсув всього вмісту відеобуфера на 160 байт вгору по пам'яті (80 символів у рядку ×2 байти на символ) призводить до зсуву екрана вниз на один рядок.

12.2.1.2. Горизонтальний зсув екрану.

Відеобуфер є одним довгим рядком. Тому якщо кожен символ буфера зрушити на 10 байт вниз, то сумарний ефект полягатиме в тому, що ліві 5 символів кожного рядка будуть пересунуті в останні 5 позицій попереднього рядка. Таким чином, весь екран буде зсунутий вліво на 5 позицій, пересуваючи 5 непотрібних стовпців у праву частину екрана. Все, що після цього

залишається, – це очистити праві 5 стовпців або запам'ятати їх "невидимими" за межами екрана символами, які зберігаються в ОЗП або вводяться користувачем.

12.2.3. Сторінкова організація відеопам'яті.

Оскільки всі відеосистеми, крім монохромного дисплея, мають достатньо пам'яті для кількох відеобуферів, одночасно можуть бути сконструйовані кілька екранів, кожен із яких може бути виведений у потрібний момент. Замість переміщення даних у відеопам'яті, монітор надсилає дані з іншої області відеопам'яті. Кількість доступних сторінок може змінюватись в залежності від відеосистеми та режиму дисплея.

У режимах 0–3 та 7 є 8 сторінок. BIOS зберігає у своїй області даних однобайтову змінну, яка вказує, яка зі сторінок виводиться зараз. Діапазон значень цієї змінної від 0 до 7. Вона розташована за адресою 0040:0062h.

Дисплейні сторінки вибираються за рахунок зміни точки відеопам'яті, з якої монітор приймає дані. Ця точка пам'яті встановлюється регістрами 0Ch (старший байт) та 0Dh (молодший байт) мікросхеми 6845, які називаються регістрами стартової адреси. Для програмування регістрів стартової адреси необхідно записати номер регістра в адресний регістр блоку (надіслати номер у порт 3D4h, після чого записати дані в порт 3D5. Значення адреси розділу сторінок для буфера, що починається з B800h, представлені в табл. 12.1.

Таблиця 12.1 – Початкові адреси у відеопам'яті.

№ сторінки	Початкова адреса сторінки у відеопам'яті	
	Для сторінок 40*25	Для сторінок 80*25
0	0000h	0000h
1	0400h	0800h
2	0800h	1000h
3	0C00h	1800h
4	1000h	2000h
5	1400h	2800h
6	1800h	3000h
7	1C00h	3800h

12.2.4. Апаратний зсув.

Оскільки сторінки тексту прилягають одна до одної у відеобуфері, невеликий текстовий масив може повністю поміщатися у пам'яті. В цьому випадку текст може зсуватися вгору і вниз екраном, не пересуваючись реально в буфері. Натомість екран починає показувати вміст буфера, починаючи з різних точок, і тим самим створюється ілюзія зсуву. Цей метод називається апаратним зсувом.

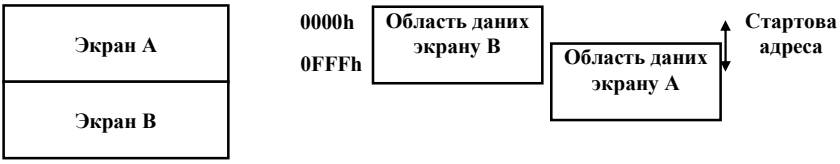
Апаратний зсув досягається за рахунок зміни стартової адреси виведення. Це число, що вказує на символ відеобуфера, який буде виводитися в лівому верхньому куті екрана. Додавання 80 до цього значення "зсуває" весь екран на один рядок вгору, а віднімання 80 - на один рядок вниз. У режимі з 40 символами в рядку замість 80 треба додавати або віднімати 40.

Зазначимо, що регістр стартової адреси не зважає на байти атрибутів, тому ви повинні обчислювати адреси пам'яті інакше, а не так, як при прямому відображенні в пам'ять. Майте також на увазі, що, незважаючи на наявність розривів пам'яті між межами сторінок (96 байт між 80-символьними сторінками та 48 байт між 40-символьними сторінками), мікросхема 6845 пропускає ці області і зсув безперервно відбувається з однієї сторінки на наступну. Апаратний зсув відбувається настільки швидко, що може виявитися необхідним вставити процедуру затримки, щоб користувач міг побачити, наскільки зсунувся екран.

BIOS зберігає поточне значення регістра стартової адреси змінної у своїй області даних. Ця двобайтова змінна розташована за адресою 0040:0004h.

12.2.5. Створення розділеного екрану

Адаптери EGA VGA SVGA дозволяють на апаратному рівні створювати так званий розділений екран в алфавітно-цифровому режимі. Верхню частину екрана називатимемо екраном А, а нижню – екраном В, як це показано на рисунку (рис. 12.1).



а) структура розділеного екрана б) відображення екранів на відеопам'ять

Рис. 12.1 – Організація розділеного екрану

На рис. 12.1.б показано відображення екрану на відеопам'ять розміром 32 Кб. Зауважте, що в текстовому режимі адаптер має відеобуфер розміром 32 Кб. Інформація в екрані А знаходиться за адресою, визначеною значеннями регістрів старшої та молодшої складових початкової адреси (0Ch і 0Dh) блоку управління ЕПТ (електронно-променевої трубки). Дані екрана В завжди розміщені у відеобуфері за адресою 0000h.

Для роботи з розділеним екраном використовується регістр порівняння рядків (18h) блоку управління ЕПТ. Необхідно спочатку надіслати в порт 3D4h 18h (адреса регістру), а потім у порт 3D5h – номер рядка, який є межею двох

екранів. У цьому блоці міститься внутрішній лічильник виведених у поточному кадрі рядків растру, значення якого постійно порівнюється зі значенням регістру порівняння рядків. Як тільки їх значення стають рівними, генератор адреси пам'яті скидається в 0. Після цього генератором адреси формуються послідовні адреси, починаючи з нульової адреси до завершення кадру.

Екран В може плавно переміщатися вгору/вниз екраном монітора для чого необхідно під час зворотного вертикального ходу променя проводити відповідні зміни значення у регістрі порівняння рядків. У екрані В неможливо проводити апаратний скролінг, т.к. для його організації використовуються регістри старшої та молодшої складових початкової адреси.

12.3. Порядок виконання.

12.3.1. Написати програму, яка виконує такі дії:

12.3.1.1. Очікує в циклі натискання алфавітно-цифрової клавіші, і при натисканні клавіші виводить набраний користувачем текст шляхом прямого відображення у відеобуфер у вигляді "рядка, що біжить". При цьому символи спочатку заповнюють рядок зліва направо до 80 позицій, а при подальшому наборі тексту на 1 символ, останній введений символ записується на 80 позицію, перший символ рядка зникає. Для організації " рядка, що біжить" використовувати апаратний зсув екрана. Номер рядка вибирається із індивідуального завдання.

12.3.1.2. Завантажує текстовий файл об'ємом 32 Кбайта у відеобуфер, починаючи з адреси B800:0000h та при натисканні клавіш 0-7 виводить на екран поточну відеосторінку.

12.3.1.3. При натисканні клавіші СТРІЛКА ВГОРУ і СТРІЛКА ВНИЗ виконує скролінг екрана на один рядок вгору або вниз відповідно.

12.3.1.4. Створює розділений екран шляхом програмування регістру порівняння рядків, і клавішами СТРЕЛКА ВГОРУ і СТРЕЛКА ВНИЗ виконує скролінг екрану А.

12.4. Особливості програмування.

12.4.1. Мовою Турбо-Паскаль.

12.4.1.1. Буфер відеоконтролера в текстовому режимі можна подати у вигляді масиву:

buff:array[0..3999] of byte absolute $B800:$0000.

Тоді для виведення на екран символу в i-й рядок і в j-й стовпець необхідно за адресою buff[i*160+j*2] записати ASCII код символу, а до наступного – байт відеоатрибута.

12.4.1.2. Для читання текстового файлу з перезаписом у відеобуфер необхідно:

- оголосити файлову змінну, наприклад,

```
var
        f:text;
```

- зв'язати файлову змінну з ім'ям текстового файлу на диску (зовнішнім ім'ям) за допомогою процедури Assign, наприклад, для файлу на диску D у каталозі CAT з ім'ям FIL.PAS треба зазначити:

```
Assign(f, 'D:CAT\FIL.PAS');
```

- відкрити файл читання процедурою Reset: Reset(f);
- читати посимвольно в циклі з текстового файлу в змінну, наприклад, ch (ch:char) за допомогою процедури Read із переписуванням коду прочитаного символу та його атрибуту в відеобуфер Buff (див. п. 4.1.1):

```
Read (f, ch);
buff[2*i]:=ord(ch);
buff[2*i+1]:=$07; {білий на чорному}
i – номер прочитаного символу (починаючи з 0);
```

- читати з текстового файлу доти, доки не зустрінеться символ "кінець файлу" (функція Eof(f) при цьому набуде значення true);
- після виявлення кінця файлу закрити файл за допомогою процедури Close: Close (f).

12.4.2. Мовою Турбо-С.

12.4.2.1. Звернення до буфера відеоконтролера в текстовому режимі здійснюється аналогічно зверненню до комірок ОЗП за допомогою далеких покажчиків, оголошених так

```
char far*uk;
```

Тоді для занесення початкової адреси відеобуфера необхідно записати: uk = (char far *) 0xB8000000;

а для виведення на екран символу в i-му рядку та в j-му стовпці записати:

```
* (uk + i * 160 + j * 2) = kod; * (uk + i * 160 + j * 2 +1) = attr;
```

де:kod та attr – змінні типу char, що описують ASCII код та атрибут символу відповідно.

12.4.2.2. Завантаження файлу у відеобуфер виконується так:

- відкриття файлу та встановлення покажчика на початок файлу за виразом

```
fp = fopen (fname, "rb");
```

де: fp - покажчик, який має бути описаний як FILE * fp;

fname-ім'я файлу, що завантажується,

"rb"-режим читання байтового файлу.

- читання в циклі байту із файлу за виразом

```
*uk = getc (fp);
```

де *uk - покажчик на відеобуфер, який має бути описаний як
char far *uk

- закриття файлу функцією

fclose(fp);

Для роботи із зазначеними функціями необхідно підключити бібліотеку stdio.h директивою

#include <stdio.h>.

12.5. Індивідуальні завдання

У табл. 12.2 за другою цифрою номера студента за журналом вибрати параметри "рядка, що біжить" і розділеного екрану.

Таблиця 12.2 – Варіанти індивідуальних завдань

Параметри	Номер у журналі									
	0	1	2	3	4	5	6	7	8	9
N _s	5	6	7	8	9	10	11	12	13	14
BEG_S	0	1	2	3	4	10	9	8	7	6
END _ S	60	65	70	75	80	75	70	65	60	50
N_CR	7	8	9	10	11	12	13	14	15	16

де: N_S – номер рядка дисплея, де організується "рядок, що біжить";
BEG_S, END_S – початкова та кінцева позиція "рядка, що біжить".
N_CR - номер рядка дисплея, який є межею розділеного екрана.

12.6. Зміст звіту

12.6.1. Тема лабораторної роботи
12.6.2. Мета роботи.
12.6.3. Індивідуальне завдання.
12.6.4. Текст програми.
12.6.5. Результати роботи програми.
12.6.6. Висновки.

Лабораторна робота 13

Тема: Робота у графічному режимі. Керування екраном, кольором. Зображення точки

Мета роботи: Набуття практичних навичок роботи з відеомонітором у графічному режимі.

13.1. Теми для попереднього опрацювання
13.1.1. Призначення та режими роботи адаптера у графічному режимі.
13.1.2. Організація відеобуфера у графічному режимі.

13.2. Опис роботи

Кольоровий графічний адаптер має три графічні режими, PCjr – шість, а EGA – сім, VGA, SVGA – безліч недокументованих режимів. Вимоги до розміру пам'яті суттєво відрізняються для доступних режимів залежно від роздільної здатності екрана та кількості кольорів. У своїх покращених графічних режимах EGA використовує пам'ять дисплея зовсім по-іншому, ніж решта відеосистем, але він точно емулює їх методи роботи з пам'яттю в трьох загальних режимах.

13.2.1. Кольоровий графічний адаптер CGA та система PCjr.

Спочатку розглянемо кольоровий адаптер та систему PCjr. Два кольори (чорний та білий) вимагають лише один біт пам'яті для кожної точки на екрані. Чотири кольори займають 2 біти, а 16 кольорів – 4 (8-кольорові режими не використовуються, оскільки три біти, які потрібні для їх задавання, не можна зручно розмістити в бітах байта). Для всіх режимів по вертикалі є 200 пікселів. Низька роздільна здатність (використовувана лише на PCjr) виводить 160 пікселів по горизонталі, середня роздільна здатність – вдвічі більша (320 пікселів), а висока роздільна здатність – ще вдвічі більша (640 пікселів).

У дво- та чотириколірному режимах PCjr має можливість вибрати будь-який з 16 доступних кольорів. Кольоровий адаптер більш обмежений. У двоколірному режимі він завжди обмежений білим і чорним, а в чотириколірному режимі тільки колір фону може вибиратися з 16 кольорів, тоді як основний колір потрібно брати лише з двох визначених палітр. Палітра 0 містить коричневий, зелений та червоний кольори, а палітра 1 – ціан, магента та білий.

На відміну від текстових даних у режимах 4-6 і 8-Ah, графічні дані розбиті на відеосторінці на частини. У більшості режимів дані розбиваються на дві частини, при цьому перша половина буфера містить дані для парних

рядків екрану, а друга – для непарних (рядки нумеруються починаючи з верху вниз).

13.2.2. Графічні адаптери EGA, VGA, SVGA .

Адаптер EGA забезпечує можливість роботи у різних відеорежимах разом із кольоровими чи монохромними моніторами з цифровими входами. Крім того, адаптер забезпечує можливість роботи зі світловим пером. Адаптер може функціонувати в декількох графічних режимах (використовуються 4 бітові площини) і має можливість завантаження у відеопам'ять шрифтів в алфавітно-цифрових режимах.

Адаптери VGA, SVGA мають аналогові виходи (рівень сигналу кожного аналогового виходу R, G, B задає інтенсивність відповідної складової кольору), їх архітектура є розвитком архітектури EGA, до якої додано блок цифро-аналогового перетворювача, тому VGA, SVGA емулюють всі режими EGA , а також мають власні графічні режими високої роздільної здатності та якості кольору, для реалізації яких потрібні значні обсяги відеопам'яті. Структурна схема відеоадаптерів представлена на рис. 13.1

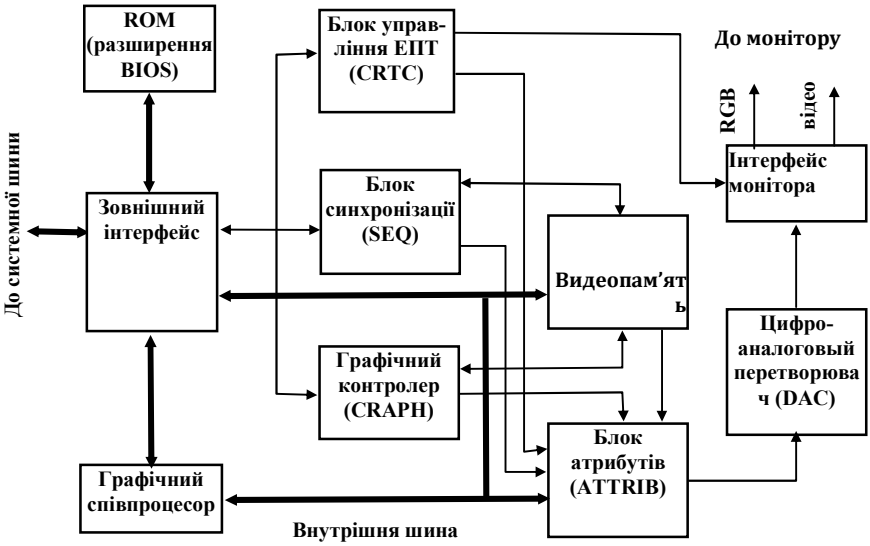


Рис. 13.1 - Архітектура відеоадаптера

13.2.2.1. Основні компоненти.

Блок керування електронно-променевою трубкою (CRT Controller) управляє сигналами горизонтальної та вертикальної синхронізації, початковою адресою виведення у відеобуфері, положенням та формою курсору та ін.

Блок синхронізації (Sequencer) генерує тактові сигнали та сигнали для синхронізації доступу до відеопам'яті. Даний пристрій забезпечує можливість доступу до відеопам'яті з боку процесора в спеціально виділені моменти часу в проміжку між інтервалами часу, необхідними для доступу до відеопам'яті в процесі регенерації зображення на екрані дисплея. У цьому блоці містяться регістри управління записом даних у бітові площині.

Графічний контролер (Graphics Controller) направляє дані з пам'яті в контролер атрибутів та процесор.

У графічних режимах дані відеопам'яті пересилаються в мікросхему контролера атрибутів послідовно. У текстових режимах дані пересилаються в паралельній формі в обхід графічного контролера. Для швидкої зміни зображення на екрані дисплея апаратурою забезпечується можливість запису 32 біт даних за цикл пам'яті (8 біт для кожної площини), а додаткова логіка дозволяє процесору записувати дані в дисплейну пам'ять не дотримуючись меж байтів.

Контролер атрибутів (Attribute controller) виконує перший рівень індексації кольору в текстових і графічних режимах, тобто, розширює 4-бітовий код кольору, що отримується з відеопам'яті (індекс кольору) до 6-бітового коду, який записаний в одному з 16 регістрів палітри. Цією ж мікросхемою виконуються дії з управління мерехтінням та підкресленням. Контролер отримує дані з відеобуфера і перетворює їх на керуючі сигнали, що подаються на вхід монітора.

Цифро-аналоговий перетворювач (DAC) з'явився в адаптері VGA, виконує другий рівень індексації кольору в текстових і графічних режимах, тобто, розширює 8-бітовий код кольору, що отримується з контролера атрибутів або безпосередньо з відеопам'яті (індекс кольору) до 18-ти розрядного коду (по 6 біт на кожну колірну складову - R, G, B), який записаний в одному з 256 регістрів PEL. Керуючі сигнали кольору подаються на вхід монітора в аналоговій формі лініями R, G і B.

Відеобуфер (Display Buffer – називається також відеопам'яттю або пам'яттю адаптера). Відеобуфер є двопортовою пам'яттю, тому доступний як з боку відеоадаптера (читання інформації та виведення на монітор), так і процесора в режимах запису і читання.

В адресному просторі процесора під відеобуфер виділено 2 сегменти пам'яті - A0000h і B0000h. Так як сучасні графічні режими високої роздільної здатності і якості передачі кольору вимагають істотно великих обсягів

відеопам'яті, то ця відеопам'ять відображається в область старших адрес. Адресація відеобуфера з боку процесора залежить від відеорежимів: у монохромних режимах – починається із сегментної адреси B0000h; у кольорових текстових – з адреси B8000h; у кольорових графічних – за адресою A0000h.

Базова система введення/виведення відеоадаптера (BIOSV) знаходиться в пам'яті спеціального ПЗП, встановленого на платі адаптера. BIOSV поєднується із системною базовою системою введення/виведення. Тут розміщуються шрифти, які використовуються для генерації символів та керуючі програми відеоадаптера. Початкова адреса ПЗП – C0000h.

13.2.2.2.Робота у графічному режимі 16/64 (10h)

Для графічного режиму 10h пам'ять організована у вигляді 4-х бітових площин, кожна з яких організована так само, як для чорно-білого режиму високої роздільної здатності: коли байт даних надсилається в певну адресу відеобуфера, кожен біт відповідає точці на екрані, причому вони описують горизонтальний сегмент рядка і біт 7 відповідає лівій точці. Записуються чотири такі бітові площини, відповідні одним і тим самим адресам, у відеобуфері. Це відводить кожному пікселю 4 біти, що дозволяє описувати 16 кольорів.

EGA може використовувати перший рівень індексації кольорів за допомогою регістрів палітри. При цьому стають доступними 64 кольори, кодування для яких rgbRGB. Складові r, g, b – слабкі інтенсивності 1/3 від max, RGB – нормальні інтенсивності 2/3 від max. Різні комбінації створюють 64 відтінки. Як завжди, 111111b відповідає білому кольору, а 000000b - чорному.

VGA, SVGA можуть використовувати другий ступінь індексації кольорів за допомогою регістрів PEL, що дозволяє задавати 218 відтінків.

13.2.2.3. Призначення регістрів.

13.2.2.3.1. Регістри блоку синхронізації

Призначення регістрів блоку синхронізації представлені у табл. 13.1.

Адресний – регістр, що вказує на один із регістрів блоку синхронізації, розташований за адресою 3C4h. У цей регістр завантажується двійковий номер регістру блоку синхронізації, в який буде здійснюватися запис. Номери регістрів, що записуються в адресний регістр, представлені в полі "Індекс" табл. 13.1, після чого за адресою 3C5h виконується запис/читання інформації у вибраний регістр.

Таблиця 13.1 – Регістри блоку синхронізації

№	Найменування	Порт	Індекс	Режим
1	Адреса (Addres)	3C4h	-	W
2	Ініціалізація (Reset)	3C5h	00h	W
3	Тактовий режим (Clocking mode)	3C5h	01h	W
4	Маски біт. площини (Map mask)	3C5h	02h	W
5	Вибір символьного блоку (Character map select)	3C5h	03h	W
6	Режим використання пам'яті (Memory mode)	3C5h	04h	W

13.2.2.3.2. Регістри блоку управління ЕПТ представлені у табл. 13.2.

Таблиця 13.2 - Призначення регістрів блоку управління ЕПТ

№	Найменування	Порт	Індекс	Режим в EGA
1	Адреса (Address)	3?4h	-	W
2	Загальна тривалість рядка (horizontal tota)	3?5h	00h	W
3	Довж. ділянки відображення в рядку (Horizontal display enable end)	3?5h	01h	W
4	Початку горизонтального гасіння (Start horizontal blank)	3?5h	02h	W
5	Закінчення горизонтального гасіння променя (End horizontal blan)	3?5h	03h	W
6	Початку горизонт. зворотнього ходу променя (Start horizontal retrac)	3?5h	04h	W
7	Закінч. горизонт. зворотнього ходу променя (End horizontal retrace)	3?5h	05h	W
8	Загальна кількість рядків растру в кадрі (vertical total)	3?5h	06h	W
9	Переповнення (Overflow)	3?5 h	07 h	W
10	Установка рядка растру (Preset row scan)	3?5h	08h	W
11	Вертикального розміру символу (Max scan line)	3?5h	09h	W
12	Початок курсору (Cursor start)	3?5h	0Ah	W
13	Закінчення курсору (Cursor end)	3?5h	0Bh	W
14	Старша складова початкової адреси (Start address high)	3?5h	0Ch	R/W
15	Молодша складова початкової адреси (Start address low)	3?5h	0Dh	R/W
16	Старша складова позиції курсору (Cursor location high)	3?5h	0Eh	R/W
17	Молодша складова позиції курсору (Cursor location low)	3?5h	0Fh	R/W
18	Початок верт. зворотнього ходу променя (Vertical retrace start)	3?5h	10h	W
19	Старша складова адреси світлового пера (Light pen high)	3?5h	10h	R
20	Закінчення зворотнього ходу променя (Vertical retrace end)	3?5h	11h	W
21	Молодша складова адреси світлового пера (Light pen low)	3?5h	11h	R
22	Тривалість ділянки відображення в кадрі (Vertical display end)	3?5h	12h	W
23	Зсув (Offset)	3?5 h	13 h	W
24	Положення символу підкреслення (Underline location)	3?5h	14h	W
25	Початок вертикального гасіння променя (Start vertical blank)	3?5h	15h	W
26	Закінчення вертикального гасіння променя (End vertical blank)	3?5h	16h	W
27	Управління режимом (Mode control)	3?5h	17h	W
28	Порівняння рядків (Line compare)	3?5h	18h	W
? = B у монохромних режимах та D у кольорових				

Експерименти з регістрами блоку управління ЕПТ, які формують зображення кадру, можуть призвести до поломки відеотерміналу.

13.2.2.3.3. Регістри графічного контролера представлені у табл. 13.3.

Таблиця 13.3 - Призначення регістрів графічного контролера

N	Найменування	Порт	Індекс
1	Регістр позиції графіки 1 (Graphics 1 Position)	3CC	-
2	Регістр позиції графіки 2 (Graphics 2 Position)	3CA	-
3	Адресний регістр графічного контролера (Graphics 1 & 2 Address)	3CE	-
4	Кольори (Set/Reset)	3CF	00
5	Дозволу кольору (Enable Set/Reset)	3CF	01
6	Порівняння кольору (Color Compare)	3CF	02
7	Обертання даних (Data rotate)	3 CF	03
8	Вибір площини для читання (Read Map Select)	3 CF	04
9	Вибір режиму (Mode)	3CF	05
10	Багатоцільовий (Miscellaneous)	3CF	06
11	Регістр незалежності від значення площини під час читання (Color Don't Care)	3CF	07
12	Битової маски (Bit Mask)	3CF	08

13.2.2.3.4. Регістри контролера атрибутів представлені у табл. 13.4.

Таблиця 13.4 - Призначення регістрів контролера атрибутів

№	Найменування	Порт	Індекс
1	Адресний регістр (Address Register)	3C0h	
2	Регістри палітри (Palette Registers)	3C0h	00 - 0Fh
3	Регістр керування режимом (Mode Control Register)	3C0h	10h
4	Регістр управління кольором бордюру (Overscan Color Register)	3C0h	11h
5	Регістр дозволу відображення бітової площини (Color Plane Enable Register)	3C0h	12h
6	Горизонтального зсуву пікселів (Horisontal Pel Panning Register)	3C0h	13h
7	Реєстр вибору кольору (тільки VGA)	3C0h	14h

13.2.2.4. Малювання точок.

Враховуючи організацію графічної інформації у відеобуфері виведення однієї точки передбачає зміну окремих біт пам'яті. Режими двох, чотирьох та шістнадцяти кольорів вимагають, щоб для установки характеристик однієї

точки були змінені один, два та чотири біти відповідно. Для цих операцій необхідні величезні витрати процесорного часу, тому графічне програмне забезпечення, зазвичай, працює дуже повільно. Ретельне обмірковування часто дозволяє відразу встановити всі біти одного байта, а не звертатися до того самого байта 4 або 8 разів.

Функція Ch переривання 10h встановлює точку. DX містить рядок, а CX – стовпець. Вони відраховуються від 0. Код кольору міститься в AL. Зазначимо, що вміст AX буде зруйнований під час переривання.

У той час як колір палітри записується в молодші біти AL, старший біт також має значення. Якщо він дорівнює 1, то над кольором проводиться операція XOR з поточним кольором. Нагадаємо, що операція XOR встановлює біт тільки в тому випадку, якщо з двох порівнюваних бітів встановлено лише один. Якщо обидва порівнювані біти дорівнюють 1 або обидва дорівнюють 0, то результат буде 0.

На низькому рівні ми маємо можливість прямого доступу до відеобуфера (відображення у пам’ять). Спочатку ви повинні обчислити зсув точки: а) - всередині буфера і б) - всередині байта, що містить біти, що належать до цієї точки. Після цього бітові операції забезпечать відповідне встановлення.

У режимах Dh, Eh та 10h пам'ять розбита на 4 бітові площини. Кожна площина організована таким же чином, як для чорно-білого режиму високої роздільної здатності кольорового адаптера, коли байт даних посилається в певну адресу відеобуфера, кожен біт відповідає горизонтальному сегменту лінії, а біт 7 - лівій точці. Виводяться чотири такі бітові площини, що належать до тих самих адрес у відеобуфері. Це призводить до того, що кожна точка описується чотирма бітами (даючи 16 кольорів), причому кожен біт знаходиться в окремому байті окремої бітової площини.

Але як ви можете записати 4 різних байти даних, розташованих за тією ж самою адресою? Відповідь на це питання полягає в тому, що ви не надсилаєте послідовно чотири байти на цю адресу. Натомість один із трьох режимів запису дозволяє змінити всі 4 байти на підставі одного байта даних, отриманого від процесора. Вплив даних, надісланих процесором, залежить від установки декількох регістрів, що включають два регістри маски, які визначають, на які біти і в яких бітових площинах будуть змінюватися біти. Для розуміння цих регістрів ми повинні спочатку розібратися з чотирма регістрами засувки (latch register). Вони містять дані для чотирьох бітових площин у тій позиції, до якої було останнє звернення. (Зауважимо, що термін "бітова площина" використовується як для цілої області відеобуфера, так і для однобайтового буфера, що тимчасово зберігається в регістрі засувки). Коли процесор надсилає дані за певною адресою, ці дані можуть замінити або повністю змінити дані регістру засувки, а згодом саме дані з регістру засувки записуються відеобуфер. Як дані процесора впливають на регістр засувки,

залежить від режиму запису, а також встановлення деяких інших регістрів. Під час читання адреси з відеобуфера регістри засувки заповнюються чотирма байтами з чотирьох бітових площин за цією адресою. Регістрами засувки легко маніпулювати, виробляючи з вмістом різні логічні операції, що дозволяє влаштовувати різні графічні трюки.

Регістр маски бітів (дивися блок графічного контролера регістр 08h) та регістр маски (дивися блок синхронізації регістр 02h) карти діють на регістри засувки, захищаючи певні біти або бітові площини від зміни під дією даних, що надходять від процесора. Установка біта регістру маски бітів в 0 маскує цей біт у всіх чотирьох бітових площинах, роблячи відповідну точку недоступною для зміни. Однак оскільки обладнання працює в байтових термінах, реально "незмінювані" біти перезаписуються в чотири бітові площини. Дані для цих бітів, що маскуються, зберігаються в регістрах засувки, тому програма повинна бути впевнена, що поточний вміст регістрів засувки відноситься до правильної адреси. Тому перед записом у відеобуфер треба спочатку зчитувати з нього. Біти 0-3 регістра маски карти відповідають бітовим площинам 0-3. Старші 4 біти регістру не використовуються. Коли біти 0-3 дорівнюють 0, відповідні бітові площини не змінюються при операціях запису. Дані, що посилаються в відеобуфер, можуть бути модифіковані і логічно оброблені разом з даними, що зберігаються в регістрах-засувках за допомогою регістра обертання даних (регістр 03h блоку графічного контролера). Біти 0-2 регістра обертання визначають число розрядів, на яке проводиться циклічний зсув даних процесора під час запису даних у відеобуфер. Для запису даних без зсуву цей біт має бути встановлений у 0. При записуванні задають логічну функцію даних процесора з даними регістрів засувок.

Значення бітів:

00 – дані не змінюються

01 – логічне "AND"

10 – логічне "OR"

11 - "XOR"

Операція зсуву виконується раніше за логічну операцію.

Ця властивість використовується по-різному в різних режимах запису.

Три режими запису і два режими читання встановлюються регістром установки режиму (див. блок графічного контролера регістр 05h). Режим запису встановлюється в бітах 0 та 1 як число від 0 до 2. Біт 2 повинен дорівнювати0, так само які біти4-7.

Біт 3 встановлює один із двох режимів читання з відеобуфера.

Цей біт може бути 0 або 1. BIOS EGA встановлює режим запису 0, режим читання 0.

13.2.2.4.1. Режим запису 0.

У найпростішому випадку режим запису 0 копіює дані процесора у кожну з чотирьох бітових площин. Нехай, наприклад, на певну адресу відеобуфера надіслано 11111111b і дозволені всі біти і всі бітові площини (тобто ніщо не масковано описаними вище регістрами масок). Тоді кожен біт у всіх чотирьох площинах буде встановлений в 1 так що ланцюжок бітів для кожної з відповідних точок буде 1111b. Це означає, що вісім точок буде виведено у кольорі 15, який спочатку відповідає білому кольору, хоча регістри палітри дозволяють, щоб насправді був будь-який з допустимих кольорів.

Тепер розглянемо ту саму ситуацію, коли надсилається значення 00001000b. Ланцюжок бітів для точки 3 буде 1111b, а для інших 0000b, що відповідає чорному (спочатку). Тому в даному випадку тільки точка 3 з'явиться на екрані, решту 7 точок буде вимкнено.

Якщо ви надішлете код палітри бажаного кольору в регістр маски карти, то регістр маскуватиме певні бітові площини таким чином, що буде відтворено необхідний колір. Наведене вище обговорення стосувалося одночасного виведення восьми точок. Ну а як вивести меншу кількість точок? У цьому випадку, звичайно, необхідно зберегти існуючі дані для деяких точок, а щоб це було можливо, поточний вміст цієї адреси зберігається в регістрах засувки. Потім використовується регістр маски бітів для маскування тих точок, які повинні змінюватися. Якщо біт цього регістру скинуто в 0, то дані, які отримують від процесора для цього біта, ігноруються і замість них використовуються дані, що зберігаються в регістрах засувки. Заповнення регістрів-засувок змінюється при ненульовому значенні регістра дозволу кольору (регістр 01h блоку графічного адаптера), розряди 0-3 якого дозволяють заповнення байта вибраних бітових площин значеннями з відповідних розрядів регістра кольору (регістр 00h блоку графічного адаптера). У цьому випадку значення даних процесора (маски картки) ігноруються.

13.2.2.4.2. Режим запису 1.

Режим запису 1 призначений для спеціальних програм. У цьому режимі поточний вміст регістру записується за вказаною адресою. Нагадуємо, що регістри засувки заповнюються операцією читання. Цей режим дуже корисний для швидкого перенесення даних під час операцій зсуву екрана. Регістр маски бітів та регістр маски карти не впливають на цю операцію. Не має значення також, які дані надсилає процесор – вміст регістрів засувки записується в пам'ять без змін.

13.2.2.4.3. Режим запису 2.

Режим запису 2 є альтернативним способом встановлення окремих точок. Процесор посилає дані, у яких мають значення лише 4 молодших біта, що розглядаються як колір (індекс регістру палітри). Можна сказати, що цей ланцюжок бітів вставляється впоперек площин. Ланцюжок дублюється на всі вісім точок, що відносяться до цієї адреси, доки регістр маски бітів не оберігає певні точки від зміни. Регістр маски карти активний, як і в режимі запису 0. Звичайно, процесор повинен надіслати повний байт, але тільки молодші 4 біти суттєві.

13.2.2.4.4. Режим читання: 0.

Процесор здійснює читання даних (8 точок) з регістру засувки тієї площини, яка дозволена регістром вибору площини для читання (розряди 0-2 регістра блоку 04h графічного контролера). Для читання кольорів (кодів регістру палітри) всіх 4 бітових площин необхідно послідовно дозволити читання та прочитати всі площини за однією адресою.

13.2.2.4.5. Режим читання 1.

В результаті виконання операції читання в 1 будуть встановлені тільки ті біти в байті, для яких колір збігається з кольором, заданим у бітах 0-3 регістра порівняння кольору (реєстр 02h блоку графічного адаптера).

13.2.2.5. Програмування регістрів.

Кожна з мікросхем адаптера має адресний регістр і кілька регістрів даних. Адресний регістр використовується як покажчик на той чи інший регістр даних. Адресний регістр є регістром типу "тільки запис", в який процесором за допомогою команди OUT може бути розміщений індекс обраного регістру даних.

Регістри даних кожної мікросхеми адаптера доступні через відповідний порт введення/виведення. Доступ до різних регістрів даних здійснюється шляхом попереднього занесення в адресний регістр індексу необхідного регістру даних з подальшою видачею команди OUT з боку процесора для внесення в нього необхідного значення. Відмінність мають регістри контролера атрибутів, де адресний регістр та регістр даних мають один порт з адресою 3C0h. Всередині контролера адресація організована таким чином, що кожного разу після запису в порт 3C0h здійснюється перемикання: адресний регістр – регістр даних, що відповідає значенню адресного регістру і навпаки. Для ініціалізації процесу перемикання адресний регістр – регістр даних

(спочатку доступ до адресного регістру, а потім до регістра даних тощо) необхідно виконати операцію читання з порту з адресою 3BAh або 3DAh. Після виконання операції читання перший доступ до порту 3C0h буде зверненням до адресного регістру контролера атрибутів. Після завантаження адресного регістру, наступна команда виведення в порт 3C0h призведе до запису необхідного значення у відповідний регістр даних контролера атрибутів. Виконання цієї команди знову робить адресний регістр доступним для запису та процес може бути продовжений.

Після запису даних у регістри контролера атрибутів необхідно встановити 5 біт адресного регістру контролера на 1 (дозвіл виведення), тобто, надіслати порт 3C0h значення 20h.

Відеоадаптер підтримує стандартні графічні функції BIOS. Можна вивести точку за допомогою функції Ch переривання 10h. При вході DX повинен містити номер рядка, а CX – номер стовпця, і те й інше відраховується від 0. Код кольору міститься в AL. Вміст AX змінюється під час переривання.

13.3. Порядок виконання роботи

13.3.1. Написати програму виведення на екран у графічному режимі, яка виконує такі дії:

13.3.1.1. Переводить відео систему у графічний режим 10h за допомогою переривання 10.

13.3.1.2. Встановлює заданий колір фона шляхом програмування регістра 00h палітри.

13.3.1.3. Виводить 16 рядків точок за різних значень регістрів адаптера. Початкова адреса i-го рядка відеобуфера обчислюється в режимі 10h за виразом

adp=A000:0000+80*i

13.3.1.4. Повторює виведення наступних 16 рядків із тими самими параметрами як і попередні, але з урахуванням зсуву даних у регістрі обертання.

13.3.1.5. Повторює виведення наступних 16 рядків із тими самими параметрами, але з урахуванням логічної операції.

13.3.1.6. Решту екрана після отриманого зображення за допомогою режиму запису 1 заповнює байтом, номер рядка і стовпця в отриманому вище зображенні відповідає номеру студента в журналі.

13.3.1.7. В отриманому зображенні "загасити" колір першого рядка шляхом занесення коду фону у відповідний регістр палітри.

13.3.1.8. Виконати апаратний горизонтальний зсув екрана шляхом зміни початкової адреси.

13.3.1.9. Повернутися до текстового режиму.

13.3.2. Написати, відлагодити та зберегти для виконання робіт 14, 15 процедуру put_pix читання байта та виведення за одне звернення до пам'яті до 8 точок із заданими параметрами

put_pix(x,y,color,bit_m,log_op)

де: x, y – координати байта відеопам'яті

color – код кольору

bit_m-значення бітової маски

log_op - код логічної операції з прочитаним значенням, що задається в регістрі обертання.

13.4. Особливості програмування.

13. 4.1. Мовою Турбо-Паскаль.

13.4.1.1. Для звернення до відеопам'яті застосовується попередньо визначений масив Mem. Наприклад, для читання байта за адресою A000:0000h використовується вираз b:=Mem[$ A 000:$0000] (b змінна типу byte).

13.4.1.2. Для звернення до портів ПК застосовується попередньо визначений масив Port. Наприклад, для запису значення 3Dh до порту 3C0h використовується вираз Port[$3C0]:=$3D; для читання з порту 3DAh використовується вираз data: = Port [$3DA], де data - змінна типу byte.

13.4.2. Мовою Турбо-С.

13.4.2.1. Для звернення до відеопам'яті використовуються далекі покажчики, які оголошуються у програмі так:

char far*uk; - для роботи з байтами.

Для читання байта за адресою A000:FFF5h використовується вираз

uk = (char far *) 0xA0000000;

b=* uk; де b - змінна типу char;

13.4.2.2. Для доступу до портів ПК застосовуються функції читання та запису у порт, які зберігаються в бібліотеці <dos.h>. Бібліотека підключається командою

#include <dos.h>

після чого, наприклад, для запису значення 3Dh в порт 3C0h використовується вираз:

outportb(0x3C0,0x3d);

Для читання з порту 3DAh використовується вираз:

data = intportb (0x3DA), де data - змінна типу char;

13.5. Індивідуальне завдання.

13.5.1. По 2 цифрі номера журналу вибрати параметри з табл. 13.5

Таблиця 13.5 - Варіанти індивідуальних завдань

Параметри	Номер у журналі									
	0	1	2	3	4	5	6	7	8	9
Номер початкового рядка	100	140	200	80	120	150	60	180	210	40
Колір фону	1A	23	31	18	2B	35	13	2F	39	1D
Колір 1-го рядка	A	1	У	3	З	5	6	D	9	8
Збільшення кольору і-го рядка	1	2	4	5	8	7	9	6	3	A
Маска бачимих точок	61	63	67	82	86	8 E	72	76	7E	E2
Зсув даних	1	2	3	4	5	6	7	1	2	3
Логічна операція	AND	OR	XOR	AND	OR	XOR	AND	OR	XOR	AND

13.6. Зміст звіту

13.6.1. Тема лабораторної роботи
13.6.2. Мета роботи.
13.6.3. Індивідуальне завдання.
13.6.4. Текст програми.
13.6.5. Результати роботи програми.
13.6.6. Висновки.

Лабораторна робота 14

Тема: Зображення ліній, затушовування

Мета роботи: Набуття практичних навичок створення малюнків.

14.1. Теми для попереднього опрацювання
14.1.1. Зображення точки у графічному режимі.

14.2. Опис роботи.

Функції креслення ліній є основними підпрограмами графіки та використовуються для відображення ліній у заданому кольорі шляхом завдання початкових та кінцевих координат. У той час, як зображення вертикальних та горизонтальних ліній не становить особливих труднощів, більш важким завданням є створення функцій, які малюють лінії вздовж діагоналей. Наприклад, які точки становлять лінію, що викреслюється від точки з координатами 0,0 до точки з координатами 80,120?

Один із підходів до розробки функцій креслення ліній використовує відношення між зсувом по координатах X та Y. Щоб показати цей підхід у дії, проведемо лінію між точками з координатами 0,0 та 5,10. Зміщення по X дорівнює 5, а по Y – 10. Відношення дорівнює 1/2. Воно буде використовуватися для визначення коефіцієнта залежності, за якою повинні змінюватися координати X і Y при зображенні ліній. У цьому випадку це означає, що збільшення координати X становить половину величини зміни координати Y. Початківець програміст часто використовує цей метод при розробці функцій креслення ліній. Хоча підхід математично вірний і простий для розуміння, для його правильної роботи і для того, щоб уникнути серйозних помилок округлення, необхідно використовувати математичні операції з числами з плаваючою точкою. Це означає різке зниження швидкодії, якщо у систему не буде включений математичний співпроцесор. Тому цей метод використовується досить рідко.

Найбільш загальний метод зображення ліній включає використання алгоритму Брезенхейма. Хоча основою у ньому є також відношення між відстанями по координатах X і Y, у цьому разі не потрібно виконувати розподіл чи обчислення чисел з плаваючою точкою. Натомість, відношення між значеннями координат X і Y представляється непрямим чином через серії додавань і віднімань. Основною ідеєю алгоритму Брезенхейма є реєстрація середніх значень похибок між ідеальним положенням кожної точки і тією позицією на екрані дисплея, в якій вона дійсно відображається. Похибка між ідеальним і дійсним положенням точки виникає через обмежені можливості технічних засобів. Фактично не існує дисплеїв з нескінченно великою

роздільною здатністю, і, отже, дійсне положення кожної точки лінії потребує найкращої апроксимації. У кожній ітерації циклу креслення лінії викликаються дві змінні X_err та Y_err, які збільшуються в залежності від зміни величин координат X та Y відповідно.

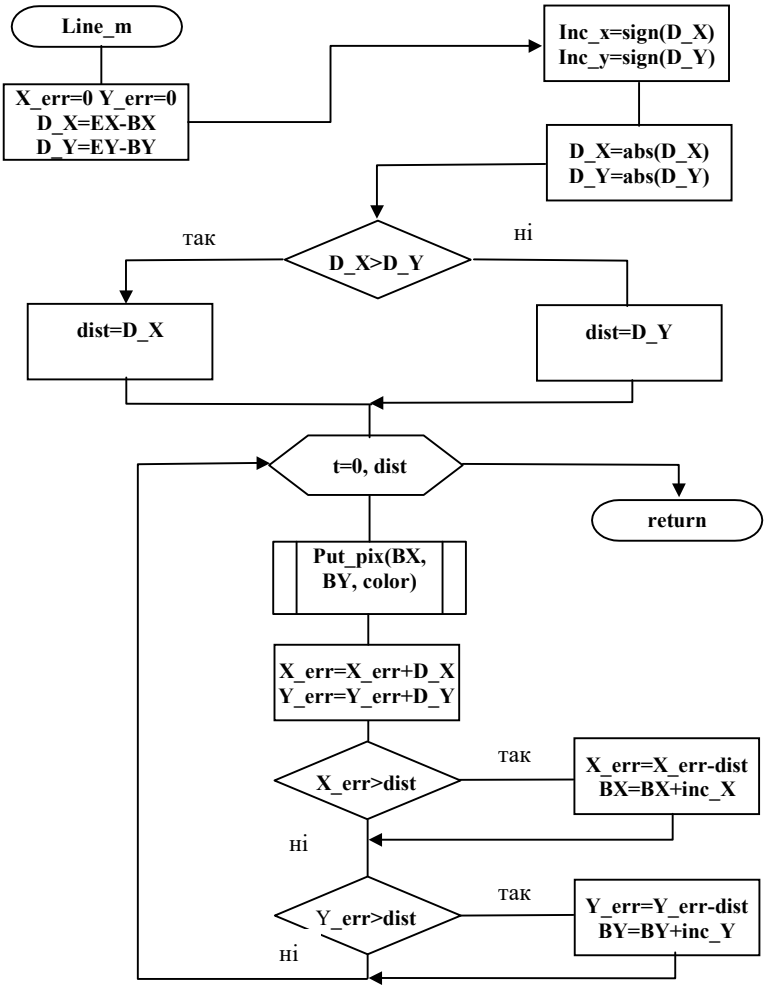


Рис. 14.1 – Алгоритм зображення ліній (Брезенхейма)

Коли значення похибки досягає певного значення, воно знову встановлюється у вихідне положення, а відповідний лічильник координат

збільшується. Цей процес триває доти, доки лінія не буде повністю накреслена. У цьому алгоритмі прийнято такі позначення: BX, BY – координати початкової точки; EX, EY – координати кінцевої точки; D_X, D_Y - максимальна відстань по відповідній координаті; c – колір точки; inc_X, inc_Y – напрямок руху за відповідною координатою.

Як видно з алгоритму, при русі по більшій координаті (dist) приріст по цій координаті зображення точки виконується завжди, у той час як по іншій координаті тільки при виконанні умови - err> = dist.

Якщо у вас є функції креслення ліній, то не складе особливих труднощів створити функції креслення прямокутників. Приклад, наведений тут, накреслює прямокутники у заданому кольорі шляхом створення координат двох протилежних кутів.

```
/*Креслення прямокутника */
void box(startx,starty,endx,endy,color_code)
int startx,starty,endx,endy,color_code;
{
        line(startx,starty,endx,starty,color_code);
        line(startx,starty,startx,endy,color_code);
        line(startx, endy, endx, endy, color_code);
        line(endx,starty,endx,endy,color_code);
}
```

Щоб зафарбувати прямокутник, потрібно виконати запис у кожну точку растру всередині прямокутника. При цьому необхідно в циклі малювати заданим кольором ряд горизонтальних ліній, що зміщені на одиницю по координаті y – функція bar_m().

Ретельне продумування дозволяє виключити зайву повільність, властиву багатьом програмам заповнення областей для графічного екрану. Коли заповнення засноване на простих обчисленнях, які діють по черзі для кожної точки, потрібно бітові операції, що витрачають багато часу. Більш економічний код може визначати, чи всі бітові позиції певного байта відеобуфера повинні мати той самий колір, і коли ця умова виконується, цьому байту присвоюється заздалегідь заготовлене значення, яке встановлює всі точки в правильний колір. При цьому немає необхідності повторювати операції над тим самим байтом, щоразу встановлюючи біти тільки для однієї з точок, інформацію про яку містить даний байт.

При затушовування прямокутника лініями тільки перший і останній байт кожного рядка може бути неповним (якщо BX або EX не є кратним 8). При затушовуванні горизонтальними лініями немає сенсу використовувати процедуру line_m(), що реалізує алгоритм Брезенхейма, необхідно просто визначати адресу байта у відеопам'яті, який відповідає в даний момент 8

точкам, що виводяться з початковими координатами X і Y, причому X кратен 8 : adr=A000: 0000h + X / 8 + Y * 80.

У роботі 11 пояснено, як створити опис символу у вигляді матриці 8×8 точок, що має необхідний вам вигляд. Хоча такі символи можуть виводитись лише у стандартні символьні позиції, їх використання може суттєво полегшити заповнення графіків. Зразок, що висвічує всі 8×8 пікселів можна вивести в інтервалі у кілька рядків і стовпців, заповнюючи область набагато швидше, ніж це досягається при крапковій замальовці.

При затушовування прямокутника в центральній частині бажано використовувати символи псевдографіки з кодами ASCII 219, 220, 221, 222, 223, які заповнюють все місце (219), нижню половину (220), ліву (221), праву (222) та верхню (223). При зображенні прямокутника теж немає сенсу застосовувати алгоритм Брезенхейма, оскільки він складається з горизонтальних та вертикальних відрізків. Зображення горизонтальних відрізків виконується аналогічно розглянутому вище способу затушовування горизонтальними лініями, а зображення вертикальних відрізків зводиться до виведення в циклі одного байта із заданою маскою (при поточній лінії 1 піксель маска містить одну 1) і збільшення адреси на 80 – перехід на наступний рядок. Модифікація зображення похилої лінії.

При використанні алгоритму Брезенхейма у разі D_X > D_Y у кожному рядку виводиться кілька точок. Якщо перед блоком 9 алгоритму аналізувати значення BX при незмінному BY і дописувати 1 в бітову маску, то при використанні розробленої в лабораторній роботі процедури виведення байта із заданою маскою звернення до неї виконується тільки у разі кратності 8 значення BX (перехід на адресу відеопам'яті) або зміни BY (перехід на наступний рядок). Якщо D_Y > D_X і товщина лінії становить 2, 3, і т.д. пікселя за одне звернення до put_pix() можна виводити відповідне число точок (якщо вони відносяться до однієї адреси відеопам'яті).

14.3.Порядок виконання роботи.

14.3.1. Написати програму, яка використовує розроблену в роботі 13 процедуру put_pix() і виконує такі дії:

14.3.1.1. Переводить систему у графічний режим 10h.

14.3.1.2. Накреслює прямокутник з параметрами, що відповідають індивідуальному завданню (rest_m()).

14.3.1.3. Зафарбовує прямокутну область з параметрами, що відповідають індивідуальному завданню (bar_m()).

14.3.1.4. Малює похилу лінію, використовуючи модифікацію алгоритму Брезенхейма з параметрами, що відповідають індивідуальному завданню (line_m()).

14.3.1.5. Повертає систему до текстового режиму.

14. 4. Особливості програмування.

14.4.1. Мовою Паскаль.

14.4.1.1. Для звернення до відеопам'яті застосовується попередньо визначений масив Mem. Наприклад, для читання байта за адресою A000:0000h використовується вираз b:=Mem[$ A 000:$0000] (b змінна типу byte).

14.4.1.2. Для звернення до портів ПК використовуються попередньо визначений масив Port. Наприклад, для запису значення 3Dh до порту 3C0h використовується вираз Port[$3C0]:=$3D; для читання з порту 3DAh використовується вираз data: = Port [$3DA], де data - змінна типу byte.

14.4.2. Мовою С.

14.4.2.1. Для звернення до відеопам'яті використовуються далекі покажчики, які оголошуються у програмі так:

char far*uk; - Для роботи з байтами.

Для читання байта за адресою A000:0000h використовується вираз:

uk = (char far *) 0xA0000000;
b=* uk; де b - Змінна типу char;

14.4.2.2. Для доступу до портів ПК застосовуються функції читання та запису у порт, які зберігаються в бібліотеці <dos.h>. Бібліотека підключається командою

#include <dos.h>

після чого, наприклад, для запису значення 3Dh в порт 3C0h використовується вираз:

outportb(0x3 C 0,0x3 D);

Для читання з порту 3DAh використовується вираз:

data = intportb (0x3DA), де data - змінна типу char;

14. 5. Індивідуальні завдання.

За другою цифрою номера студента вибрати параметри згідно з табл. 14.1.

Таблиця 14.1 - Варіанти та індивідуальних завдань

Параметри	Номер у журналі									
	0	1	2	3	4	5	6	7	8	9
Xn1	30	0	1 0	3 0	5 0	6 0	4 0	3 0	2 0	0
Yn1	60	0	20	50	10	40	30	10	40	50
Xk1	260	200	250	280	300	310	210	240	290	190
Yk1	120	150	140	130	120	110	100	150	140	130
L1	5	3	4	2	4	3	5	4	3	4
color	C	1	3	4	5	6	7	8	A	B
Xn2	350	400	410	450	430	460	360	330	420	340
Yn2	0	60	50	40	10	30	40	10	50	20
Xk2	580	500	600	550	630	560	680	510	520	530
Yk2	150	120	130	140	150	100	110	120	130	140
Color2	7	D	E	F	1	2	3	4	5	6
Xn2	10	30	0	20	30	40	60	50	100	90
Yn2	320	160	360	140	310	120	290	180	280	150
Xk2	630	500	400	600	580	530	420	510	620	460
Yk2	150	300	160	310	140	290	120	280	180	320
Color3	1	8	9	A	B	C	D	E	F	2
L3	3	5	4	3	4	5	3	4	2	4

де: Xn1, Yn1, Xk1, Yk1 – координати верхнього лівого (індекс n), та правого (індекс k) кута прямокутника; L1 – товщина лінії у прямокутнику; color1 – колір; Xn2, Yn2, Xk2, Yk2, color2 – відповідні параметри затушовування. Xn3, Yn3, Xk3, Yk3, L3, color3 – координати, товщина та колір похилої лінії.

14.6. Зміст звіту

14.6.1. Тема лабораторної роботи
14.6.2. Мета роботи.
14.6.3. Індивідуальне завдання.
14.6.4. Текст програми.
14.6.5. Результати роботи програми.
14.6.6. Висновки.

Лабораторна робота 15

Тема: Збереження та завантаження графічних зображень. Переміщення та обертання об'єкта.

Мета роботи: Набуття практичних навичок у маніпуляції графічними об'єктами.

15.1. Теми для попереднього опрацювання

15.1.1. Зображення ліній у графічному режимі

15.2. Опис роботи

15.2.1. Збереження та завантаження графічних зображень.

Будь-яка мова програмування має доступ до двох режимів читання відеопам'яті. У режимі 0 повертається байт, що міститься у всіх чотирьох площинах, за вказаною адресою. Режим 1 шукає вказаний код кольору і повертає байт, в якому біт встановлений 1, коли відповідна точка має даний колір. Біт 3 регістра режиму визначає, який режим читання встановлено (0 = режим 0). Доступ до цього регістру здійснюється через порт 3CFh і ви повинні попередньо послати 5 в порт 3CEh, щоб вибрати цей регістр. Зазвичай решта бітів цього регістру, в який можна тільки писати, скинуті в 0, крім бітів 0 і 1, які визначають режим запису. Оскільки при ініціалізації BIOS встановлює ці біти в режим запису 0 (так що вони обидва рівні 0), вам потрібно просто надіслати в цей регістр 0, щоб встановити режим читання 0, і послати 8, щоб встановити режим читання 1.

Режим читання 0 вимагає, щоб ви попередньо встановили регістр вибору картки. Єдине завдання цього регістру – встановити, яка з бітових площин має бути прочитана. Тому в нього треба надіслати число від 0 до 3. Цей регістр має адресу порту 3CFh і треба попередньо надіслати 4 в порт 3CEh, щоб вказати цей регістр.

Режим читання 1 складніший. Спочатку регістр порівняння кольорів повинен бути заповнений ланцюжком бітів для коду кольору, який ви шукаєте. Цей код зберігається в молодші 4 біти регістра; старші чотири біти не суттєві. Цей регістр має адресу порту 3CFh і вказується попереднім засиланням 2 в порт 3CEh. Після читання комірки пам'яті повертається байт, який має біти, встановлені в 1 для кожної точки, що має потрібний колір. Однак за рахунок використання регістру байдужості кольору (color don't care register) один і більше біт коду кольору можуть при порівнянні ігноруватися. Зазвичай 4 молодших біти цього регістру встановлені в 0; логічна 1 в одному з цих бітів призведе до того, що вміст відповідної бітової площини ігноруватиметься.

Регістр байдужості кольорів має адресу порту 3CFh і індексується

засилкою 7 в порт 3CEh. Старші 4 його біти не грають жодної ролі.

Жоден із цих двох режимів читання не може дати швидкої відповіді.

У режимі читання 0 необхідні 4 окремі читання, по одному для кожної бітової площини. У режимі читання 1 може знадобитися до 16 читань, перш ніж для потрібної точки буде повернуто встановлений біт, що вказує, що ця точка має даний колір.

Функція Dh переривання 10h повертає код кольору вказаної точки. Потрібно помістити номер рядка (що відраховується від 0) у DX, а номер стовпця (також відраховується від 0) – у CX. Результат повертається до AL.

15.2.2. Обертання точки в площині екрану.

Обертання точки в площині екрану (двовимірному просторі) є досить простим завданням у декартовій системі координат. Ви можете згадати з курсу аналітичної геометрії, що обертання точки навколо центру, що знаходиться на початку координат на кут theta, описується формулами:

new_Y = Y * cos (theta) + X * sin (theta) ;

new_X = X * cos (theta) - Y * sin (theta) .

Єдиною складністю при використанні цих формул для графічних дисплеїв буде той факт, що екран дисплея не є декартовим простором. Декартові осі визначають 4 квадранти. Однак екран терміналу є одним з квадрантів, осі X і Y в якому перевернуті: вісь X зліва направо від верхнього кута, а вісь Y - зверху вниз.

Для вирішення цієї проблеми необхідно визначити новий центр і привести у відповідність координати X і Y екрану до координат осей D_X і D_Y декартового простору за виразами:

D_X = X - X0;

D_Y = Y0 - Y.

Будь-яка точка на екрані може бути використана як центр, але зазвичай центр визначається якомога ближче до центру об'єкта, який ми збираємося обертати. Кут обертання слід задавати у радіанах. Таким чином, для обертання точки необхідно виконати наступне:

- "Загасити" крапку, записавши в неї колір фону.
- Привести у відповідність координати дисплея до декартових.
- Повернути точку в декартових координатах на кут theta.
- Відновити нові координати дисплея за виразами:

  X = D_X + X0;

  Y = Y0 - D_Y.
- Вивести точку на екран із заданим кольором.

15.2.3. Обертання плоских фігур у площині екрану. Під об'єктом тут і далі розумітимемо набір сегментів прямих відрізків. Координати крайніх точок кожного відрізка містяться у двовимірному масиві чисел з плаваючою точкою.

Кожен рядок масиву містить початкові та кінцеві координати даного відрізка. Це означає, що перша розмірність масиву являє собою кількість відрізків, що входять до складу об'єкта, а друга розмірність дорівнюватиме 4 (кількість координат крайніх точок відрізка). Наприклад, масив, наведений нижче

double object [3] [4]; визначає об'єкт, що складається із 3 відрізків (див. табл. 15.1).

Таблиця 15.1 – Опис об'єкта (трикутника)

Перший індекс	Другий індекс			
	0	1	2	3
0	start_X1	start_Y1	end_X1	end_Y1
1	start_X2	start_Y2	end_X2	end_Y2
2	start_X3	start_Y3	end_X3	end_Y3

Визначити об'єкт – це означає розмістити у масиві координати початкових і кінцевих точок відрізків, що становлять об'єкт. Наприклад, якщо об'єкт є прямокутником з координатами 0×0 15×10, то масив, що визначає даний прямокутник, заносяться наступні числа:

object[0][0] = 0; object[0][1] = 0; object[0][2] = 0; object[0][3] = 10;
object[1][0] = 0; object[1][1] = 10; object[1][2] = 15; object[1][3] = 10;
object[2][0] = 15; object[2][1] = 10; object[2][2] = 15; object[2][3] = 0;
object[3][0] = 15; object[3][1] = 0; object[3][2] = 0; object[3][3] = 0.

Після того, як об'єкт визначено, ви можете обертати його, використовуючи функцію rotate_object(), за годинниковою стрілкою, задаючи відповідний знак theta або в протилежний бік. При обертанні об'єкта необхідно виконати такі дії:

- Зображення об'єкта шляхом організації циклу за першим індексом масиву object, зчитуючи з відповідного рядка масиву координати двох точок і за допомогою функції line_m() зображення відрізка прямий.
- Організація циклу натискання клавіші (вихід із циклу при натисканні клавіші ESC) всередині якого виконуються дії:
- "Загасання" на екрані об'єкта, яке полягає у зображенні об'єкта фоновим кольором. Якщо "гасити" об'єкт кольором відмінним від фону, то фігура, що обертається, залишатиме за собою слід.
- Повертання кожної точки об'єкта за допомогою розглянутої раніше процедури повороту точки. При цьому треба врахувати, що кожен рядок опису об'єкта містить координати двох точок, тому для кожного рядка необхідно двічі викликати функцію rotate_point з параметрами:

bject [i][0], object[i][1], theta, X0, Y0 – для 1 точки
object[i][2], object[i][3], theta, X0,Y0 – для 2 точки

- Зображення об'єкта з новими координатами.

15.2.4. Обертання тривимірних об'єктів.

Перш ніж обертати тривимірні об'єкти, необхідно розглянути відображення тривимірних об'єктів, заданих відрізками прямих у декартових координатах екрану X, Y. Як і при обертанні плоских фігур вважаємо, що початок системи декартових координат збігається з центром обертання. Без порушення об'єкта розглянемо перетворення координат однієї точки.

Як відомо з курсу інженерної графіки, для відображення тривимірних координат у двовимірні використовуються різні види проекцій.

Найбільш поширеною є ізометрична проекція, у якої вісь D_Z спрямована вертикально вгору, а вісь D_X і D_Y повернені на кут +120 і -120 градусів відповідно. Коефіцієнт стиснення по всіх трьох осях дорівнює 0,97 (для простоти можна взяти 1) .

Іншою проекцією є диметрична кабінетна, у якої вісь D_Z вертикально вгору, вісь D_Y горизонтально вліво, а вісь D_X повернута на +135 градусів щодо осі D_Z (або на -135 градусів щодо осі D_Y). Коефіцієнт стиснення по осях D_Z і D_Y дорівнює 1, а по осі D_X дорівнює 0,5. Таким чином, враховуючи особливості координатної системи екрана (вісь X вліво, а вісь Y вниз), перетворення координат точки виконується:

- для ізометричної проекції

$$X = X0 + \cos(3.1418/6) * (D\_Y - D\_X);$$
$$Y = Y0 - D\_Z + \sin(3.1418/6) * (D\_X + D\_Y);$$

- для кабінетної проекції

$$X = X0 + D\_Y - 0{,}5 * \sin(3.1418 / 4) * D\_X;$$
$$Y = Y0 - D\_Z + 0{,}5 * \sin(3.1418 / 4) * D\_X;$$

де: X0, Y0 – координати центру обертання та початку декартових тривимірних координат у площині екрану;

D_X, D_Y, D_Z - декартові координати точки;

X, Y – координати точки на екрані.

Зробивши перетворення координат 2 точок відрізка, можна побудувати проекції кожного відрізка прямої, які створюють проекцію тривимірного об'єкта на екрані.

Вісь обертання тривимірного об'єкта розташована у просторі довільним чином та розглядаються її складові – проекції осей обертання на координатні вісі D_Z, D_Y, D_X. Тому розглянемо перетворення координат точки під час обертання навколо відповідної вісі (координата вісі обертання не змінюється).

Обертання навколо вісі Z:

$$D\_Y = D\_Y * \cos(\text{theta}) + D\_X * \sin(\text{theta});$$
$$D\_X = D\_X * \cos(\text{theta}) - D\_Y * \sin(\text{theta}) .$$

Обертання навколо вісі Y:

D_X = D_X * cos (theta) + D_Z * sin (theta);
D_Z = D_Z * cos (theta) - D_X * sin (theta) .

Обертання навколо осі X:

D_Z = D_Z * cos (theta) + D_Y * sin (theta);
D_Y = D_Y * cos (theta) - D_Z * sin (theta) .

Тривимірний об'єкт зручно спочатку описувати в декартовій тривимірній системі координат у вигляді двох масивів:

– двомірний масив point описів вершин (точок), у якому перша розмірність відповідає числу точок, а друга розмірність дорівнює 3 і відповідає трьом координатам кожної точки D_X, D_Y, D_Z відповідно.

– двомірний масив опису ребер (ліній, у яких перша розмірність відповідає числу ліній, а другий індекс (розмірність дорівнює 2) містить індекси початкової та кінцевої точки лінії (індекс точки відповідає номеру рядка у попередньому масиві point).

Наприклад, для опису піраміди з трикутною основою (є 4 вершини), масив опису точок має вигляд:

```
double point [ 4 ] [3] = {60, 00, -40, /* координати 1 точки */
                          -40, -60, -40,
                         -40, 60, -40,
                         00, 00, 60}
```

Масив опису ребер має вигляд:

```
int K_point [6] [2] = {0,1, 1,2, 2,0, 0,3, 1,3, 2,3};
```

Використовуючи розглянуті масиви і виконавши перетворення декартових тривимірних координат, програма формує опис об'єкта в координатах екрана аналогічно масиву object в двовимірному випадку.

Таким чином, при обертанні тривимірного об'єкта програма виконує такі дії:

- За заданими масивами point і K_point формує опис проекції об'єкта на екрані – object.
- Виводить зображення об'єкта як у двомірному випадку, використовуючи опис object і функцію line_m.
- Організує цикл очікування натискання клавіші, (вихід із циклу по ESC) усередині якого виконуються такі дії:
- "Загасання " об'єкта, тобто, малювання його кольором фону.
- Повертання точок масиву point навколо заданої вісі. При повороті точок перетворення повороту виконуються для кожного рядка масиву point.
- Формування нового значення масиву об'єкта та виведення зображення на екран.

15.3. Порядок виконання.

15.3.1. Написати програму, яка використовує розроблені в роботі функції bar_m() і line_m() і виконує наступні дії:

15.3.1.1. Переводить систему у графічний режим.

15.3.1.2. За допомогою функції bar_m() зафарбовує частину екрана у різні кольори.

15.3.1.3. Копіює зафарбовану частину екрана до решти екрана.

15.3.1.4. Зберігає отримане зображення у файл (запит про ім'я файлу висвічується у верхніх 14 рядках).

15.3.1.5. Заповнює екран кольором фону.

15.3.1.6. Виводить та обертає плоску фігуру, параметри якої відповідають індивідуальному завданню.

15.3.1.7. Виводить та обертає тривимірний об'єкт, параметри якого відповідають індивідуальному завданню.

15.3.1.8. Повертає систему до текстового режиму.

15.4. Особливості програмування.

15.4.1. Мовою Паскаль.

15.4.1.1. Для звернення до відеопам'яті застосовується попередньо визначений масив Mem. Наприклад, для читання байта за адресою A000:0000h використовується вираз b:=Mem[$ A 000:$0000] (b – змінна типу byte).

15.4.1.2. Для звернення до портів ПК застосовується попередньо визначений масив Port. Наприклад, для запису значення 3Dh до порту 3C0h використовується вираз Port[$3C0]:=$3D; для читання з порту 3DAh використовується вираз data: = Port [$3DA], де data - змінна типу byte.

15.4.2. Мовою С.

15.4.2.1. Для звернення до відеопам'яті використовуються далекі покажчики, які оголошуються у програмі так:

char far*uk; - для роботи з байтами.

Для читання байта за адресою A000:0000 h використовується вираз:

uk = (char far *) 0xA0000000;

b=* uk; де b - змінна типу char;

15.4.2.2. Для доступу до портів ПК застосовуються функції читання та запису у порт, які зберігаються в бібліотеці <dos.h>. Бібліотека підключається командою

#include <dos.h>

після чого, наприклад, для запису значення 3Dh в порт 3C0h використовується вираз:

outportb(0x3C0,0x3D);

Для читання з порту 3DAh використовується вираз:
data=intportb(0x3DA),де data- змінна типу char;

15.5. Індивідуальні завдання

За першою цифрою номера студента у журналі вибрати проекцію для тривимірного об'єкта.

0 – ізометрична

1 – диметрична кабінетна.

У табл. 15.2 за другою цифрою номера студента у журналі вибрати параметри двовимірного та тривимірного об'єкта.

Таблиця 15.2 – Варіанти індивідуальних завдань

Параметри	Друга цифра в номері по журналу _									
	0	1	2	3	4	5	6	7	8	9
obj_XY	4	3	4	5	5	4	6	5	3	6
obj_XYZ	15	44	13	33	44	14	44	15	55	33
Колір фону	4	0	1	2	3	4	3	1	2	0
Колір об'єкту	B	E	C	D	A	B	8	4	5	6
theta	-0.12	0.05	-0.05	0.06	0.08	0.1	-0.08	0.1	-0.01	0.12
Вісь обертання	X	X	Y	Z	X	Y	X	Y	Z	X

де : obj_XY – тип плоского опуклого багатокутника: 3 – трикутник, 4 – квадрат тощо; obj_XYZ – тип тривимірного об'єкта, який у разі рівності двох цифр є прямокутною призмою, а кожна цифра відповідає числу сторін основи. Якщо одна цифра дорівнює 1, то тривимірний об'єкт є пірамідою. Наприклад, 13 - піраміда з трикутною основою 44 -чотирикутна призма (паралелепіпед) і т.д.

Центр обертання та розмір об'єкта студент обирає самостійно.

15.6. Зміст звіту

15.6.1. Тема лабораторної роботи

15.6.2. Мета роботи.

15.6.3. Індивідуальне завдання.

15.6.4. Текст програми.

15.6.5. Результати роботи програми.

15.6.6. Висновки.

ЛІТЕРАТУРА

1. Архітектура комп’ютера. Частина 1: навчальний посібник/ Кравченко Ю.В., Лещенко О.О., Герасименко О.Ю., Труш О.В., Дахно Н.Б. – К. : КНУ імені Тараса Шевченка, 2022. – 259 с.

2. Тарарака В.Д. Архітектура комп'ютерних систем: навчальний посібник. – Житомир : ЖДТУ, 2020. – 383 с.

3. James L. Antonakos Pentium Microprocessor. – Pearson 2020 – 539 P.

4. Архітектура комп'ютерів. Методичні вказівки до виконання та оформлення курсового проекту для студентів денної та заочної форми навчання за напрямком 123 «Комп’ютерна інженерія» / Поворознюк А. І., Поворознюк О. А., Філатова Г.Є. – Харків: НТУ "ХПІ" , 2022. – 42 с.

5. Povoroznyuk A. I., Povoroznyuk O. A., Filatova G.E. Computer architecture. Methodical instructions guidelines for the implementation and design of a course project for full-time and part-time students of speciality 123 "Computer Engineering" – Kharkiv: NTU "KhPI", 2024. - 42 p.

6. O.A. Povoroznyuk Laboratory workshop on the course “Computer Architecture” for full-time and part-time students of speciality 123 "Computer Engineering” / O.A. Povoroznyuk, A. I. Povoroznyuk. – Kharkiv: NTU “KhPI”, 2024 – 82 с.

7. Крупельницький, Л.В. Архітектура комп’ютерів. Частина 1 : лабораторний практикум / Л.В. Крупельницький, А.В. Снігур, С.В. Богомолов – Вінниця : ВНТУ, 2020. – 104 с.

Навчальне видання

ПОВОРОЗНЮК Анатолій Іванович
МЕЗЕНЦЕВ Микола Вікторович
ПОВОРОЗНЮК Оксана Анатоліївна

АРХІТЕКТУРА КОМП'ЮТЕРІВ

Лабораторний практикум

Відповідальний за випуск проф. Заковоротний О.Ю.

Роботу до видання рекомендував проф. М.Й. Заполовський

В авторській редакції

План 2024, поз. 92

Підп. до друку_____ Гарнітура Times New Roman.
Видавничий центр НТУ «ХПІ», вул. Кирпичова, 2, м. Харків, 61002 Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
Електронне видання