

**ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО
МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
імені Олеся Гончара**

**Методичні вказівки
до виконання лабораторних робіт
з дисципліни
«Архітектура комп'ютерів»**

**Дніпро
2023**

Зміст

ЛАБОРАТОРНА РОБОТА № 1	3
ЛАБОРАТОРНА РОБОТА №2	10
ЛАБОРАТОРНА РОБОТА №3	16
ЛАБОРАТОРНА РОБОТА №4	22
ЛАБОРАТОРНА РОБОТА №5	32
ЛАБОРАТОРНА РОБОТА №6	40
ЛАБОРАТОРНА РОБОТА №7	46
ЛАБОРАТОРНА РОБОТА №8	54
ЛАБОРАТОРНА РОБОТА №9-10	57
ЛАБОРАТОРНА РОБОТА №11	63
ЛАБОРАТОРНА РОБОТА №12	74
ЛАБОРАТОРНА РОБОТА №13	80
ЛАБОРАТОРНА РОБОТА №14	88

ЛАБОРАТОРНА РОБОТА № 1

Тема: *Визначення складу ПК. BIOS.*

Мета: *Вміти визначити типову архітектуру ПК и виконувати порівняльний аналіз наданих конфігурацій за критеріями, вказаними користувачем. Ознайомлення з базовою системою введення/виведення BIOS. З його архітектурою та організацією, з основними розділами та командами. Навчитись працювати у BIOS. Змінювати параметри ПК за допомогою BIOS.*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Призначення основних блоків ПК



До основних компонентів ПК відносяться: системний блок, монітор, клавіатура, маніпулятор типу «миша».

У структуру системного блоку входять:

1. корпус;
2. материнська плата;
3. процесор;
4. оперативна пам'ять;
5. жорсткої диск;
6. накопичувач компакт (або DVD) дисків;
7. відеокарти;
8. звукової карти.

За конструкцією системні блоки (SU - SystemUnit) можуть бути:

- настільні (Desk Top),
- підлогові (Desk Size),
- вертикальні (Tower),
- переносні (Lap Top, Brief-Case-Size),
- мініатюрні (Book-Size, Pocket, Hand-Held, Note-Book).

Системний блок (SU) завжди містить:

- системну плату (SB - System Board), яка об'єднує навколо локальної шини (LB) мікропроцесора всі електронні компоненти підсистем і периферійних пристроїв введення-виведення,
- імпульсний блок живлення (Power Supply),
- пристрої підсистеми зовнішніх запам'ятовуючих пристроїв (ВЗП),
- систему примусового охолодження (вентиляції),

- набір карт адаптерів USB (I/O Card Adapter), що встановлюються в роз'єми розширення системної шини, але в деяких типах PC адаптери USB можуть бути встановлені і прямо на SB.

Подання компонентів ПК за допомогою засобів «Диспетчера пристроїв».

(Мій комп'ютер / Властивості системи / Устаткування / Диспетчер пристроїв)

Загальні відомості про диспетчері пристроїв

У вікні диспетчера пристроїв представлено графічне зображення обладнання, що встановлено на комп'ютері. Диспетчер пристроїв використовують для оновлення драйверів (або програмного забезпечення) обладнання, зміни настройки обладнання, а також для усунення неполадок і для вирішення конфліктів пристроїв і зміни параметрів ресурсів.

Диспетчер пристроїв дозволяє:

- визначати правильність роботи обладнання комп'ютера;
- змінювати параметри конфігурації обладнання;
- визначати драйвери пристроїв, що завантажуються для кожного пристрою, і отримувати відомості щодо кожного драйвера;
- змінювати додаткові параметри і властивості пристроїв;
- встановлювати оновлені драйвери пристроїв;
- відключати, включати і видаляти пристрої;
- здійснювати повернення до попередньої версії драйвера;
- роздруковувати список пристроїв, встановлених на комп'ютер.

Налаштування пристроїв

При установці пристрою Plug and Play Windows автоматично налаштовує його, забезпечуючи його правильну роботу з іншими встановленими на комп'ютері пристроями. В ході процесу настройки Windows призначає пристрою, що встановлюється, унікальний набір системних ресурсів. Ці ресурси можуть включати один або декілька з наступних параметрів:

- номери рядків запиту на переривання (IRQ).
- канали прямого доступу до пам'яті (DMA).
- адреси портів вводу / виводу (I / O).
- діапазони адрес пам'яті.

Кожен ресурс, який призначається пристрою, повинен бути унікальним. Це необхідно для правильної роботи пристрою. Для пристроїв Plug and Play Windows автоматично перевіряє правильність настройки ресурсів.

Іноді двом пристроям потрібні однакові ресурси, що призводить до конфлікту пристроїв. В цьому випадку необхідно вручну змінити налаштування ресурсів таким чином, щоб всі параметри були унікальними. Деякі ресурси, наприклад переривання пристроїв PCI, можуть в залежності від драйверів і комп'ютера використовуватися спільно.

При установці пристроїв не Plug and Play автоматична настройка ресурсів не проводиться. Деякі типи пристроїв потрібно налаштовувати вручну. За інструкціями містяться в посібника, який постачався з пристроєм.

Змінювати параметри ресурсів вручну зазвичай не рекомендується, оскільки при цьому значення фіксуються, що знижує можливості Windows по виділенню ресурсів для інших пристроїв. Якщо зафіксовано занадто багато значень параметрів для окремих ресурсів, Windows не зможе автоматично встановлювати нові пристрої Plug and Play.

Загальні уявлення о програмі «Інформація про систему»

Програма «Інформація про систему» – це засіб для технічної підтримки, що дозволяє швидко збирати дані про комп'ютер та його операційну систему. В лівій частині вікна програми знаходиться дерево категорій, подібно до дерева папок провідника Windows. В правій частині вікна програми знаходиться інформація, в якій наводяться дані, що відносяться до елемента, виділеного в дереві категорій.

«Інформація про систему» – це коріння вузла в дереві категорій програм. Коли він виділений, в області інформування відображаються дані загальної характеристики комп'ютера та його операційної системи. Тут можна дізнатись назву операційної системи, її версію, виробника та місцезнаходження системного каталога. Крім того, можна перевірити версію BIOS або EFI, тип процесора та об'єм пам'яті.

За допомогою кореневого вузла можна:

- дізнатися версію та дату виготовлення BIOS, визначити каталог, в якому встановлена операційна система;
- переконатися в тому, що нова пам'ять встановлена правильно;
- перевірити параметр Файл сторінки, при наявності порушення в роботі пам'яті. Значення цього параметра відповідає розміру фізичного простору на жорсткому диску, що використовується операційною системою для збільшення віртуального розміру ОЗУ;
- перевірити стан активізації продукту.

Ключові компоненти системи:

1. Ресурси апаратури – зібрані дані про призначення ресурсів і можливих конфліктах, обумовлених спільним використанням каналів DMA, обладнання зі зворотним зв'язком, портів введення-виведення, ліній IRQ і адрес пам'яті.
2. Компоненти – мультимедіа, CD-ROM, звуковий пристрій, дисплей, інфрачервоні пристрої, введення, модем, мережа, порти, запам'ятовуючі пристрої, друк, пристрої з помилками, USB
3. Програмне середовище – включені дані про конфігурацію системи, в тому числі відомості про системні драйвери, змінних середовища і наявних завданнях друку. Елементи, що входять в категорію: системні драйвери; сертифіковані драйвери; змінні середовища; завдання друку; мережеві підключення; завантажені модулі; служби; групи програм; програми, що завантажуються автоматично; реєстрація OLE; повідомлення про помилки Windows
4. Параметри оглядача – деякі мережеві настройки Internet Explorer.
5. Додатки – ця програма є інформаційною і не дозволяє змінювати будь-які налаштування і параметри, отримавши докладну інформацію, виявивши конфлікт в розподілі ресурсів або встановивши причину несправності, слід звернутися до інших засобів.

Специфікація Plug & Play ("підключай і працюй") була розроблена фірмою Intel, при безпосередній участі фірм IBM і Microsoft, в 1993 році. Вона дозволяє виконувати автоматичне конфігурування комп'ютера при підключенні до нього додаткового периферійного обладнання або його заміни на нове. Специфікація Plug & Play спеціально розроблена для розпізнавання і узгодження всіх змін в конфігурації комп'ютера без втручання користувача. Суть роботи специфікації полягає в тому, щоб комп'ютер міг самостійно розпізнати встановлене в ньому обладнання і відповідним чином виділити для нього необхідні ресурси. Користувачеві немає необхідності замислюватися про адреси введення / виводу, каналах прямого доступу до пам'яті, лініях запиту на переривання і т.і. Практична реалізація специфікації пов'язана не тільки зі схемотехнічними рішеннями, але, і в рівній мірі, з програмним забезпеченням. Це можливо, якщо в реалізації специфікації приймають участь наступні три компоненти:

- Апаратні засоби, що підтримують Plug & Play;
- BIOS;
- Операційна система.

Під апаратними засобами, що підтримують специфікацію Plug & Play слід розуміти як системні плати, так і плати адаптерів, що встановлюються в слоти розширення. В BIOS, що підтримує цю специфікацію, додатково включено півтора десятка функцій, використовуваних надалі операційною системою. В процесі роботи апаратні засоби інформують BIOS і операційну систему про те, що це за пристрої, і які ресурси їм необхідні для нормальної роботи. Операційна система виконує завантаження відповідних драйверів і, при необхідності, дозволяє вручну налаштувати їх параметри. Вперше специфікація Plug & Play була включена, як складова частина в операційні системи Windows 95 і Windows NT.

Частина II

BIOS (англ. Basic Input/Output System – базова система введення/виведення) – є набором спеціальних підпрограм, які використовуються комп'ютерами архітектури x86 для ініціалізації компонентів персональної платформи, необхідних для її первинного завантаження та подальшої роботи. Такими є процесор, системна логіка (чипсет), оперативна пам'ять, клавіатура, відеокартка та інші.

Всі сучасні комп'ютери мають утиліту *Setup*, вбудовану в BIOS (зображена на рисунку 1). Утиліта BIOS Setup має інтерфейс у вигляді меню, іноді навіть віконний з підтримкою миші. Віконний інтерфейс в даному випадку незручний, оскільки замість швидкого входу в текстове меню комп'ютер довго шукає підключену мишу, після чого виводить вікна в режимі графіки низького дозволу (дань сумісності). При цьому ніяких принципово нових можливостей (в порівнянні з текстовим режимом і управлінням від клавіатури) не з'являється.

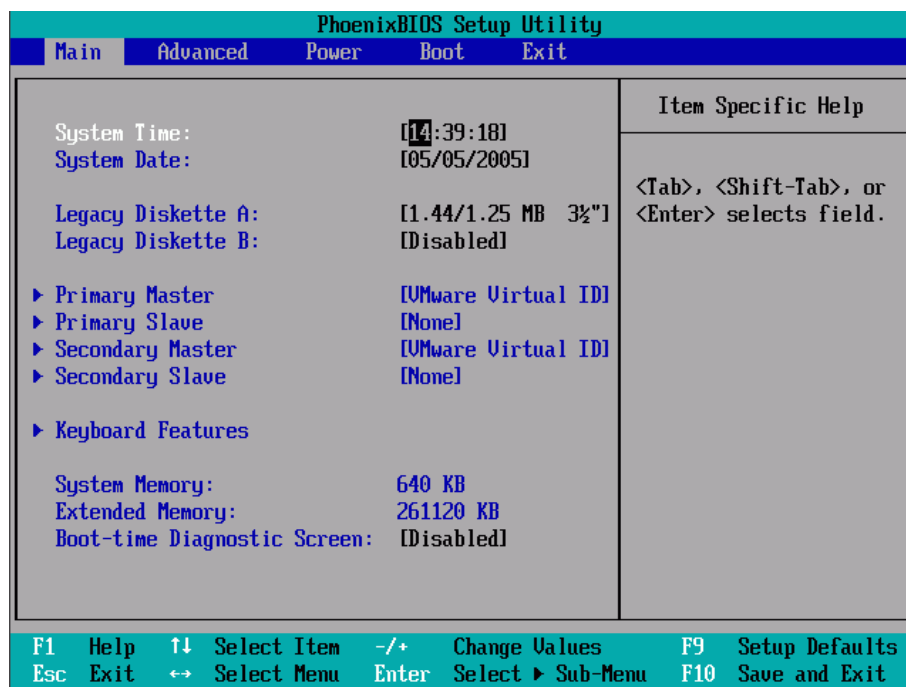


Рисунок 1 – Вікно утиліти Setup, що вбудована в BIOS

Для входу в Setup під час виконання POST з'являється пропозиція натискувати клавішу Del. Іноді для цього використовується комбінація Ctrl+Alt+Esc, Esc, Ctrl+Esc, бувають і екзотичні варіанти (натискувати клавішу F12 в ті секунди, коли в правому верхньому кутку екрану видний прямокутник). Деякі версії BIOS дозволяють увійти в Setup покомбінації Ctrl+Alt+Esc у будь-який момент роботи комп'ютера. Пропозиція (і спосіб – натиснення F1 чи F2) входу в Setup з'являється, якщо POST знайде помилку устаткування, яка може бути усунена за допомогою Setup. Утримання клавіші Ins під час POST у ряді версій BIOS дозволяють встановити настройки за умовчанням, відміняючи всі "прискорювачі". Це допомагає відновити працездатність після надмірно агресивних спроб "розігнати" комп'ютер.

Початковий запуск і самотестування

При включенні блок живлення виробляє сигнал апаратного скиду, що приводить все вузли в початковий стан. Цей же сигнал виробляється і при натисканні кнопки *RESET*. Процесор готується до роботи, сприймаючи зі своїх виводів сигнали, що задають його конфігурацію (коефіцієнт множення, роль в багато процесорних системах і деякі інші параметри). Внутрішній кеш очищується, реєстри (в повному обсязі) наводяться в певному стані. Сигнал скиду надходить на всі пристрої (контролери і адаптери), які потребують переведення в початковий стан. Після закінчення сигналу процесор за певною адресою вибирає з пам'яті і виконує першу інструкцію – управління передається на точку входу в програму ініціалізації комп'ютера (**POST**).

Першою справою необхідно виконати ініціалізацію процесора – встановити значення деяких реєстрів. Після скиду процесор завжди починає роботу в реальному режимі (сумісному з 8086/88). Далі відбуваються перевірка працездатності та ініціалізація підсистем комп'ютера. Це виконується в кілька етапів. Якщо ПЗП справний, можна рухатися далі, в іншому випадку краще зупинитися. Далі ініціалізується ОЗП (програмується

реєстри чіпсета, завідувачі налаштуванням контролера пам'яті і регенерацією) і виконують тестування невеликого блоку на початку ОЗП.

Якщо тест проходить успішно, то для подальшої роботи вже можна користуватися і викликами, і перериваннями (не забивши проініціалізувати таблицю переривань), і задіяти пам'ять для зберігання змінних. Потім ініціалізується і тестується дисплейний адаптер. Далі тестують ОЗП в повному обсязі, визначають наявність контролерів і адаптерів, ініціалізують їх і тестують.

Після цього програма **POST** дізнається реальну конфігурацію комп'ютера і готова до завантаження операційної системи. Вектори переривань, за які відповідає **BIOS**, проініціалізовані – ними можна користуватися. У цього моменту можна увійти в меню вбудованої утиліти конфігурації **CMOS SETUP** і змінити налаштування різних підсистем комп'ютера. Після закінчення роботи цієї утиліти тест **POST** доводиться виконувати знову – конфігурація може стати вже іншою.

До моменту закінчення тесту **POST** всі стандартні пристрої (клавіатура, дисплей, диски, порти) наводяться в стан готовності до роботи в стандартному режимі за замовчуванням, частина налаштувань може виконуватися відповідно до вибраних установок **CMOS SETUP**. Список виявлених пристроїв і їх основні параметри можуть відображатися в таблиці на екрані, але можлива і така настройка **CMOS SETUP**, при якій екран буде пустим до появи логотипу (або текстового повідомлення) завантажувача ОС.

Останнім кроком програми **POST** є виконання процедури початкового завантаження (bootstrap loader), яка викликається як програмне переривання *Int 19h BIOS*. Її завдання - з вибраного пристрою завантажити в пам'ять один блок (сектор) даних, і якщо він схожий на завантажувач – передати йому управління. На цьому діяльність **POST** закінчується, і комп'ютер віддається у владу завантажувача ПЗ.

Програма MyBIOS

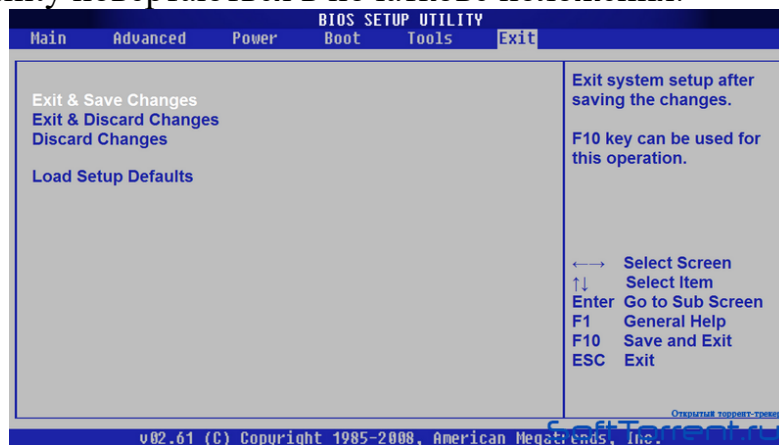
Програма MyBIOS має повну схожість з інтерфейсом і меню оригінальної програми BIOS Setup Utility материнської плати ASUS P5K.

Програма призначена в першу чергу для вищих і середніх спеціальних навчальних закладів, в яких може бути використана для проведення практичних занять, мета яких – навчити студентів налаштовувати різні опції BIOS. Без використання даної програми проведення подібних практичних занять ускладнюється тим, що неправильне налаштування BIOS може привести до непрацездатності або до збоїв в роботі комп'ютера.

Програма MyBIOS не виконує реальної настройки пристроїв на комп'ютері, вона тільки емулює меню і опції BIOS Setup Utility материнської плати, тому не може порушити працездатність комп'ютера.

Важливою особливістю програми є наявність режиму роботи, при якому студентам даються завдання з налаштування BIOS, після виконання яких виводиться оцінка в балах (один бал за кожне правильно виконане завдання). Список доступних завдань згрупований по розділах меню BIOS, завдання зі списку вибираються в довільному порядку.

Після виходу з емулятора і його повторного запуску все значення опцій BIOS Setup Utility повертаються в початкове положення.



ХІД РОБОТИ

Частина I

1. За допомогою «Диспетчера пристроїв» визначити конфігурації складових частин ПК наданого у лабораторії коледжу, а також будь-якого іншого комп'ютера.
2. Інформацію, яка характеризує визначені складові ПК представити у таблиці (див. Таблиця 1).
3. За допомогою спеціального програмного забезпечення, наприклад Everest, визначити конфігурації складових частин ПК наданого у лабораторії коледжу, а також будь-якого іншого комп'ютера та проілюструйте скрин-шотами.

Таблиця 1

Найменування пристрою	Тех. характеристики (модель)	Примечание (описание тех. характеристик)

Частина II

1. Запустити емулятор MyBIOS
2. Виконати завдання у емуляторі MyBIOS

ЗМІСТ ЗВІТУ

Звіт повинен містити найменування роботи, мету, виконані завдання, записи досліджень і вимірів, висновки по виконаній роботі.

ЛАБОРАТОРНА РОБОТА №2

Тема: Центральний процесор персонального комп'ютера.

Мета: Ознайомитись з призначенням, конструкцією та характеристиками центрального процесора персонального комп'ютера.

Теоретична частина

Процесор – головна мікросхема комп'ютера, його "мозок". Він дозволяє виконувати програмний код, що знаходиться у пам'яті і керує роботою всіх пристроїв комп'ютера. Швидкість його роботи визначає швидкодію комп'ютера. Конструктивно, процесор – це кристал кремнію дуже маленьких розмірів. Процесор має спеціальні комірки, які називаються регістрами. Саме в цих регістрах містяться команди, які виконуються процесором, а також дані, якими оперують ці команди. Робота процесора полягає у вибиранні з пам'яті у певній послідовності команд та даних і виконанні їх. На цьому і базується виконання програм. У ПК обов'язково має бути присутній центральний процесор (Central Rprocessing Unit – CPU), який виконує всі основні операції. Часто ПК оснащений додатковими співпроцесорами, орієнтованими на ефективне виконання специфічних функцій, такими як, математичний співпроцесор для обробки числових даних у форматі з плаваючою точкою, графічний співпроцесор для обробки графічних зображень, співпроцесор введення/виведення для виконання операції взаємодії з периферійними пристроями.

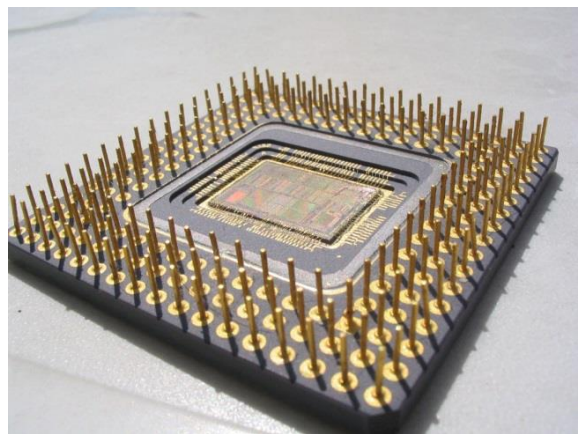


Рисунок 1 – Зовнішній вигляд процесору.

Основними параметрами процесорів є:

- ☐ тактова частота,
- ☐ розрядність,
- ☐ робоча напруга,
- ☐ коефіцієнт внутрішнього домноження тактової частоти,
- ☐ розмір кеш пам'яті.

Тактова частота визначає кількість елементарних операцій (тактів), що виконуються процесором за одиницю часу. Тактова частота сучасних процесорів вимірюється у Гц (1 Гц відповідає виконанню однієї операції за

одну секунду). Чим більша тактова частота, тим більше команд може виконати процесор, і тим більша його продуктивність. Перші процесори, що використовувалися в ПК працювали на частоті 4,77 МГц, а сьогодні робочі частоти найсучасніших процесорів досягли позначки в 3 ГГц (1 ГГц = 103 МГц).

Розрядність процесора показує, скільки біт даних він може прийняти і обробити в свої регістрах за один такт. Розрядність процесора визначається розрядністю командної шини, тобто кількістю провідників у шині, по якій передаються команди. Сучасні процесори сімейства Intel є 32-розрядними.

Робоча напруга процесора забезпечується материнською платою, тому різним маркам процесорів відповідають різні материнські плати. Зараз робоча напруга процесорів не перевищує 3 В. Пониження робочої напруги дозволяє зменшити розміри процесорів, а також зменшити тепловиділення в процесорі, що дозволяє збільшити його продуктивність без загрози перегріву.

Коефіцієнт внутрішнього домноження тактової частоти – це коефіцієнт, на який слід помножити тактову частоту материнської плати, для досягнення частоти процесора. Тактові сигнали процесор отримує з материнської плати, яка з чисто фізичних причин не може працювати на таких високих частотах, як процесор.

Кеш-пам'ять. Обмін даними всередині процесора відбувається набагато швидше ніж обмін даними між процесором і оперативною пам'яттю. Тому, для того щоб зменшити кількість звертань до оперативної пам'яті, всередині процесора створюють так звану надоперативну або кеш-пам'ять. Коли процесору потрібні дані, він спочатку звертається до кеш-пам'яті, і тільки якщо там потрібні дані відсутні, відбувається звертання до оперативної пам'яті. Чим більший розмір кеш-пам'яті, тим більша ймовірність, що необхідні дані знаходяться там. Тому високопродуктивні процесори оснащуються підвищеними обсягами кеш-пам'яті. Розрізняють кеш-пам'ять першого рівня (виконується на одному кристалі з процесором), другого рівня (виконується на окремому кристалі, але в межах процесора) та третього рівня (виконується на окремих швидкодійних мікросхемах із розташуванням на материнській платі).

Шини. З іншими пристроями, і в першу чергу з оперативною пам'яттю, процесор зв'язаний групами провідників, які називаються шинами. Основних шин три:

- ☐ шина даних,
- ☐ адресна шина,
- ☐ командна шина.

Адресна шина. Дані, що передаються по цій шині трактуються як адреси комірок оперативної пам'яті. Саме з цієї шини процесор зчитує адреси команд, які необхідно виконати, а також дані, із якими оперують команди. У сучасних процесорах адресна шина 32-розрядна, тобто вона складається з 32 паралельних провідників.

Шина даних. По цій шині відбувається копіювання даних з оперативної пам'яті в регістри процесора і навпаки. У сучасних ПК шина даних 64-

розрядна. Це означає, що за один такт на обробку поступає відразу 8 байт даних.

Командна шина. По цій шині з оперативної пам'яті поступають команди, які виконуються процесором. Команди представлені у вигляді байтів. Прості команди вкладаються в один байт, але є й такі команди, для яких потрібно два, три і більше байтів. Більшість сучасних процесорів мають 32-розрядну командну шину, хоча існують 64-розрядні процесори з командною шиною.

Процесор вставляється в спеціальне місце – сокет (англ. socket – гніздо, розетка) на системній (материнській) платі (рис. 2).

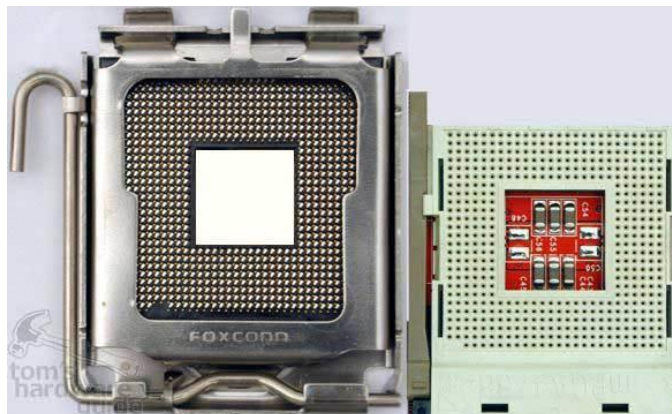


Рисунок 2 – Зовнішній вигляд сокету.

Охолодження процесорів

Неминучість нагріву. У міру підвищення обчислювальної продуктивності процесорів ПК вони більше споживають електроживлення і сильніше нагріваються, а отже, збільшується і тепловиділення. Так, наприклад, якщо для процесора Celeron значення потужності не перевищувало і 20 Вт, то для Pentium III, Duron це значення зросло до 30 – 40 Вт, а для Pentium IV і Athlon вже склало більше 80 Вт. Якщо не розсіювати тепло, що виділяється, то процесор перегрівається і відмовляється працювати. Щоб уникнути цього, необхідно ефективне охолодження. Можна виділити три технології охолодження, застосовувані в обчислювальній техніці.

Повітряне охолодження. Ця технологія набула найбільшого поширення в світі ПК. Для охолодження процесора на нього встановлюється радіатор (рис. 3, а), а на радіатор – вентилятор (рис. 3, б). Така комбінація приладів охолодження називається кулером.

Основні характеристики радіатора – це матеріал, з якого він виготовлений, а також чистота контактної поверхні між радіатором і процесором. Збільшення коефіцієнтів тепловіддачі та теплопередачі досягається підбором матеріалу радіатора. Радіатори виготовляються з алюмінію і міді (або з додаванням міді).

Через мікроскопічні нерівності між процесором і радіатором неминучим являється повітряний прошарок, який негативно позначається на теплообміні між процесором і радіатором. Для цих цілей застосовуються різні силікономістка термопласти, які покращують передачу тепла радіатора.



а



б

Рисунок 3 – Система охолодження процесорів.

Електричне охолодження. Кулери Пельтьє засновані на явищі Пельтьє, суть якого полягає у виділенні або поглинанні тепла на контакті двох різних провідників в залежності від напрямку електричного струму. Цей ефект виявив французький фізик Жан Пельтьє, коли пропустив постійний струм через смужку вісмуту, підключену за допомогою двох мідних дрітків. Він зауважив, що з'єднання «мідь-вісмут» (струм від міді до вісмуту) нагрівається, інша сполука – «вісмут-мідь» (струм від вісмуту до міді) – охолоджується. Було відмічено, що кількість виділеної теплоти пропорційна силі струму. Якщо подати на пластину елементів Пельтьє сильний струм, то одна її сторона нагріється, а інша – охолоне. Холодний бік встановлюють на процесор, а гарячий з'єднують з радіатором.

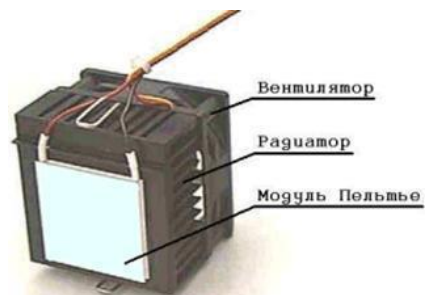
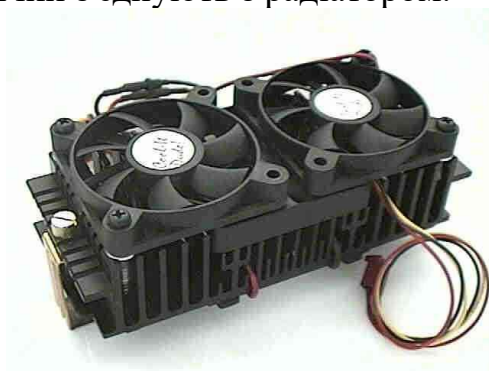


Рисунок 4 – Електричне охолодження процесорів.

Водяне охолодження. Принцип дії водяного (рідинного) охолодження подібний системі повітряного охолодження. Необхідність циркуляції рідини в охолоджувачі вимагає наявності в ньому таких елементів, як трубки (як правило, з силікону), по яких тече охолоджена рідина, і водяного насоса, що забезпечує її циркуляцію. Перевагами такої системи є висока якість охолодження і значне зниження шуму. Але в той же час виникає проблема герметичності контурів охолодження.

Термопасти. Термопасти створюються на основі порошкоподібних матеріалів, а в'язкою сполукою у них служить силікон. В якості порошкоподібних складових виступають оксид цинку, нітрит алюмінію і графіт. Термопасти ефективно відводять тепло, якщо їх перетворити на

якісний тонкий прошарок між процесором і кулером. Якщо доведеться знімати кулер, то необхідно ретельно видалити стару і нанести нову пасту.

Практична частина

Завдання 1. Запишіть у вигляді таблиці значення наступних параметрів мікропроцесору вашого персонального комп'ютера.

Параметр	Що характеризує	Одиниці вимірювання	Значення параметру вашого процесору
Розрядність			
Тактова частота			
Швидкодія			
Кеш-пам'ять			
Кількість ядер			

Завдання 2. Проведіть ретроспективний аналіз поколінь центральних процесорів. Для цього:

1. Визначте основні ознаки призначення центральних процесорів.
2. Опишіть ознаки структури центральних процесорів кожного покоління.
3. Визначте принцип дії процесорів. Надайте характеристику техпроцесу виготовлення процесорів.
4. Визначте залежності між принципом дії центрального процесору та його структурою, призначенням і параметрами.
5. Визначте основні технологічні, функціональні, економічні та ергономічні характеристики центральних процесорів. Отримані дані запишіть у вигляді таблиці.

Завдання 3. Визначте найперспективніші технології виробництва процесорів. Для цього:

1. За кількісними показниками функціональних критеріїв процесорів побудуйте S-криву їх розвитку у часі.
2. Визначте за отриманою S-кривою сучасний етап еволюції процесорів.
3. На основі ретроспективного аналізу та визначених у завданні 2 закономірностей визначте принцип дії, який може бути покладений в основу розробки нових поколінь процесорів для покращення їх характеристик.

Контрольні питання

1. В чому полягає виконання програм центральним процесором?
2. Які основні параметри процесора?
3. Для чого призначені шини? Які є типи шин?
4. Охарактеризуйте системи охолодження процесора ПК.
5. Назвіть основні ознаки призначення центральних процесорів.
6. Опишіть ознаки структури центральних процесорів кожного покоління.

7. Який принцип дії центральних процесорів. Надайте характеристику техпроцесу виготовлення центральних процесорів.
8. Яким чином принцип дії центрального процесору обумовлює його структуру, призначення та характеристики?
9. На якому етапі еволюції знаходяться сучасні покоління процесорів.
10. Які перспективні технології виробництва процесорів є найбільш перспективними?
11. Який принцип дії може бути покладений в основу розробки нових поколінь центральних процесорів для покращення їх характеристик?

ЛАБОРАТОРНА РОБОТА №3

Тема: Внутрішня пам'ять персонального комп'ютера.

Мета: Ознайомитись з призначенням, конструкцією та характеристиками внутрішньої пам'яті персонального комп'ютера.

Теоретична частина

Внутрішні запам'ятовуючі пристрої поділяють на **оперативні** (ОЗП) і **постійні** (ПЗП).

ОЗП виконується у вигляді інтегральних мікросхем. В одній елементарній комірці ОЗП зберігається 1 біт інформації у вигляді електричного сигналу чи заряду. Інформацію в ОЗП можна як записати, так і зчитати необмежену кількість разів. Інформація в комірці ОЗП змінюється тоді, коли у неї записується нова інформація. Доступ до комірки ОЗП здійснюється по адресним шинам практично миттєво (декілька наносекунд). Тому ОЗП – найшвидший вид пам'яті – часто позначають RAM (Random Access Memory – пам'ять з довільним доступом).

Від кількості встановленої в ПК ОП залежить, з якими програмами можна з ними працювати. При недостатній кількості ОП програми чи зовсім не будуть працювати, чи стануть працювати досить повільно.

Різні типи оперативної пам'яті

Чим швидша оперативна пам'ять, тим краще. Швидкість пам'яті визначається **частотою** її шини, яка залежить від типу пам'яті. Сьогодні можна зустріти оперативну **пам'ять наступних типів** (розміщені за хронологією появи):

- **SDR SDRAM** (тактова частота шини 66 – 133 МГц);
- **DDR SDRAM** (100 – 267 МГц);
- **DDR2 SDRAM** (400 – 1066 МГц);
- **DDR3 SDRAM** (800 – 2400 МГц).

Всі ці типи пам'яті з'являлися по черзі, кожна нова версія отримувала значні поліпшення в порівнянні з попередньою. Вони не сумісні одна з одною. Тому в комп'ютер оснащений роз'ємом для пам'яті DDR не можна підключити пам'ять DDR2, і так далі.

Щоб уникнути помилкової установки пам'яті в материнську плату, модулі пам'яті мають різну і не сумісну один з одним форму. Це видно на рис. 1.

Принцип роботи пам'яті зазначених типів однаковий. Вони обробляють потік команд процесора як своєрідний **конвеєр**. Головною особливістю цього конвеєра є те, що при надходженні до запам'ятовуючого пристрою команди зчитування, дані на виході з'являються не відразу, а через який час (через деяку кількість тактів шини). Це час називається **затримкою або таймінгами пам'яті** (англ. SDRAM latency) і чим він коротший, тим пам'ять продуктивніша. Цей параметр, як і частоту шини, також потрібно враховувати при виборі ОЗП.

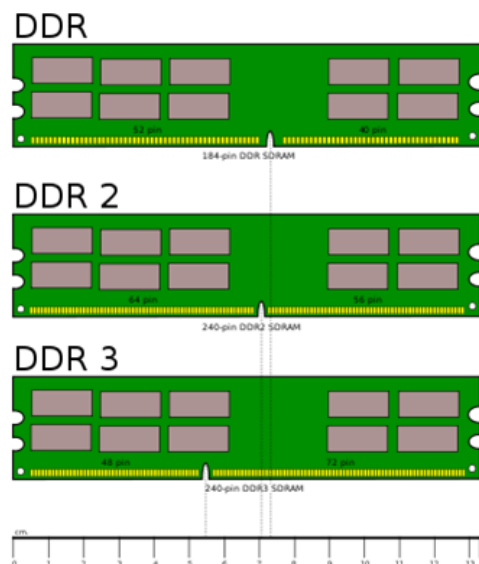


Рисунок 1 – Різні форми модулів пам'яті.

Наприклад, є два модулі ОЗП одного типу з частотою шини 800 МГц і затримками пам'яті 4-4-4 і 5-5-5. З них продуктивнішим буде перший варіант. Складніше порівняти пам'ять з різними частотами. Як правило, в модулях пам'яті з більш високою частотою вищими виявляються і затримки, і виграв у швидкості від цієї частоти насправді буде не настільки великим, як здається на перший погляд. Наприклад, DDR3-1333МГц з таймінгами 9-9-9 лише трохи випереджає DDR2-800МГц з затримками 4-4-4, а DDR3-1333МГц із затримками 7-7-7 по продуктивності приблизно дорівнює DDR2-1067МГц.

Але майбутнє все ж за більш новими типами оперативної пам'яті. Вже розроблена DDR4 SDRAM (2133 – 4266 МГц).

Різні типи модулів ОЗП істотно відрізняються також і зовні (роз'ємом, кількістю контактів і т.д.). Якщо материнська плата розрахована на використання одного типу пам'яті, встановити на неї інший тип оперативної пам'яті не можна, оскільки навіть фізично в слот він не ввійде. Існують перехідники, що дозволяють встановлювати модулі DDR2 в слоти DDR, але широкого поширення вони не набули, оскільки використовувати їх можна тільки на материнських платах, системна логіка яких підтримує роботу одночасно з DDR і DDR2.

Крім швидкості роботи, важливою характеристикою оперативної пам'яті є також її **об'єм**, який повинен відповідати колу завдань, що вирішуються за допомогою комп'ютера, а також встановленому на ньому програмному забезпеченню. Наприклад, офісному комп'ютеру з системою Windows XP для роботи з текстом, перегляду сторінок Інтернету та здійснення інших нескладних операцій цілком достатньо навіть 512 МВ оперативної пам'яті. Якщо на комп'ютері буде встановлена операційна система Windows7, для вирішення тих же завдань потрібно буде вже як мінімум 2048 МВ ОЗУ, оскільки сама Windows7 вимагає більше пам'яті. Якщо в системі буде недостатньо пам'яті, то при запуску ресурсомістких програм вільна пам'ять може закінчитися. У цьому випадку комп'ютер для її розширення буде використовувати частину жорсткого диска (так званий файл підкачки або

swar-файл, спеціально зарезервований операційною системою). Враховуючи, що швидкість доступу до даних на жорсткому диску в сотні разів нижче швидкості доступу до оперативної пам'яті, швидкодія комп'ютера в таких випадках сильно падає, на системному блоці постійно горить індикатор зайнятості жорсткого диску і чути характерний тріск його напруженої роботи.

Все, викладене вище, стосується модулів ОЗУ для звичайних («класичних») комп'ютерів. Якщо йдеться про ноутбуки, ситуація виглядає дещо інакше. Принципи роботи ОЗУ портативного комп'ютера, звичайно, такі ж, але є специфіка.

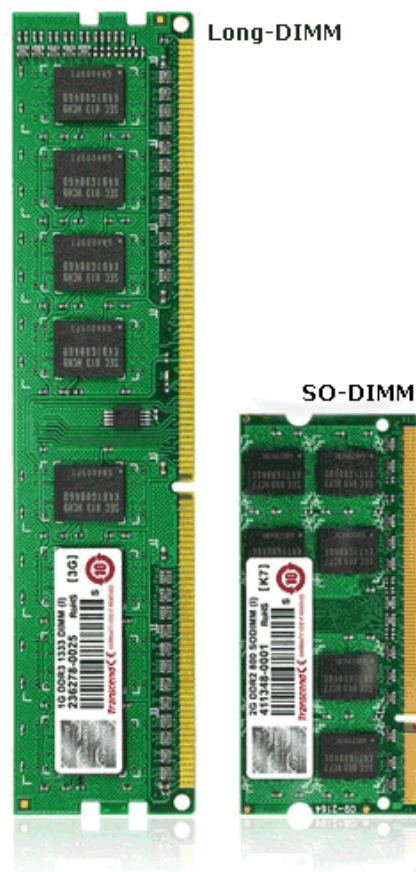
По-перше, розміри модулів ОЗУ для ноутбуків інші. У них встановлюється оперативна пам'ять у форм-факторі **SO-DIMM** (англ. small outline dual in-line memory module). У стаціонарному комп'ютері використовуються модулі формату **Long-DIMM**. Тому пам'ять для ноутбуків і звичайних комп'ютерів – не взаємозамінні речі! У форм-факторі SO-DIMM є такі ж типи пам'яті (DDR, DDRII, DDRIII), але підходять вони лише для ноутбуків і деяких інших пристроїв.

По-друге, на відміну від стаціонарного комп'ютера, замінити або доставити додатковий модуль ОЗУ в ноутбук досить складно.

Якщо на комп'ютері буде використовуватися 32-бітна операційна система, ставити на цю машину більше 4 ГБ оперативної пам'яті особливого сенсу немає, оскільки система буде "бачити" тільки 3 ГБ ОЗУ і ще 25% від того, що залишилося (тобто, якщо поставити 4 ГБ, буде використовуватися тільки 3,25 ГБ). Для використання ОЗУ більшого обсягу необхідна 64-бітна операційна система.

Більшість материнських плат підтримує двоканальний (іноді навіть трьохканальний) режим роботи з оперативною пам'яттю, що забезпечує до неї більш швидкий доступ процесора. Але для цього необхідно, щоб в слотах обох каналів ОЗУ (роз'єми на материнській платі) було встановлено однакову кількість модулів однакових об'ємів. Вкрай бажано, щоб частота шин і таймінги цих модулів також збігалися. Тобто замість 1 модуля пам'яті об'ємом 4 ГБ доцільніше придбати 2 модуля по 2ГБ (по одному на кожен канал).

Постійний запам'ятовуючий пристрій (ПЗП) виконується у вигляді мікросхем, в які необхідна інформація записується на заводі-виробнику тільки один раз і її можна тільки читати. Такий вид інформації називають ROM (Read Only Memory – пам'ять тільки для читання). Хоча доступ до ПЗП повільніший, ніж до ОЗП, основна його перевага у тому, що інформація не щезає при



вимиканні живлення. Тому в ПЗП зберігають програми початкового завантаження ПК, тестування, виконання операцій з пристроями введення-виведення, а вміст ПЗП називають BIOS (Basic Input-Output System – базова система введення-виведення). Завдяки BIOS ПК при вмиканні “оживає” і готує усю систему до роботи з людиною.

Крім ОЗП і ПЗП у ПК є пам’ять для збереження параметрів конфігурації ПК – CMOS-пам’ять (Complementary Metal-Oxide Semiconductor) з низьким енергоспоживанням (рис. 2). Вміст CMOS-пам’яті не змінюється при вимиканні електроживлення ПК, так як для її живлення використовують спеціальний акумулятор.



Рисунок 2 – CMOS та BIOS.

Практична частина

Завдання 1. Заповніть таблицю, в якій необхідно вказати відмінності оперативної та постійної пам’яті.

Відмінності оперативної та постійної пам’яті		
	Оперативний ЗП	Постійний ЗП
1		
2		
...		
п		

Завдання 2. Проведіть ретроспективний аналіз розвитку засобів оперативного зберігання інформації. Для цього:

1. Визначте основні ознаки призначення засобів оперативного зберігання інформації.
2. Опишіть ознаки структури засобів оперативного зберігання інформації кожного покоління.

3. Визначте принцип дії засобів оперативного зберігання інформації. Надайте характеристику техпроцесу виготовлення засобів оперативного зберігання інформації.

4. Визначте залежності між принципом дії засобів оперативного зберігання інформації та їх структурою, призначенням і параметрами.

5. Визначте основні технологічні, функціональні, економічні та ергономічні характеристики засобів оперативного зберігання інформації. Отримані дані запишіть у вигляді таблиці.

Завдання 3. Визначте найперспективніші технології виробництва засобів оперативного зберігання інформації. Для цього:

1. За кількісними показниками функціональних критеріїв засобів оперативного зберігання інформації побудуйте S-криву їх розвитку у часі.

2. Визначте за отриманою S-кривою сучасний етап еволюції засобів оперативного зберігання інформації.

3. На основі ретроспективного аналізу та визначених у завданні 2 закономірностей визначте принцип дії, який може бути покладений в основу розробки нових поколінь засобів оперативного зберігання інформації для покращення їх характеристик.

Завдання 4. Визначте, якими параметрами має володіти модуль оперативної пам'яті, за умови його встановлення до геймерського комп'ютера. Запишіть у таблицю усі необхідні технічні характеристики модуля оперативної пам'яті.

Таблиця характеристики модуля оперативної пам'яті

Параметр модуля пам'яті	Значення параметру модуля пам'яті
Фірма виробник	
Тип оперативної пам'яті	
Пропускна здатність модуля (одноканальний режим)	
Тактова частота шини пам'яті	
Ефективна частота обміну даними	
Об'єм модуля пам'яті	

Контрольні питання

1. Вкажіть призначення внутрішніх запам'ятовуючих пристроїв.
2. Яка інформація зберігається в енергонезалежній пам'яті?
3. Що таке оперативна пам'ять ПК?
4. Розкрийте принцип функціонування динамічної оперативної пам'яті.
5. Чому в сучасному персональному комп'ютері використовується і динамічна, і статична оперативна пам'ять?

6. В яких пристроях персонального комп'ютера використовується статична оперативна пам'ять? Як по іншому називають статичну оперативну пам'ять?

7. Що називається ключем модуля ПЗП?

8. Для чого потрібна CMOS-пам'ять?

9. Охарактеризуйте основні параметри оперативної пам'яті.

ЛАБОРАТОРНА РОБОТА № 4

ТЕМА: Склад та тестування НЖМД..

Мета: Отримати досвід роботи і тестування НЖМД.

Теоретична частина

Жорсткий диск (накопичувач на жорстких магнітних дисках (НЖМД), "вінчестер", англ. – hard disk drive (HDD) – пристрій для зберігання інформації, в якому використовується принцип магнітного запису. Всередині жорсткого диску запис даних здійснюється на жорсткі пластини, виготовлені з легкометалевого сплаву або скла, вкриті шаром спеціального магнітного матеріалу (найчастіше – двоокисом хрому). Залежно від конструкції, в *HDD* можуть використовуватися одна або кілька таких пластин, що швидко обертаються на одній осі. За рахунок обертання створюється своєрідний підпір повітря, завдяки якому зчитувальні головки не торкаються поверхні пластин, хоч і знаходяться дуже близько від них (всього кілька нанометрів). Це гарантує надійність запису та зчитування даних. При зупинці пластин головки переміщуються за межі їх поверхні, тому механічний контакт між головками та пластинами практично виключений. Така конструкція забезпечує довговічність запам'ятовуючих пристроїв цього типу.

Крім пластин, до складу HDD входить накопичувач, привод і блок електроніки.

Завдяки високій надійності роботи і відносно невисокій вартості, жорсткі диски є найпоширенішим пристроєм зберігання інформації.



Рисунок 1 – Склад НЖМД.

SSD (solid state-drive) або твердотільний накопичувач – запам'ятовуючий пристрій відносно нового типу, який працює на основі використання мікросхем пам'яті і на відміну від HDD не містить рухомих частин.

Цей тип пристроїв порівняно з HDD має ряд переваг: відсутність будь-якої вібрації і шуму, низьке енергоспоживання, більш висока швидкість роботи при невеликих розмірах, стійкість до температурних коливань і механічного впливу й ін.

Найбільшими *недоліками SSD* є їх висока вартість і швидкість зношування (зазвичай, близько 10 тис. циклів перезапису, у більш дорогих виробках - до 100 тис.). Останнє обов'язково повинне враховуватися при їх експлуатації. Не рекомендується проводити дефрагментацію таких носіїв (це ніяк не прискорить пошук інформації), розміщувати на них [файл підкачки](#), а також здійснювати інші дії, пов'язані з їх "невиправданим" використанням. Серед операційних систем сімейства Windows тільки Windows 7 та вище враховує ці особливості. При використанні більш ранніх версій ОС Windows термін служби **SSD** скорочується.

HDDScan це утиліта для тестування накопичувачів інформації (HDD, SSD, RAID, Flash). Програма призначена для діагностики накопичувачів інформації на наявність BAD-блоків, перегляду S.M.A.R.T атрибутів накопичувача, зміни спеціальних налаштувань, таких як: управління живленням, старт / стоп шпинделя, регулювання акустичного режиму та ін.

Можливості

Підтримувані типи накопичувачів:

- HDD з інтерфейсом ATA/SATA
- HDD з інтерфейсом SCSI
- HDD з інтерфейсом USB
- HDD з інтерфейсом FireWire або IEEE 1394
- RAID масиви з ATA / SATA / SCSI інтерфейсом (тільки тести)
- Flash накопичувачі з інтерфейсом USB (тільки тести)
- SSD з інтерфейсом ATA / SATA

Тести накопичувачів:

- Тест в режимі лінійної верифікації
- Тест в режимі лінійного читання
- Тест в режимі лінійного запису
- Тест в режимі читання Butterfly (штучний тест випадкового читання)

S.M.A.R.T.:

- Читання і аналіз S.M.A.R.T. параметрів з дисків з інтерфейсом ATA/SATA/USB/FireWire
- Читання і аналіз таблиць логів з дисків з інтерфейсом SCSI
- Запуск S.M.A.R.T. тестів на накопичувачах з інтерфейсом ATA/SATA/USB/FireWire
- Монітор температури на накопичувачах з інтерфейсом ATA / SATA / USB / FireWire / SCSI

Інтерфейс користувача



Основний вид програми при запуску:

Елементи управління головного вікна:

- Select Drive – список, що випадає який містить всі підтримувані накопичувачі в системі. Виводиться модель накопичувача і серійний номер. Поруч знаходиться іконка, яка визначає можливий тип накопичувача.
- Кнопка S.M.A.R.T. – дозволяє отримати звіт про стан драйву зробленому на основі атрибутів S.M.A.R.T.
- Кнопка New Task (в центрі) – після натискання на цю кнопку викликається меню з основними завданнями для програми.
- Елемент меню Surface Tests – після натискання на цей елемент викликається вікно з вибором тестів накопичувача.
- Елемент меню S.M.A.R.T. – натискання на цей елемент аналогічно натискання кнопки S.M.A.R.T.
- Елемент меню S.M.A.R.T. Offline tests – при активації цього елемента викликається підміню S.M.A.R.T. тестів Short, Extended, Conveyance.
- Елемент меню Temperature Monitor – після натискання на цей елемент буде запущена завдання моніторингу температури
- Елемент меню Features – при активації цього елемента викликається підміню додаткових можливостей програми
- Елемент меню Identity Info – при натисканні на цей елемент програма виведе звіт про ідентифікаційної інформації накопичувача
- Елемент меню Skin Selection – при натисканні на цей елемент програма відкриє вікно вибору «скінів»
- Елемент меню Build Command Line – при натисканні на цей елемент програма відкриє вікно побудови командного рядка

Окно вибора тестов:



Елементи управління:

- Поле Start LBA – початковий логічний номер сектора для тестування
- Поле End LBA – кінцевий логічний номер сектора для тестування
- Поле Block Size – розмір блоку в секторах для тестування
- Блок радіокнопок Test – дозволяє вибрати тип тесту: верифікація, читання, стирання, читання в режимі Butterfly.
- Кнопка Add Test – додає тест в чергу завдань

Можливості та обмеження тестів:

Може бути запущений тільки один тест поверхні в один час. Це пов'язано з тим, що автору програми не вдалося поки отримати стабільних якісних результатів при запуску 2-х і більше тестів одночасно (на різних накопичувачах).

Тест в режимі Verify може мати обмеження на розмір блоку в 256, 16384 або 65536 секторів. Це пов'язано з особливостями роботи Windows.

Тест в режимі Verify може неправильно працювати на USB / Flash накопичувачах.

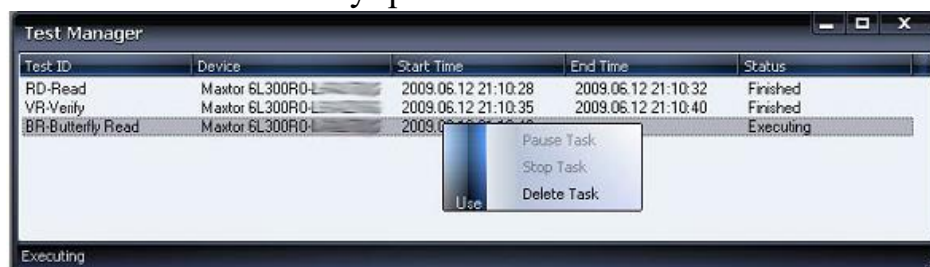
При тестуванні в режимі Verify накопичувач зчитує блок даних у внутрішній буфер і перевіряє їх цілісність, передача даних через інтерфейс не відбувається. Програма заміряє час готовності накопичувача після виконання цієї операції після кожного блоку і виводить результати. Блоки тестуються послідовно - від мінімального до максимального.

При тестуванні в режимі Read накопичувач зчитує дані у внутрішній буфер, після чого дані передаються через інтерфейс і зберігаються в тимчасовому буфері програми. Програма заміряє сумарний час готовності накопичувача і передачі даних після кожного блоку і виводить результати. Блоки тестуються послідовно - від мінімального до максимального.

При тестуванні в режимі Erase програма готує блок даних заповнених спеціальним паттерном з номером сектора і передає дані накопичувача, накопичувач записує отриманий блок (інформація в блоці безповоротно втрачається!). Програма заміряє сумарний час передачі і запису блоку і готовності накопичувача після кожного блоку і виводить результати. Блоки тестуються послідовно - від мінімального до максимального.

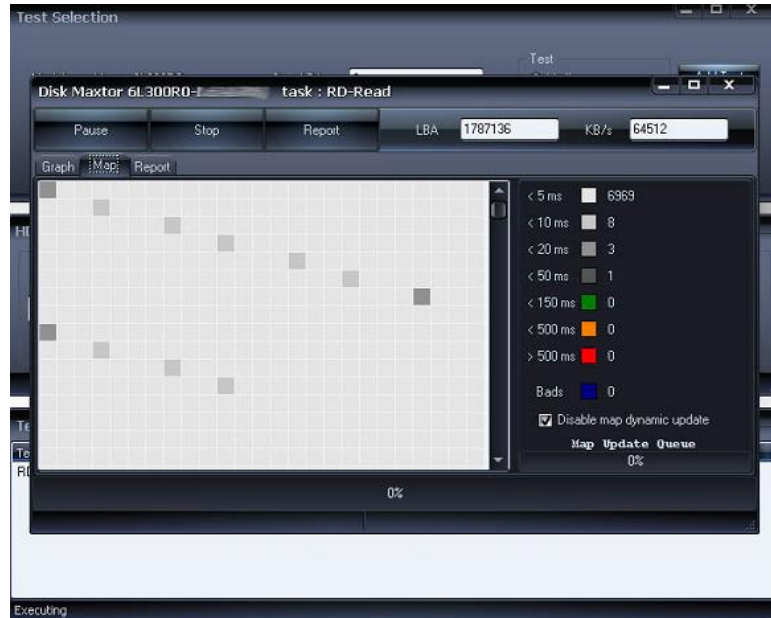
Тестування в режимі Butterfly Read аналогічно тестуванню в режимі Read. Різниця полягає в порядку тестування блоків. Блоки тестуються парами. Перший блок в першій парі буде Блок 0. Другий блок в першій парі буде Блок N, де N це останній блок заданої ділянки. Наступна пара буде Блок1, Блок N-1 і т.і. Завершується тестування в середині заданого ділянки. Цей тест вимірює час позиціонування і час читання накопичувача.

Вікно управління тестами:



Це вікно містить чергу тестів. Сюди потрапляють всі тести, S.M.A.R.T. тести, а також монітор температури, які запускає програма. Менеджер дозволяє видаляти тести з черги. Деякі тести можна ставити на паузу або зупиняти. Подвійний клік на записи в черзі викликає вікно з інформацією про поточне завдання.

Приклад вікна інформації про завдання:



Інформаційне вікно тестів

Вікно містить інформацію про тест, дозволяє ставити тест на паузу або зупиняти, а також генерує звіт.

Вкладка Graph:

Містить інформацію залежності швидкості тестування від номера блоку, представлена у вигляді графа



Вкладка Map:

Містить інформацію залежності часу тестування від номера блоку, представлена у вигляді карти.



За замовчуванням динамічна промальовування карти відключена, це пов'язано з тим, що на слабких машинах промальовування карти займає дуже багато процесорного часу і може впливати на точність тестів. Щоб зменшити вплив промальовування карти на точність тестування, був введений спеціальний буфер Map Update Queue. Потік який тестує накопичувач, складає завдання для промальовування карти в цей буфер. Інший потік забирає завдання і малює карту. Якщо буфер заповниться повністю, то потік тестування накопичувача може працювати неправильно і результати тестування будуть менш точними. Якщо ви бачите що буфер Map Update Queue заповнюється надто швидко - вимкніть динамічну промальовування карти. Ви можете переглядати карту прокручуючи її мишкою, так як результати все одно зберігаються на карті, незалежно від динамічного промальовування.

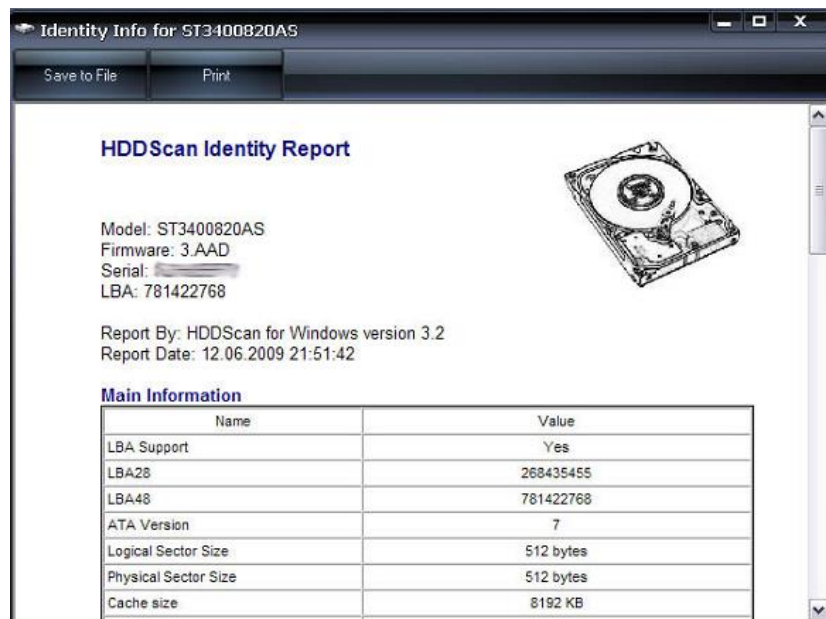
Вкладка Report:

Містить інформацію про тест і всіх блоках, час тестування яких зайняло більш ніж 50 мс.



Ідентифікаційна інформація:

Звіт містить інформацію про основні фізичних і логічних параметрах накопичувача. Звіт можна роздруковувати та зберігати в файл МНТ.



S.M.A.R.T. звіт:

Звіт містить інформацію про продуктивності і «здоров'я» накопичувача у вигляді атрибутів. Якщо на думку програми атрибут в нормі, то поруч з ним стоїть іконка зеленого кольору. Жовтим позначаються атрибути на які слід звернути увагу особливо, як правило вони вказують на будь-яку несправність накопичувача. Червоним позначаються атрибути знаходяться за межами норми. Звіти можна роздруковувати або зберігати в файл типу МНТ.

S.M.A.R.T. attributes for ST3400820AS

Save to File Print

Model: ST3400820AS
Firmware: 3.AAD
Serial:
LBA: 781422768

Report By: HDDScan for Windows version 3.2
Report Date: 12.06.2009 22:00:26

Num	Attribute Name	Value	Worst	Raw(hex)	Threshold
001	Raw Read Error Rate	116	099	000005EB-0250	006
003	Spin Up Time	093	093	00000000-0000	000
004	Start/Stop Count	100	100	00000000-02C0	020
005	Reallocation Sector Count	100	100	00000000-0000	036
007	Seek Error Rate	076	060	000002B8-0DB4	030
009	PowerOn Hours Count	098	098	00000000-070A	000
010	Spin Retry Count	100	100	00000000-0000	097
012	Device Power Cycle Count	100	100	00000000-02CD	020
187	Reported Uncorrectable Error	100	100	00000000-0000	000
189	High Fly Writes	100	100	00000000-0000	000
190	Airflow Temperature	061	047	39 C	045
190	Airflow Temperature Minimum	061	047	28 C	045
190	Airflow Temperature Maximum	061	047	39 C	045

Монітор температури:

Дозволяє оцінювати температуру накопичувача. Інформація виводиться в панель завдань, а також в спеціальне вікно інформації про тест. Рисунок містить свідчення для двох накопичувачів.



Для ATA/SATA/USB/FireWire накопичувачів вікно інформації містить 2 значення. В панель задач виводиться друге значення. Перше значення береться з атрибуту Airflow Temperature, друге значення береться з атрибуту HDA Temperature.



Для SCSI накопичувачів вікно інформації містить 2 значення. В панель задач виводиться друге значення. Перше значення містить максимально допустиму температуру для накопичувача, друге показує поточну температуру.

S.M.A.R.T. тести:

Програма дозволяє запускати три типи S.M.A.R.T. тестів

1. Short test – триває зазвичай 1-2 хвилини. Перевіряє основні вузли накопичувача, а також сканує невелику ділянку поверхні накопичувача і сектора знаходяться в Pending-List (сектора які можуть містити помилки читання). Тест рекомендується для швидкої оцінки стостояння накопичувача.

2. Extended test – триває зазвичай від 0.5 до 2 годин. Перевіряє основні вузли накопичувача, а також повністю сканує поверхню накопичувача.

3. Conveyance test – триває зазвичай кілька хвилин. Перевіряє вузли і логи накопичувача які можуть вказувати на неправильне зберігання або перевезення накопичувача.



Додаткові можливості:

Для ATA/SATA/USB/FireWire накопичувачів програма дозволяє змінювати деякі параметри.

1. ААМ – функція управляє шумом накопичувача. Включення цієї функції дозволяє зменшити шум накопичувача за рахунок більш плавного позиціонування головок. При цьому накопичувач трохи втрачає в продуктивності при випадковому доступі.

2. АРМ – функція дозволяє економити харчування накопичувача за рахунок тимчасового зниження швидкості обертання (або повної зупинки) шпинделя накопичувача в момент простою.

3. PM – функція дозволяє налаштувати таймер зупинки шпинделя на певний час. При досягненні цього час шпиндель буде зупинений за умови що накопичувач знаходиться в режимі простою. Звернення до накопичувача будь-якою програмою викликає примусове розкручування шпинделя і скидання таймера на нуль.

4. Disable Seagate PM – спеціальна команда яка може вимкнути таймер зупинки шпинделя на деяких Seagate-ах, додана на прохання користувачів, знайти на будь драйвах вона працює мені не вдалося

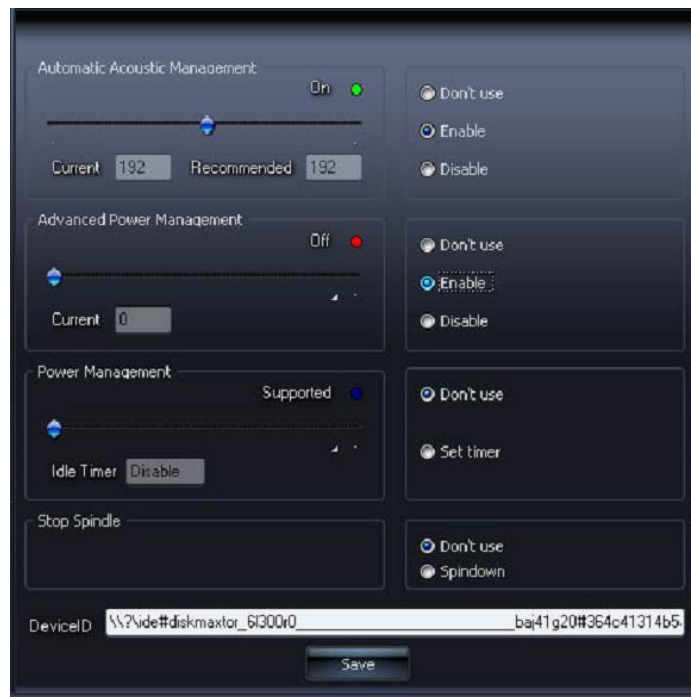
5. Програма також дозволяє зупинити або запускати шпиндель накопичувача примусово. Звернення до накопичувача будь-якою програмою викликає примусове розкручування шпинделя.



Для SCSI накопичувачів програма дозволяє переглядати дефект-листи і запускати / зупиняти шпиндель.



Програма може будувати командний рядок для управління деякими параметрами накопичувача і зберігати цей рядок в bat або cmd файл. При запуску такого файлу програма викликається в фоновому режимі, змінює параметри накопичувача відповідно до заданих і автоматично закривається.



Вікно побудови командного рядка

Практична частина

Завдання 1. Виконати повне тестування НЖМД.

Завдання 2. Визначити основні характеристики НЖМД.

Завдання 3. Проаналізувати отримані результати.

ЗМІСТ ЗВІТУ

Звіт повинен містити найменування роботи, мету, виконані завдання, записи досліджень і вимірів, висновки по виконаній роботі.

ЛАБОРАТОРНА РОБОТА №5

Тема: Накопичувачі на оптичних дисках.

Мета: Ознайомитись з призначенням, конструкцією та характеристиками накопичувачів на оптичних дисках персонального комп'ютера.

Теоретична частина

Починаючи з 1995 року в базову конфігурацію персонального комп'ютера замість дисководів на 5,25 дюймів почали включати дисковод CD-ROM. Аббревіатура CD-ROM (Compact Disk Read Only Memory) перекладається як постійний запам'ятовуючий пристрій на основі компакт-дисків. Принцип дії цього пристрою полягає у зчитуванні цифрових даних за допомогою лазерного променя, що відбивається від поверхні диска. В якості носія інформації використовується звичайний компакт-диск CD. Цифровий запис на компакт-диск відрізняється від запису на магнітні диски високою щільністю, тому стандартний CD має ємність порядку 650-700 Мбайт. Такі великі об'єми характерні для мультимедійної інформації (графіка, музика, відео), тому дисководи CD-ROM відносяться до апаратних засобів мультимедіа. Крім мультимедійних видань (електронні книги, енциклопедії, музикальні альбоми, відеофільми, комп'ютерні ігри) на компакт-дисках розповсюджується також різноманітне системне та прикладне програмне забезпечення великих обсягів (операційні системи, офісні пакети, системи програмування і т.д.).

Компакт-диски виготовляють із прозорого пластику діаметром 120 мм і товщиною 1,2 мм. На пластикову поверхню напилюється шар алюмінію або золота. В умовах масового виробництва запис інформації на диск відбувається шляхом витиснення на поверхні доріжки, у вигляді ряду заглиблень. Такий підхід забезпечує двійковий запис інформації. Заглиблення (pit – піт), поверхня (land – ленд). Логічний нуль може бути представлений як пітом, так і лендом. Логічна одиниця кодується переходом між пітом та лендом. Від центру до краю компакт-диску нанесена єдина доріжка у вигляді спіралі шириною 4 мкм із кроком 1,4 мкм. Поверхня диска розбита на три ділянки. Початкова (Lead-In) розташована в центрі диска і зчитується першою. В ній записано вміст диска, таблиця адрес всіх записів, мітка диска й інша службова інформація. Середня ділянка містить основну інформацію і займає більшу частину диска. Кінцева ділянка (Lead-Out) містить мітку кінця диску. Для штампування існує спеціальна матриця-прототип (мастер-диск) майбутнього диска, яка витискує доріжки на поверхні. Після штампування, на поверхню диска наносять захисну плівку з прозорого лаку.

Накопичувач CD-ROM містить:

- електродвигун, що обертає диск;

- оптичну систему, яка складається з лазерного випромінювача, оптичних лінз та датчиків і призначена для зчитування інформації з поверхні диска;
- мікропроцесор, що керує механікою привода, оптичною системою і декодує прочитану інформацію у двійковий код.

Компакт-диск розкручується електродвигуном. На поверхні диска за допомогою привода оптичної системи фокусується промінь із лазерного випромінювача. Промінь відбивається від поверхні диска і скрізь призму подається на датчик. Світловий потік перетворюється в електричний сигнал, який поступає у мікропроцесор, де він аналізується й перетворюється у двійковий код.

Основними характеристиками CD-ROM є:

- **швидкість передачі даних** – вимірюється в кратних долях швидкості програвача аудіо компакт-дисків (150 Кбайт/сек) і характеризує максимальну швидкість з якою накопичувач пересилає дані в оперативну пам'ять комп'ютера, наприклад, 2-швидкісний CD-ROM (2х CD-ROM) буде зчитувати дані зі швидкістю 300 Кбайт/сек., 50-швидкісний (50х) – 7500 Кбайт/сек.;
- **час доступу** – час, потрібний для пошуку інформації на диску, вимірюється у мілісекундах.

Основний недолік стандартних CD-ROM – неможливість записування даних, але існують пристрої однократного записування CD-R та багаторазового записування CD-RW.

Накопичувач CD-R (CD-Recordable). Зовні схожі на накопичувачі CD-ROM і сумісні з ними за розмірами диска та форматами запису. Дають змогу виконати одноразовий запис і необмежену кількість зчитувань. Запис даних здійснюється за допомогою спеціального програмного забезпечення.

Накопичувач CD-RW (CD-ReWritable). Використовуються для багаторазового запису даних, причому можна як просто дописати нову інформацію на вільний простір, так і повністю перезаписати диск новою інформацією (попередні дані знищуються). Як і у випадку з накопичувачами CD-R, для запису даних необхідно встановити в системі спеціальні програми, причому формат запису сумісний зі звичайним CD-ROM.

Накопичувач DVD (Digital Video Disk)

Пристрій для читання цифрових відеозаписів. Зовні DVD-диск схожий на звичайний CD-ROM (діаметр – 120 мм, товщина 1,2 мм), однак відрізняється від нього тим, що на одній стороні DVD-диску може бути записано до 4,7 Гбайт, а на обох – до 9,4 Гбайт. У разі використання двошарової схеми запису на одному його боці можна розмістити вже до 8,5 Гбайт інформації, відповідно на обох боках – близько 17 Гбайт. DVD-диски допускають також перезапис інформації.

Оптичні носії нового покоління об'єднує тільки стандартний розмір диска і синьо-фіолетовий (а зовсім не голубий, як вважає абсолютна більшість користувачів) лазер з довжиною хвилі 405 нм. Перехід на більш

короткохвильовий лазер дозволяє значно щільніше розміщувати інформацію на диску, умістивши на один шар до 27 Гбайт даних.

HD-DVD. Мабуть, вирішальним аргументом у прийнятті цього стандарту стала його спадкоємність з існуючими DVD-дисками. Для переходу на випуск нової продукції потрібна мінімальна модернізація вже існуючих ліній. Навіть мікросхеми в приводах можна залишити колишні: використовуються ті ж самі алгоритми читання і корекції даних, потрібно лише змінити оптичну голівку.



Рисунок 1 – Привід HD-DVD.



Рисунок 2 – Диск HD-DVD.

Оптичні болванки мають колишній розмір і «слоеність». Записуючий шар знаходиться посередині під захистом 0,6 мм пластику.

У порівнянні з DVD, відстань між доріжками і розмір пітів зменшилися майже у два рази, що дозволило збільшити обсяг диска з 4,7 Гбайт до 15 Гбайт. На даний момент вже прийнята специфікація двошарового диска ємністю 30 Гбайт. Крім того, компанії Memory Tech і Toshiba планують створити тришарові диски місткістю 45 Гбайт.

Blu-ray

Цей формат корінним чином відрізняється від DVD. У ньому використовуються абсолютно нові алгоритми зчитування й обробки інформації, що дозволяє домогтися більшої гнучкості фізичної структури накопичувачів. Наприклад, довжина піта може бути 0,138, 0,149 або 0,160 мкм. Записуючий шар на диску розташовується всього лише на відстані 0,1 мм від поверхні. У результаті зменшуються спотворення лазерного променя і час відгуку. Все це дозволяє значно знизити розміри пітів і відстані між доріжками в порівнянні із звичайним DVD. Підсумок: на BD-диску міститься 23,3, 25 або 27 Гбайт даних.

Затверджена специфікація двошарових BD-дисків ємністю 46,6 та 50 Гбайт. Компанія Toshiba випустила чотиришаровий диск ємністю 100 Гбайт. Одне зі слабких місць Blu-ray – дуже маленька відстань між записуючим шаром і поверхнею – 0,1 мм в порівнянні з 0,6 в DVD і HD-DVD. Спочатку єдиним способом захисту від пошкоджень в BD-дисках був картридж. Але потім рядом компаній (наприклад, TDK) були розроблені спеціальні захисні покриття, що протистоять подряпинам і накопиченню бруду, що дозволило позбутися картриджів.



Рисунок 3 – Привід Blu-ray.



Рисунок 4 – Диск Blu-ray.

Найкращою демонстрацією різниці між трьома поколіннями оптичних носіїв є даний малюнок.

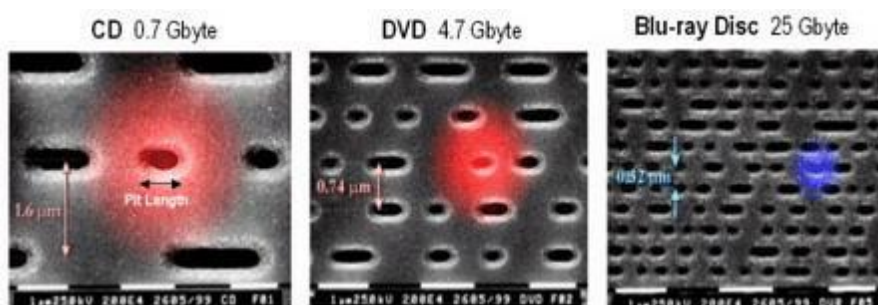


Рисунок 5 – Розміри пітів на CD, DVD і BD.

Вчені з Саутгемптонського університету і Технологічного університету Ейнховена продемонстрували технологію зберігання цифрових даних, яка дозволить створити носії інформації з фантастичними характеристиками: ємність 360 Терабайт на диск, термостійкість до 1000°C і практично необмежений термін служби. Для порівняння, найємніші на сьогоднішній день диски Blu-ray мають максимальний обсяг в 320 Гігабайт і забезпечують збереження даних протягом максимум семи років.

Нова технологія передбачає зберігання інформації в п'яти вимірах: три просторові координати наноструктури плюс її розмір і орієнтація. Кожна наноструктура певним чином змінює напрямок і поляризацію світла, яке проходить через неї, що робить можливим зчитування даних. Запис даних проводиться короткими інтенсивними імпульсами фемтосекундного лазера, що формують в кварцовому склі масив наноструктур.

Диски для такого запису виготовляються з легованого кварцового скла, завдяки чому і отримується така висока термостійкість та довговічність. Дослідники провели експеримент, в якому на такий диск був успішно проведений запис текстового файлу розміром 300 Кбіт в три віддалених один від одного на 5 мкм шари наноструктурованих областей і читання цього файлу.

«Ми розробляємо дуже стабільний і надійний тип пам'яті, який може знайти широке застосування в організаціях з великими архівами. В даний час компанії змушені виконувати резервне копіювання архівів кожні 5-10 років через відносно короткий термін служби жорстких дисків», – заявив керівник групи дослідників Цзін'юй Чжан (Jingyu Zhang).

Цікавим є те, що дослідники назвали нову технологію запису даних “Кристал пам’яті Супермена”, на згадку про вічні кристали пам’яті, які фігурували в серії фантастичних фільмів про Супермена. А професор Пітер Казанський (Peter Kazansky), один з керівників дослідницької групи додав: “Нас хвилює той факт, що ми створили технологію зберігання інформації та документів, яка може пережити людський рід. Може бути так, що саме такі диски послужать в дуже далекому майбутньому єдиними доказами існування в минулому Землі розвиненою людської цивілізації”.

Оптичний привід або оптичний накопичувач – електричний пристрій для зчитування і (залежно від конструкції) запису інформації з оптичних носіїв (наприклад, CD-ROM або DVD-ROM).

Оптичні приводи розрізняються за підтримуваними форматами лазерних дисків, а також можливістю запису на оптичний диск.



Рисунок 6 – Оптичний привід.

Для нормальної роботи привід повинен перебувати в горизонтальному положенні. Існує накопичувач, який може працювати у вертикальному положенні. При цьому диск вставляється в щілину руками, після чого спеціальний механізм утримує його і вводить всередину накопичувача.

Основні складові накопичувача CD ROM:

- Електродвигун, що обертає диск.
- Оптична система, що призначена для зчитування інформації і складається з: лазерного випромінювача, оптичних лінз, датчиків;
- Мікропроцесор, який керує механікою та оптикою, а також перетворює світловий потік у двійковий код.

Компакт-диск розкручується електродвигуном, на поверхні диску фокусується промінь з лазерного випромінювача. Промінь відбивається від поверхні, проходить скрізь лінзи і попадає на датчик. Там світловий потік перетворюється на електричний сигнал, що надходить до мікропроцесора. Мікропроцесор аналізує сигнал і перетворює його у двійковий код.

В оптичному накопичувачі є отвір для аварійного висунення лотка, якщо він не висувається. Для цього потрібно вставити тонкий стрижень, наприклад, випрямлення скріпки, і натиснути на нього.

Основні характеристики приводу:

- тип: внутрішній або зовнішній. Внутрішній привід вставляється в системний блок. Зовнішній має корпус прямокутної форми, підключається до паралельного порту (у старих комп'ютерах), USB (у сучасних) і має провід, що

сполучається з електромережею. Існує також зовнішній варіант для переносних комп'ютерів, що підключається за допомогою роз'єму PCMCIA;

- швидкість передачі даних (Data Transfer Rate, DTR), відповідно вказується як двошвидкісний, чотирьох-, тридцяти двох-і т.д.;

- обсяг буферної пам'яті (Buffer Memory). Кеш-пам'ять – мікросхема оперативної пам'яті, яка розташовується на платі накопичувача.

- середній час між поломками (Mean Time Between Failure, MTBF). Дана характеристика є у багатьох пристроїв, проте не скрізь описується;

- тип інтерфейсу або шини, до якого підключається;

- середній час доступу (Access Time, AT). Він в CD-ROM накопичувачів більше, ніж у жорстких дисків, що визначено принциповими відмінностями в конструкції накопичувача, і розрізняється в десятки разів, причому чим більше кратність, тим менше час доступу. Так, у 4-кратного накопичувача воно приблизно дорівнює 150, а у 32 – 80 мс. Це значення можна довідатися з паспорта пристрої;

- коефіцієнт помилок (Error Time);

- перелік підтримуваних форматів.

Можуть бути також інші параметри, такі, як рівень шумів, вібрації. Крім того, при покупці потрібно подивитися, м'яко Чи рухається лоток і чи міцно він утримується у відкритому вигляді.

Підключається пристрій за допомогою двох кабелів: живлення та інформаційного. Існує три види накопичувачів: підключаються до шини SCSI, до шини IDE або роз'єму SATA.

При роботі з дисками необхідно виконувати **наступні правила**:

- не чіпайте робочу поверхню, інакше на ній можуть залишитися жирові сліди пальців;

- беріть диск за зовнішні краї, можна брати за краї центрального отвору;

- очищення диска проводиться від центру диска до зовнішнього краю м'якою сухою ганчіркою. Не можна використовувати сильні розчинники такі як, ацетон, мийні засоби, антистатичні аерозолі;

- зберігайте диски у спеціальній коробочці або конверті для дисків;

- не згинайте диск;

- не пишіть на робочій поверхні диска;

- при зберіганні диска уникайте попадання на нього сонячних променів, а також сильного нагріву, що може призвести до викривлення диска.

Практична частина

Завдання 1. Заповнити таблицю, в якій вказати основні параметри різновидів оптичних дисків.

Основні параметри оптичних дисків

Параметри дисків	CD-ROM	DVD	HD-DVD	Blu-ray
Діаметр, мм				

Товщина, мм				
Структура				
Довжина хвилі лазера, нм				
Відстань між доріжками, мкм				
Розмір поглиблення (pit), мкм				
Швидкість обертання диска, об/хв				
Ємність, Мбайт				
Швидкість передачі інформації, байт/с				

Завдання 2. Проведіть ретроспективний аналіз розвитку змінних засобів довготривалого зберігання інформації. Для цього:

1. Визначте основні ознаки призначення змінних засобів довготривалого зберігання інформації.
2. Опишіть ознаки структури змінних засобів довготривалого зберігання інформації кожного покоління.
3. Визначте принцип дії змінних засобів довготривалого зберігання інформації.
4. Визначте залежності між принципом дії змінних засобів довготривалого зберігання інформації та їх структурою, призначенням і параметрами.
5. Визначте основні технологічні, функціональні, економічні та ергономічні характеристики змінних засобів довготривалого зберігання інформації. Отримані дані запишіть у вигляді таблиці.

Завдання 3. Визначте найперспективніші технології виробництва засобів змінних засобів довготривалого зберігання інформації. Для цього:

1. За кількісними показниками функціональних критеріїв засобів змінних засобів довготривалого зберігання інформації побудуйте S-криву їх розвитку у часі.
2. Визначте за отриманою S-кривою сучасний етап еволюції змінних засобів довготривалого зберігання інформації.
3. На основі ретроспективного аналізу та визначених у завданні 2 закономірностей визначте принцип дії, який може бути покладений в основу розробки нових поколінь змінних засобів довготривалого зберігання інформації для покращення їх характеристик.

Контрольні питання

1. Які існують різновиди накопичувачів на оптичних дисках?
2. Який матеріал використовують для основи CD дисків?
3. Що позначають поняття «піт» і «ленд»?
4. Яким чином здійснюється запис/затирання на дисках CD-RW?

5. Яким чином відбувається зчитування інформації з компакт-дисків?
6. Яка характеристика накопичувача на оптичному диску є важливою для користувача?
7. Основні відмінності між DVD та CD дисками?
8. Які існують типи DVD дисків?
9. В чому вимірюється швидкість передачі даних в накопичувачах на оптичних носіях?

ЛАБОРАТОРНА РОБОТА №6

Тема: Відеоадаптер персонального комп'ютера.

Мета: *Ознайомитись з призначенням, конструкцією та елементами графічної карти персонального комп'ютера.*

Теоретична частина

Відеокарта (відома також як відеоадаптер, графічна плата, графічний адаптер, графічна карта) – важлива і дуже складна складова частина комп'ютера. Сучасні відеокарти є свого роду спеціалізованими комп'ютерами, що складаються з власного процесора, оперативної пам'яті, BIOS і інших компонентів, які за своєю структурою і організацією взаємодії пристосовані для максимально ефективного вирішення одного завдання – обробки і формування графічних даних, а також їх виведення на монітор.

Мало хто задумується над тим, наскільки складним насправді є процес обробки різних графічних даних з метою отримання кінцевого зображення, що відображається на моніторі. Цей процес вимагає здійснення величезної кількості точних розрахунків (створення вершин, їх збирання в примітиви (трикутники, лінії, крапки і т.д.), створення піксельних блоків, операції освітлення, затінення, текстурування, присвоєння кольору та ін.).

Зазвичай, коли користувач говорить, що його відеокарта "гальмує", мається на увазі саме її нездатність вивести достатню кількість кадрів в секунду. Це явище може спостерігатися не лише в іграх, але і при роботі з об'ємними графічними програмами. Здатність відеокарти обробляти графіку з певною швидкістю залежить як від потужності самої карти, так і від складності оброблюваної графіки.

Сучасна графічна карта складається з наступних частин:

Графічний процесор (графічне ядро GPU (Graphics processing unit – графічний процесорний пристрій) – процесор, що займається розрахунками та формуванням графічної інформації, що виводиться на монітор, є основою відеокарти і по своїй складності практично не поступається центральному процесору комп'ютера, а іноді і перевершує його;

Відеопам'ять – виконує роль своєрідного буфера, в який тимчасово поміщаються зображення, що виводяться на монітор, створюються та постійно змінюються графічним ядром. У цей буфер поміщаються також елементи, необхідні для формування цих зображень;

Відеоконтролер – відповідає за правильне формування і передачу потрібної інформації з відеопам'яті на RAMDAC.

RAMDAC (Random Access Memory Digital-to-Analog Converter) або цифро-аналоговий перетворювач (ЦАП) – пристрій, що здійснює перетворення цифрових результатів роботи відеокарти в аналоговий сигнал, який відображається на моніторі. Можливостями цього пристрою визначається кількість відображуваних кольорів, насиченість картини та ін. Цифрові монітори, проектори та інші пристрої, які підключаються до

цифрових роз'ємів відеокарти, використовують власні цифро-аналогові перетворювачі і від RAMDAC відеокарти не залежать;

Відео-ПЗУ (Video ROM) – мікросхема, що містить в собі базову систему введення-виведення відеокарти, а інакше кажучи її BIOS – сукупність правил і алгоритмів, визначених виробником, за яким складові частини відеокарти працюють і взаємодіють між собою.

Відеокарти бувають двох типів: дискретні і вбудовані(інтегровані). Багато користувачів цікавить, що таке дискретна відеокарта і чим вона відрізняється від інтегрованої. У цій статті ми розповімо про дискретні і інтегровані відеокарти, а також деякі особливості їх використання.

Інтегрована відеокарта – це відеокарта, яка інтегрована(вбудована) в чіпсет материнської плати або в процесор. Тобто, інтегрована відеокарта, є невід'ємною частиною іншого чіпа(процесора або чіпсета). Замінити таку відеокарту без заміни материнської плати або процесора неможливо.

Дискретна відеокарта – це відеокарта, виконана на окремій платі. Дискретні відеокарти оснащені власними графічними процесорами, які виконують обробку зображення. Окрім цього дискретні відеокарти оснащені власною відеопам'яттю. Завдяки цим особливостям дискретні відеокарти можуть демонструвати більш високу продуктивність.

Більшість сучасних процесорів як від Intel, так і від AMD, оснащуються інтегрованою графікою(відеокартою). Але, доступність інтегрованої графіки залежить від чіпсета.

У ноутбуках можлива ситуація коли і дискретна і інтегрована відеокарта доступні користувачеві у будь-який момент часу. При цьому користувач може перемикатися між дискретною і інтегрованою відеокартою програмним способом, навіть без перезавантаження комп'ютера. Це дозволяє включати дискретну графіку тільки тоді, коли це дійсно треба, а в решту часу використати вбудовану графіку, яка споживає значно менше енергії.

Щоб остаточно розібратися з тим, що таке дискретна відеокарта і чим вона відрізняється від інтегрованої, розглянемо основні переваги кожного з варіантів.



Рисунок 1 – Дискретна відеокарта для настільного ПК.

Переваги інтегрованих відеокарт:

Низьке споживання енергії. Ноутбуки з інтегрованою графікою можуть працювати довше від одного заряду акумулятора.

Низький рівень шуму. Інтегрована відеокарта, на відміну від дискретної, не вимагає установки додаткових вентиляторів.

Комп'ютери з інтегрованою відеокартою коштують значно дешевше. Вбудувати графічний прискорювач в процесор значно простіше і дешевше за створення повноцінної дискретної відеокарти.

Переваги дискретних відеокарт:

Дискретну відеокарту можна замінити без заміни інших комплектуючих комп'ютера. Більше того, в настільному комп'ютері заміна відеокарти це настільки проста процедура, що з нею з може впорається будь-який той, що бажає.

Дискретна відеокарта надає користувачеві велику продуктивність. Інтегровані відеокарти розвиваються дуже швидко, і останні моделі демонструють цілком серйозні результати. Але, не дивлячись на це, перевершити повноцінні дискретні відеокарти їм не вдасться ніколи.

У одному комп'ютері можна використати декілька дискретних відеокарт. Завдяки режимам SLI і Crossfire у один комп'ютер можна встановити більше однієї дискретної відеокарти. Що у свою чергу дозволяє ще більше підняти продуктивність графічної частини комп'ютера.

Дискретні відеокарти дозволяють працювати з високими розділеннями і декількома моніторами. Якщо ви хочете використати монітор з високим розділенням або відразу декілька моніторів, то вам доведеться використати дискретну відеокарту. Інтегровані рішення доки не справляються з такими навантаженнями.

Відеокарти з активним і пасивним охолодженням

Дискретні відеокарти можуть оснащуватися активним або пасивним охолодженням. Активним охолодженням називають систему охолодження, в якій використовуються кулери(вентилятори). Як правило, на відеокарті може бути встановлено від одного до трьох кулеров. Така система охолодження дозволяє охолодити навіть найпотужніший графічний процесор, проте вона видає багато шуму. Тому любителям тихих комп'ютерів варто звернути увагу на пасивне охолодження.

Відеокарти з пасивним охолодженням не оснащуються кулерами і не видають ніякого шуму. Проте, без кулерів неможливо охолодити потужні графічні процесори, тому такі рішення не відрізняються високою продуктивністю.

Сучасні відеокарти це дуже потужні і продуктивні пристрої. Вони споживають багато енергії і сильно нагріваються. Тому дуже важливо щоб температура відеокарти не перевищувала критичні значення. Інакше це неминуче приведе до зниження продуктивності і навіть може стати причиною поломки.

Температура відеокарти залежить від графічного процесора, на основі якого вона побудована, а також системи охолодження. Тому залежно від

конкретної моделі значення нормальної температури може коливатися. Наприклад, відеокарти з пасивним охолодженням, а також мобільні версії відеокарт для ноутбуків, гріються значно сильніше за свої аналоги для звичайних настільних ПК.

Проте, можна назвати середні значення температури для типової відеокарти. Нормальна температура відеокарти це:

- у режимі простою до 55 градусів Цельсія;
- під навантаженням до 80 градусів Цельсія;

Якщо ваша температура вашої відеокарти регулярно перевищує ці значення, то у вас серйозні проблеми з перегріванням.

Відеокарти з додатковим живленням і без

Більшість дискретних відеокарт вимагають додаткового живлення. На таких відеокартах розміщується спеціальний роз'єм з шістьма або вісьмома контактами. На деяких найпотужніших рішеннях може бути встановлено відразу два роз'єми для додаткового живлення.

Якісні блоки живлення оснащені спеціальними кабелями для підключення додаткового живлення до відеокарти. Якщо блок живлення не оснащений спеціальними виведеннями для подання додатково живлення, то можна використати спеціальні перехідники. Для малопотужних відеокарт а також відеокарт з пасивним охолодженням додаткове живлення не використовується.

Усі сучасні відеокарти використовують для підключення до комп'ютера шину PCI Express. На материнській платі може бути один або декілька роз'ємів PCI Express. Усі вони розміщуються внизу материнської плати і забарвлені в яскравий колір. Якщо у вас на материнській платі декілька роз'ємів PCI Express те встановлювати відеокарту необхідно у верхній. Крім випадків, коли ви встановлюєте декілька відеокарт.

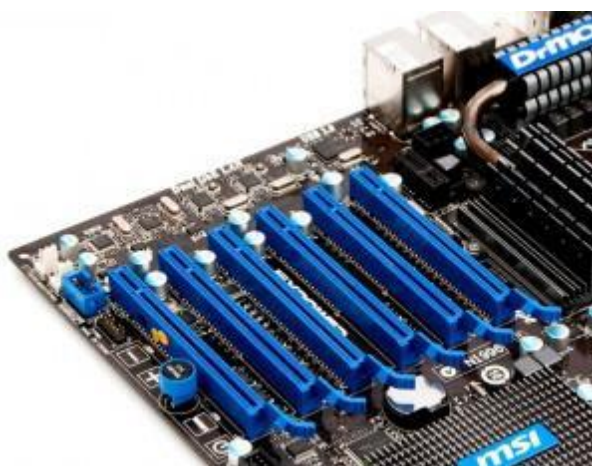


Рисунок 2 – Роз'єми PCI Express на материнській платі.

Основні показники відеокарти, що впливають на її продуктивність:
Продуктивність відеопам'яті. Як свідчить практика, відеопам'ять дуже часто є слабким місцем графічних плат. І справа в першу чергу не в її обсязі, а в пропускної здатності, що визначає швидкість доступу до даних, які в ній зберігаються. Пропускна здатність залежить від двох показників – частоти

(швидкість тактових коливань) і ширини (бітності) шини пам'яті – кількості даних, що передаються за один такт.

Тип відеопам'яті (GDDR2, GDDR3, GDDR4, GDDR5 і ін.) вказує на те, до якого покоління належить пам'ять графічної карти. Кожне наступне покоління є досконаліше попереднього і забезпечує більш високу частоту роботи.

Об'єм відеопам'яті також впливає на продуктивність графічної плати, але тільки до певної межі (коли він є слабким місцем). Набагато вигідніше придбати карту з пам'яттю GDDR3 – 256 біт і об'ємом 512 МБ ніж з пам'яттю GDDR3 – 128 біт і об'ємом 1 ГБ. Насправді графічній платі з низькою пропускнуою здатністю обсяг пам'яті 1 ГБ навряд чи коли-небудь знадобиться. Такі карти орієнтовані не на досягнення максимальної продуктивності. Вони є більше продуктом маркетингових хитрощів виробників, розрахованих на недосвідчених покупців, що оцінюють графічні прискорювачі виключно за розміром пам'яті.

Тому, вибираючи відеокарту, потрібно оцінювати збалансованість співвідношення частоти, бітності і об'єму відеопам'яті. Ці показники зазвичай вказуються в каталогах і цінниках магазинів.

Характеристики графічного ядра. Тактова частота графічного процесора є важливою, але не найголовнішою його характеристикою. Графічне ядро з порівняно невисокою частотою нерідко виявляється дуже продуктивним. Все залежить від архітектури графічного ядра, кількості і якості уніфікованих шейдерних блоків, що входять до його складу (чим більше, тим краще), і інших елементів, якими визначається піксельна і текстурна швидкості заповнення (філрейт, fill rate) відеокарти (чим вище, тим краще).

Ці показники рідко вказуються на цінниках і в каталогах. Тому перед вибором відеокарти з декількох можливих варіантів, бажано на офіційному сайті їх виробників (або на інших спеціалізованих сайтах) поцікавитися реальним станом речей і вибрати варіант з найвищими показниками.

На практиці, чим новіша лінійка відеокарт, до якої належить графічний прискорювач, тим, як правило, він потужніший. Один з непрямих ознак невисокої продуктивності відеокарти – відсутність роз'єму для підключення додаткового живлення безпосередньо від блоку живлення. Шина PCI-E материнської плати, до якої приєднується графічна плата, не може забезпечити достатнє живлення.

Система охолодження – елемент, від якого багато в чому залежить комфорт використання графічного прискорювача. При виборі краще віддати перевагу виробам, виконаним із застосуванням вакуумних термотрубок (їх видно при візуальному огляді). Такі системи насправді виявляються більш ефективними і створюють набагато менше шуму. Крім того, ефективне охолодження надає можливість краще "розігнати" відеокарту, домігшись при необхідності більш високих показників її продуктивності.

Практична частина

Завдання 1. Заповнити таблицю, в якій вказати переваги та недоліки дискретних та інтегрованих відеокарт ПК.

Завдання 2. Визначте, якими параметрами має володіти відеоадаптер, за умови його встановлення до геймерського комп'ютера, комп'ютера для майнінгу, офісного комп'ютера. Запишіть у таблицю 11.3 усі необхідні технічні характеристики відеоадаптера.

Таблиця характеристики дискретного відеоадаптера

Характеристика	Значення характеристики
Фірма виробник	
Країна виробник	
Модель графічного процесора	
Інтерфейс(тип роз'єму для підключення до материнської плати)	
Тип відеопам'яті	
Об'єм відеопам'яті	
Розрядність шини відеопам'яті	
Зовнішні роз'єми	
Наявність додаткового роз'єму для живлення	
Тип системи охолодження	

Контрольні питання

1. Які функції виконує відеоадаптер ПК?
2. З яких частин складається відеокарта? Коротко охарактеризуйте їх.
3. Що впливає на продуктивність відеокарти?
4. Для чого призначений дискретний відеоадаптер?
5. З яких частин складається дискретна графічна карта?
6. Які технічні показники впливають на продуктивність дискретного відеоадаптера?

ЛАБОРАТОРНА РОБОТА № 7

ТЕМА: Діагностика, настройка і тестування відеосистеми ПК.

МЕТА: Отримати навички роботи та налаштування відеосистем ПК.

Теоретична частина

Принцип дії рідкокристалічної панелі.

"Серцем" будь-якого рідкокристалічного монітора є LCD-матриця (Liquid Crystall Display). РК-панель це складна багатошарова структура (рис.

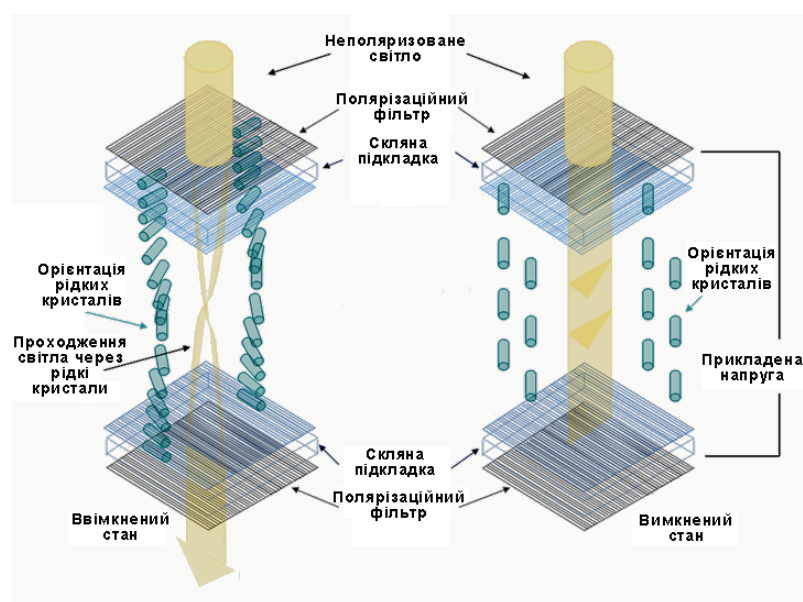


Рисунок 1 – Принцип дії РК панелі.

1).

Принцип дії будь-якого рідкокристалічного екрана заснований на властивості рідких кристалів змінювати (повертати) площину поляризації світла, що проходить через них, пропорційно прикладеної до них напрузі. Якщо на шляху поляризованого світла, що пройшло через рідкі кристали, поставити поляризаційний світлофільтр (поляризатор), то, змінюючи величину прикладеної до рідких кристалів напруги, можна управляти кількістю світла, що пропускається поляризаційним світлофільтром. Якщо кут між площинами поляризації світла, що пройшов крізь рідкі кристали, і світлофільтру становить 0 градусів, то світло буде проходити крізь поляризатор без втрат (максимальна прозорість), якщо 90 градусів, то світлофільтр буде пропускати мінімальна кількість світла (мінімальна прозорість) (див. рис. 1).

Таким чином, використовуючи рідкі кристали, можна виготовляти оптичні елементи із змінним ступенем прозорості. При цьому рівень світлопропускання такого елемента залежить від прикладеної до нього напруги. Будь-який РК-екран у моніторах комп'ютера, ноутбука, планшета або телевізора містить від декількох сотень тисяч до декількох мільйонів таких

комірок, розміром із частку міліметра. Вони об'єднані в LCD-матрицю і з їх допомогою ми можемо формувати зображення на поверхні рідкокристалічного екрана.

Рідкі кристали були відкриті ще в кінці XIX століття. Однак перші пристрої відображення на їх основі з'явилися тільки в кінці 60-х років XX століття. Перші спроби застосувати LCD-екрани в комп'ютерах були зроблені в вісімдесятих роках минулого століття. Перші рідкокристалічні монітори були монохромними і сильно поступалися за якістю зображення дисплеям на електронно-променевих трубках (ЕПТ). Головними недоліками LCD-моніторів перших поколінь були:

- – низька швидкодія і інерційність зображення;
- – «хвости» і «тіні» на зображенні від елементів картинки;
- – погана роздільна здатність зображення;
- – чорно-біле або кольорове зображення з низькою колірною глибиною.

Однак, прогрес не стояв на місці і, з часом, були розроблені нові матеріали і технології у виготовленні рідкокристалічних моніторів. Досягнення в технологіях мікроелектроніки та розробка нових речовин з властивостями рідких кристалів дозволило істотно поліпшити характеристики РК-моніторів.

Побудова і принцип дії TFT LCD матриці.

Одними з головних досягнень стал винахід технології LCD TFT-матриці – рідкокристалічної матриці з тонкоплівковими транзисторами (Thin Film Transistors). У TFT-моніторів кардинально зросла швидкодія пікселів, зросла колірна глибина зображення і вдалося позбутися «хвостів» і «тіней».

Структура панелі, виготовленої з TFT технології, наведена на рис. 2

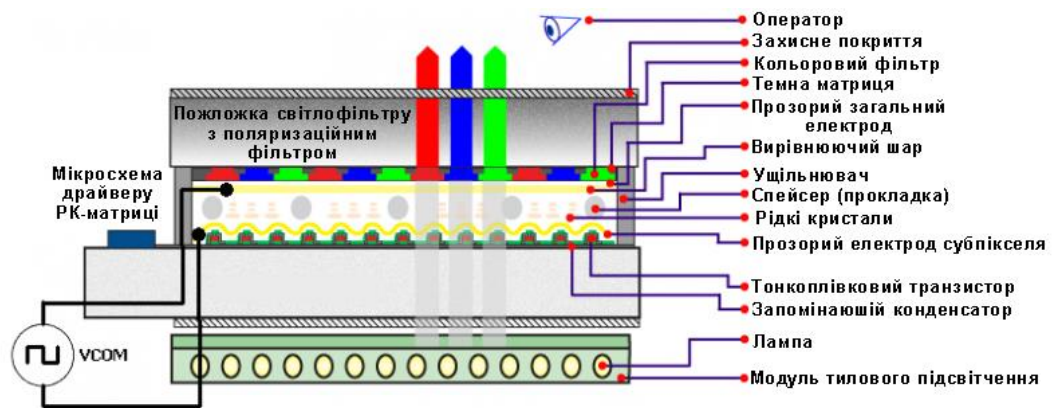


Рисунок 2 – Структура LCD TFT – матриці.

Повнокольорове зображення на РК-матриці формується з окремих точок (пікселів), кожна з яких складається зазвичай з трьох елементів (субпікселів), що відповідають за яскравість кожної з основних складових кольору – зазвичай червоного (R), зеленого (G) і синього (B) – RGB. Відеосистема монітора безперервно сканує всі субпікселі матриці, записуючи в запам'ятовуючі конденсатори рівень заряду, який пропорційний яскравості кожного субпікселя. Тонкоплівкові транзистори (Thin Film Transistor (TFT) –

власне, тому так і називається TFT-матриця) підключають запам'ятовуючи конденсатори до шини з даними на момент запису інформації і перемикають запам'ятовуючий конденсатор в режим збереження заряду на весь інший час.

Напруга, яка зберігається у пам'яті великої кількості конденсаторів TFT-матриці, діє на рідкі кристали даного субпікселя, повертаючи площину поляризації світла від тилового підсвічування, що проходить через них, на кут, пропорційний цієї напрузі. Пройшовши через комірку з рідкими кристалами, світло потрапляє на матричний світлофільтр, на якому для кожного субпікселя сформований свій світлофільтр одного з основних кольорів (RGB). Далі, сформований світловий потік основних кольорів надходить на зовнішній поляризаційний фільтр, коефіцієнт пропускання світла якого залежить від кута поляризації падаючої на нього світлової хвилі. Поляризаційний світлофільтр прозорий для тих світлових хвиль, площина поляризації яких паралельна його власній площині поляризації. Із зростанням цього кута, поляризаційний фільтр починає пропускати все менше світла, аж до максимального послаблення при куті 90 градусів. В ідеалі, поляризаційний фільтр не повинен пропускати світло, яке поляризовано ортогонально його власній площині поляризації, але в реальному житті, все-таки невелика частина світла проходить. Тому всім ЖК-дисплеїв властива недостатня глибина чорного кольору, яка особливо яскраво проявляється при високих рівнях яскравості тилового підсвічування.

В результаті, в LCD-дисплеї світловий потік від одних субпікселей проходить через поляризаційний світлофільтр без втрат, від інших субпікселей – послаблюється на певну величину, а від якоїсь частини субпікселей практично повністю поглинається. Таким чином, регулюючи рівень кожного основного кольору в окремих субпікселях, можна отримати з них піксель будь-якого колірної відтінку. А з безлічі кольорових пікселів скласти повноекранне кольорове зображення.

ПК-монітор дозволив зробити серйозний прорив в комп'ютерній техніці, зробивши її доступною великій кількості людей. Більш того, без LCD-екрану неможливо було б створити портативні комп'ютери типу ноутбуків і нетбуків, планшети і мобільні телефони.

Колориметр. Для надійної обробки кольорів монітор повинен мати спосіб «бачити» зображення на екрані, що дозволить йому виконати самонастройку внутрішніх схем для досягнення правильних колірних значень. Саме для цього призначений колориметр, який підключається до монітора. Колориметр – це пристрій, що вимірює колір з точки зору значень червоного, зеленого і синього кольорів. Колориметр монітора, що калібрується – з його трикомпонентними (червоним, зеленим і синім) сенсорами – зазвичай підключається за допомогою присоски до ділянки монітора, де виводиться еталонне зображення. Програмний пакет для калібрування послідовно відображає в цьому місці різні відтінки сірого, червоного, зеленого і синього кольорів. Колориметр вимірює кольору і передає результати по кабелю в комп'ютер.

Програма калібрування приймає значення RGB від колориметра, порівнює їх з еталонними значеннями і посилає спеціальні команди на керуючі ланцюги монітора. Ці команди налаштовують відеопідсилювачі таким чином, щоб домогтися якомога більш точного збігу вхідних і вихідних значень кольорів. Після того як все налаштовано, нові параметри фіксуються, а програма створює відкоригований профіль ICC, або в іншому форматі, який використовується в одній з систем управління кольором. Щоб переконатися, що кольори не «пішли», або виправити з'явилися зміни кольору, процедура калібрування зазвичай виконується кожні два тижні.

Слід пам'ятати, що калібрування і управління кольором – це два пов'язаних, але вирішуваних окремо питання. Калібрування – це настройка режиму роботи певного пристрою, такого як монітор, відповідно до заданих характеристик. Якщо відеоадаптер комп'ютера посилає сигнал для виведення певного кольору (який знаходиться в межах колірної охоплення монітора), то монітор повинен протягом усього дня і всіх наступних днів відображати правильний колір. З цієї точки зору калібрування є залежним від пристрою процесом. З іншого боку, управління кольором орієнтоване на отримання однакового кольору з декількох пристроїв, таких як принтери, монітори і сканери. Калібруючи монітор, користувач домагається того, що колірний профіль, який потім буде використовуватися системою управління кольором, залишається точним.

Методи калібрування. У менш дорогих продуктах для калібрування моніторів (таких як Optical компанії Color Partnership), які працюють зі звичайними, некалібруемими моніторами, також використовуються колориметри, але вони не взаємодіють з внутрішніми електронними схемами моніторів. Замість цього вони змінюють значення кольорів, що видаються відеоадаптерами комп'ютерів, коректуючи таблиці заміни кольорів (color lookup table, CLUT) адаптерів. За методом CLUT деякі колірні сигнали, що передаються на монітор, зменшуються. Наприклад, програмний калібратор, орієнтований на CLUT, не може збільшити рівень червоного кольору (цього можна досягти тільки за допомогою органів управління монітора), щоб зробити колірну температуру монітора більш «теплою». Замість цього калібруюча програма злегка приглушує величини синього і зеленого сигналів, що виходять з відеоадаптера. Зазвичай програма в змозі змінювати ці значення тільки в межах фіксованої кількості кроків, а не плавно, як виконується внутрішня настройка підсилювачів монітора. Тому калібрування через CLUT може привести до зниження яскравості зображення і, отже, недостатнє опрацювання деталей в тінях, а також до появи більш помітних, ніж в каліброваних моніторах, переходів від кольору до кольору.

Якщо в розпорядженні верстальника або художника є монітор з можливістю ручного налаштування рівнів червоного, зеленого і синього кольорів, колірної температури або точки білого, то ніщо не заважає йому налаштувати перенесення кольорів монітора вручну, тобто зробити те, що монітори виконують автоматично.

Калібрування монітора в домашніх умовах в програмі **Atrise lutcurve 1.5.2** (російська версія), калібрувати можна як РК, так і ЕПТ монітори. Ця програма володіє широким функціоналом.

Приступимо до калібрування монітора в програмі **Atrise lutcurve 1.5.2**.

1. Підготовчий етап калібрування монітора в програмі **Atrise lutcurve**.

Для початку потрібно прогріти монітор протягом, приблизно, півгодини, потім можна приступати до калібрування монітора. Нагадаю дуже важливу деталь про висвітлення робочого місця:

- на моніторі не повинно бути відблисків від вікна і інших яскравих джерел;

- перед калібруванням монітора бажано протерти його від пилу;

- робочий стіл бажано зробити сірим. Від стандартної блакитний схеми Windows також краще відмовитися, правильніше буде встановити класичну тему оформлення в сірих кольорах. Шпалери для калібрування монітора (для перевірки колірної настройки). Вони виконані в сірому кольорі.

- завантажити заводський профіль монітора (від виробника), якщо такого немає навіть на сайті виробника, вибираємо sRGB.

- відключити інші програми для калібрування монітора (якщо такі використовувалися).

- для ЕПТ моніторів вибираємо гаму $G = 2.2$, для РК – $G = 1.8$

2. Основний етап калібрування монітора в **Atrise lutcurve 1.5.2** складається з п'яти кроків, на кожному кроці можна повернутися до попереднього і підкоригувати налаштування.

1. Точка чорного (black point) – регулювання яскравості монітора (або рівня чорного для просунутих моніторів).

2. Точка білого (white point) – регулювання контрасту монітора.

3. Гамма (gamma) – настройка гами монітора.

4. Баланс кольорів (color correction) – точне регулювання установки колірної температури монітора.

5. Коректувальні точки (curve adjustment) – точне регулювання яскравості.

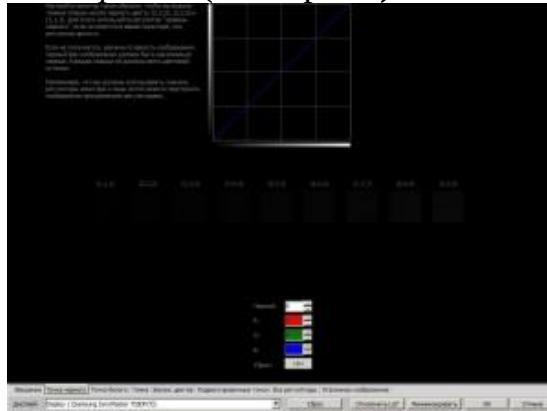
Запускаємо **Atrise lutcurve 1.5.2**.



Atrise lutcurve – програма для калібрування монітора в домашніх умовах.

У нижній частині вікна ви побачите вкладки п'яти кроків калібрування монітора. Для зручності на заключному етапі калібрування все програмні регулятори зібрані в одному місці. Після закінчення калібрування результат можна оцінити по еталонному зображенню.

Перший крок – *точка чорного* (black point).



Налаштування точки чорного складається з двох етапів:

1. Апаратне налаштування монітора.
2. Програмне налаштування монітора.

В першу чергу використовуємо апаратне налаштування – натискаємо на кнопки монітора і дивимосся результат.

На другому етапі для отримання більш точних результатів крутимо настройки програми **Atrise lutcurve**.

Необхідно налаштувати монітор таким чином, щоб плашки (3,3,3), (2,2,2), (1,1,1) стали помітні. Для цього необхідно використовувати регулятор яскравості на моніторі (або регулятор рівня чорного на дорогих моделях моніторів). Після цього можна використовувати програмні регулятори.

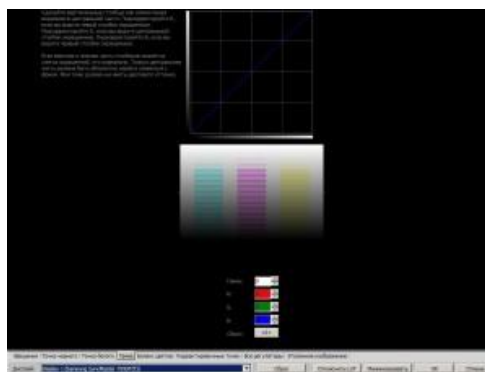
Чорний фон зображення повинен бути максимально чорним, плашки не повинні мати колірних відтінків.

Точка білого (white point).



Налаштуйте контрастність (спочатку використовуйте апаратні настройки монітора) таким чином, щоб бачити плашки (252,252,252), (253,253,253), (254,254,254). Плашки не повинні мати колірних відтінків.

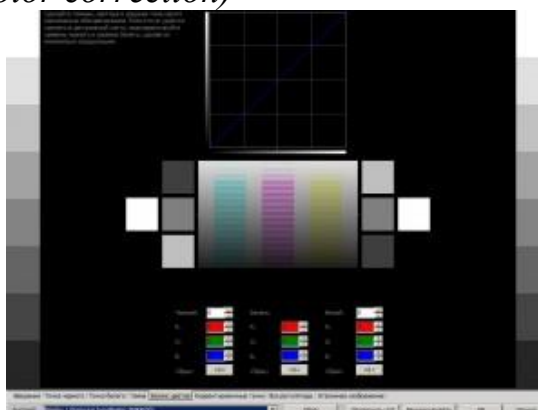
Гамма (gamma)



Необхідно налаштувати регулятори так, щоб вертикальні стовпчики стали якомога менше видимими. Підкоригуйте R, якщо ви бачите лівий стовпчик забарвленим. Підкоригуйте G, якщо ви бачите центральний стовпчик забарвленим. Підкоригуйте B, якщо ви бачите правий стовпчик забарвленим.

Наступний етап калібрування монітора в програмі **Atrise lutcurve** – баланс кольорів.

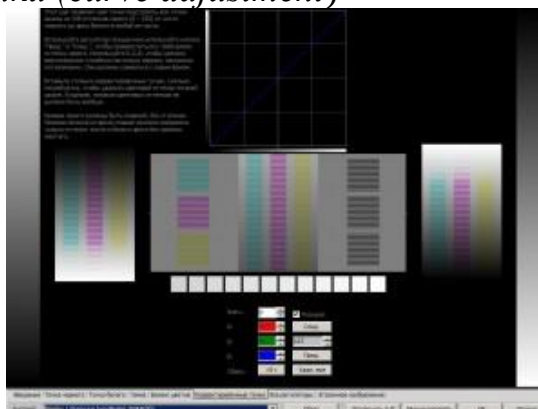
Баланс кольорів (color correction)



Зробіть темні, світлі тони та напівтони сірого максимально знебарвленими. Якщо це не вдається зробити в центральній частині, підкоректуйте рівень чорного і рівень білого, зробивши їх мінімально забарвленими.

Наступний етап калібрування монітора в Atrise lutcurve – коректувальні точки.

Коректувальні точки (curve adjustment)



Цей крок калібрування монітора дозволяє зробити точну настройку монітора, використовуючи коректувальні точки.

На даному етапі калібрування можна точно підлаштувати всі крапки шкали з 256 відтінків сірого від чисто чорного до яскраво білого в будь-якій її частини. Вставте стільки коригуючих точок, скільки буде потрібно, щоб видалити відтінок кольору по всій шкалі. В ідеалі, ніяких кольорних відтінків не повинно бути взагалі. Крива сірого повинна бути плавною, без сходинок.

На вкладці "Всі регулятори" зібрані всі налаштування програми **Atrise lutcurve**, які ми міняли в кроках 1-5. Тут ми бачимо всю "картину" і можемо підкрутити потрібні регулятори.

На цьому процес калібрування монітора в програмі **Atrise lutcurve 1.5.2** закінчено, тепер ми можемо подивитися еталонне зображення:



Якщо вас влаштовує результат то натискайте "Ok", якщо ні – поверніться до попередніх кроків.

На цьому процес калібрування монітора в програмі **Atrise lutcurve** закінчений.

Людське око на диво швидко пристосовується до будь-яких змін світла, тому будь-які програми для настройки монітора не зможуть замінити апаратного калібратора, але вони хоча б зроблять ці настройки більш реалістичними.

Практична частина

Завдання 1. Відкалібруйте монітор, використовуючи програму калібрування монітора.

Завдання 2. Порівняйте його з рідним профілем монітора.

При підготовці звіту по лабораторної роботи слід використовувати дані, які отримані в ході роботи.

Завдання 3. Вкажіть послідовність виконуваних дій і основні настройки програми калібрації.

ЗМІСТ ЗВІТУ

Звіт повинен містити найменування роботи, мету, виконані завдання, записи досліджень і вимірів, висновки по виконаній роботі.

ЛАБОРАТОРНА РОБОТА №8

Тема: Числа зі знаком. Операції з числами зі знаком.

Мета: Ознайомитись з формами представлення від'ємних чисел у комп'ютерах. Засвоїти виконання арифметичних операцій з числами зі знаком.

Теоретична частина

Розміщення різних типів даних у пам'яті ПК

Пам'ять комп'ютера є такою, що адресується, тобто кожна комірка пам'яті характеризується *вмістом* і *адресою*. В IBM ПК адресується кожна комірка розміром байт.

Карта пам'яті – це спосіб відображення вмісту та адреси пам'яті. Кожен прямокутник відображає вміст одного байта пам'яті (див. номери біт у байті: від старшого – сьомого, до молодшого – нульового). Зазвичай прямокутники зображують зверху вниз, вважаючи, що комірка з молодшою адресою є верхньою, а комірка зі старшою адресою – нижньою.

Представимо у пам'яті число 25 у зазначених форматах даних (див. рис. 1):

у форматі байта число 25 = 0001 1001_б.

у форматі слова число 25 = 0000 0000 0001 1001_б.

у форматі подвійного слова число 25 = 00000000 00000000 0000 0000 0001 1001_б.



Рисунок 1 – Розміщення різних типів даних у пам'яті комп'ютера.

ЗАМІТИМО! Старші байти числа завжди розташовуються за старшою адресою пам'яті!

ПАМ'ЯТАЄМО! 1 Мбайт = 1024 Кбайт; 1 Кбайт = 1024 байт.

Від'ємні числа

Якщо байт містить число зі знаком, його значення представляється лише молодшими сімома бітами (0-6); старший біт (біт 7) вказує знак числа, тому називається *знаковим*. *Знаковий біт S* дорівнює 0, якщо число додатне або дорівнює 0 і дорівнює 1 якщо воно від'ємне.

Існують три системи представлення чисел зі знаком:

- значення зі знаком;
- доповнення до одиниці;
- доповнення до двох.

У системі значення зі знаком від'ємні числа відрізняються від відповідних додатних чисел тим, що значення найстаршого біта у векторі

дорівнює не 0, а 1. Наприклад, число +5 представляється як 0101, а число -5 як 1101. Таке подання чисел називають ще **прямим кодом числа зі знаком**.

У поданні **доповнення до одиниці** від'ємні значення отримують шляхом доповнення кожного розряду відповідного додатного значення до одиниці, тобто. щоб зробити число від'ємним, потрібно замінити кожну одиницю нулем і кожен нуль одиницею. Таке подання чисел ще називають **оберненим кодом числа**.

У системі доповнення до двох операція доповнення проводиться шляхом віднімання числа з 2^n . Те саме значення можна отримати і шляхом додавання 1 до доповнення цього числа до одиниці. Таким чином, заперечення числа відбувається у два етапи. Спочатку кожна одиниця змінюється на нуль, а кожен нуль – на одиницю (як і в системі доповнення до одиниці). Потім до отриманого результату додається одиниця. Таке подання числа ще називають **додатковим кодом числа**.

Правила складання та віднімання n-розрядних чисел зі знаком у системі доповнення до двох:

1. Для складання двох чисел слід скласти їх n-розрядні подання, ігноруючи сигнал перенесення з позиції старшого розряду (MSB). Сумою буде правильне алгебраїчне значення, що представлене в системі доповнення до двох, якщо це значення лежить в діапазоні від -2^{n-1} до $+2^{n-1} - 1$.

2. Для віднімання чисел X і Y, тобто виконання операції $X - Y$, слід обчислити доповнення числа Y до двох, а потім додати його до числа X з урахуванням правила 1. Результатом буде правильне алгебраїчне значення, що представлене в системі доповнення до двох, якщо це значення лежить у діапазоні від -2^{n-1} до $+2^{n-1} - 1$.

У комп'ютерах числа зі знаком подаються у додатковому коді. Тобто для подання негативного двійкового числа необхідно інвертувати всі біти та додати 1.

Приклад:

Отримаємо додатковий код числа -65.

$65 = 01000001_b$

Інверсія числа $65 = 10111110_b$

Додамо 1 до інверсії, отримаємо 10111111_b , тобто **-65 = 10111111_b**.

На рисунку 2 наведено приклад виконання арифметичної операції $10-7$ в оберненому коді (доповнення до одиниці) та у доповненому коді (доповнення до двох).



Рисунок 2 – Додавання в системах з доповненням до одиниці та з доповненням до двох.

Практична частина

Завдання 1. Нарисувати карти пам'яті комп'ютера для представлення наступних чисел: -45; +45; -52; +52.

Завдання 2. Виконати наступні арифметичні операції у доповненому коді: 12-37; -8-12; 21-37.

Контрольні питання

1. Яке правило розміщення двійкових чисел у комірках пам'яті?
2. Які існують форми представлення чисел зі знаком? Яка з цих форм використовується у комп'ютерах?
3. Сформулюйте правила виконання арифметичних операцій з числами у додатковому коді.

ЛАБОРАТОРНА РОБОТА №9-10

Тема: Вивчення асемблерної архітектури мікропроцесора i8086 у середовищі відладчика TURBO-DEBBUGER (TD). Формування адреси комірки сегментованої пам'яті.

Мета: Вивчення структури регістрів мікропроцесора i8086. Ознайомлення із процесом формування фізичної адреси комірки сегментованої пам'яті.

Теоретична частина

Як відомо, МП виконує лише ту програму, яка завантажена на електронну (оперативну) пам'ять. Пам'ять складається з комірок, кожна комірка має свій унікальний номер або адресу. Програміст надає адресу ім'я, а програма-транслятор замінює ім'я на двійковий код. Розрядність *шини адреси* (ША) МП визначає допустиму кількість (простір) адрес.

Пам'ять МП i8086 організована досить незвично, що пояснюється поєднанням 1-мегабайтної пам'яті та 16-розрядних регістрів. У пам'яті ємністю 1 Мбайт для представлення адреси потрібно 20 біт. Отже, зберегти покажчик на пам'ять в одному 16-розрядному регістрі неможливо. В цілях вирішення проблеми пам'ять поділена на блоки або сегменти по 64 Кбайти, і адреси в рамках цих сегментів вміщуються в 16 біт.

Створюючи програму, програміст організовує – *визначає* – початок і кінець її сегментів та призначення осередків пам'яті в них. Для отримання програми, що виконується в ехе-форматі, необхідно визначити 3 наступні головні сегменти, а при необхідності – 1 додатковий:

1. *Сегмент кодів.* Містить комірки пам'яті, що зберігають виконувану програму (усю сукупність команд у машинних кодах). Перший байт першої команди знаходиться у першій комірці сегмента і т.д.
2. *Сегмент даних.* Містить комірки з даними програми: константами та робочими областями, зарезервованими для запису в них програмою; даних, що вводяться, наприклад, з клавіатури, а також для результатів, що обчислюються програмою. Дані, що оброблюються МП, часто називають операндами, так як над ними виконується закодована у команді операція.
3. *Сегмент стеку.* Містить комірки, запис та зчитування яких виконується за особливим алгоритмом, відмінним від інших сегментів. Використовується для тимчасового зберігання команд (кодів), даних та проміжних результатів.
4. *Додатковий сегмент.* Містить комірки для зберігання даних та результатів, додаткових до сегмента даних.

З урахуванням сегментної адресації пам'яті, адреса будь-якої комірки пам'яті визначається МП додаванням двох складових (логічних адрес). Перша складова – це адреса сегмента, в якому знаходиться шукана комірка. Друга – це адреса комірки, що відлічується від початку цього сегмента, і називається тому *адресою зміщення* комірки в сегменті. Коротко називатимемо ці адреси $A_{\text{сегм}}$ і $A_{\text{зміщ}}$. Покажемо в карті пам'яті розміщення певного сегмента та його вмісту. Слід пам'ятати, що нумерація ЕОМ прийнята з цифри 0.

На рис. 1 виділено комірку з номером 5, що містить байт, тобто $A_{зміщ1} = 5$. Якщо у пам'яті розміщено слово (рис. 2), воно вказується (адресується) через свій молодший байт: $A_{зміщ2}$ слова у пам'яті = 3.

Пам'ять має байтову організацію – двобайтове слово розміщується у суміжних комірках, причому старший байт займає осередок із великим номером. Адресою слова є адреса молодшого байта.

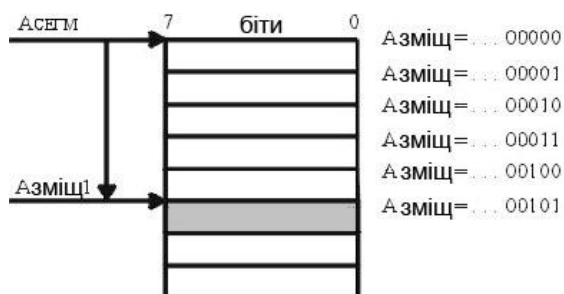


Рисунок 1 – Розміщення в пам'яті байту.

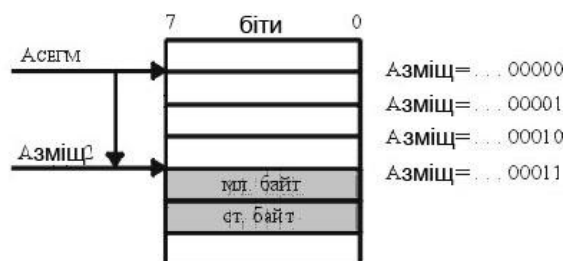


Рисунок 2 – Розміщення в пам'яті слова.

Такий спосіб адресації комірок пам'яті зручний тим, що при зміні $A_{сегм}$ не потрібно змінювати $A_{зміщ}$: значення адрес зміщень завжди відраховуються від початку сегмента і не залежать від значення адреси сегмента.

Таким чином, у МП повинні зберігатися адреси всіх сегментів програми, що виконується. При кожному зверненні до пам'яті з метою прочитати команду або операнд або записати результат, у МП має бути сформована *виконавча адреса* необхідної комірки як сума $A_{сегм}$ та $A_{зміщ}$, який МП вибирає з команди або обчислює за даними команди.

Сукупність регістрів, що доступні програмісту і використовуються ним під час складання програм мовою АСЕМБЛЕР чи іншій машинно-орієнтованій мові, називається *програмістською моделлю*.

Програмістська модель МП i8086 містить: регістри загального призначення, сегментні регістри, регістр-показчик команд та регістр ознак – прапорів. Усі вони є 16-розрядними регістрами.

Регістри загального призначення – це регістри, які можна використовувати у будь-яких арифметичних, логічних тощо машинних операціях.

Регістри даних AX , BX , CX , DX служать для зберігання операндів та результатів операцій. Можлива адресація як цілих регістрів, і їх молодшої і старшої частин, які мають назву L і H відповідно. Деяким регістрам поряд із загальним призначенням надаються й спеціальні. У останньому випадку у відповідних командах ці регістри вказуються неявно самим кодом операції. Так, у деяких командах

- регістр AX використовується як акумулятор, тобто регістр, в якому знаходиться один з операндів і який міститься результат операції;
- регістр BX – як базовий регістр, що використовується при непрямій адресації;
- регістр CX – як лічильник кількості зміщених біт у командах зсуву, лічильник числа повторів у командах управління обчислювальними циклами та в операціях з ланцюжками байт;
- регістр DX неявно адресується в командах множення та поділу, а в деяких операціях введення-виводу зберігає адресу порту введення-виводу.

Хоча всі регістри зміщень та сегментів є 16-розрядними, МП видає на шину адреси при зверненнях до оперативної пам'яті (ОП) 20-розрядні виконавчі (фізичні) адреси, що дозволяють звертатися до ОП місткістю 1 Мбайт. Це стає можливим завдяки механізму сегментації пам'яті. Розмір сегмента не може перевищувати 64 Кбайт. Допускається перекриття сегментів. Базові (початкові) 16-розрядні адреси сегментів, що зберігаються у відповідних сегментних регістрах, трактуються як 20-розрядні з нулями у чотирьох молодших розрядах. Іншими словами, початкові адреси мають бути кратними 16.

Фізична адреса операнда чи команди формується, як показано на рис. 3 додаванням початкової адреси відповідного сегмента (*вмісту сегментного регістру, зсунутого на чотири розряди вліво, тобто доповненого нулями в чотирьох молодших розрядах*) та адреси зміщення, так само званого *ефективною адресою* ЕА. Адреса зміщення обчислюється в МП на основі інформації про спосіб формування адреси операнда, що міститься в команді.

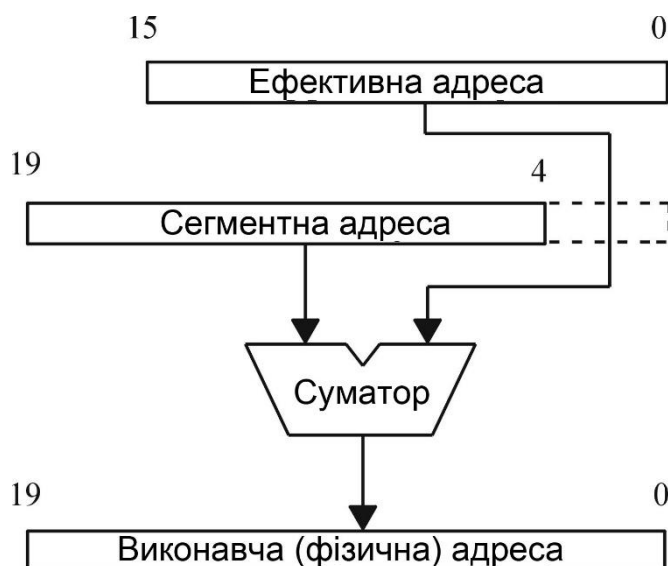


Рисунок 3 – Формування фізичної адреси.

Групу вказівних та індексних регістрів, що задають внутрішньосегментні зміщення, утворюють такі регістри: покажчика стека (SP), покажчика бази стека (BP), індексу операнда (джерела) (SI) та індексу результату (приймача) (DI). До цієї групи можна віднести і регістр покажчика (лічильника) команд – IP, що часто називається *лічильником команд*. Причому сегментні, індексні та вказівні регістри застосовуються попарно, відповідно до сегментної адресації наступним чином:

CS й *IP* – адресують разом вміст сегмента коду програми;

SS й *BP* або *SS* й *SP* – адресують сегмент стека;

DS і один із регістрів *BX*, *SI*, *DI* – адресують сегмент даних.

Спеціальне призначення регістрів наведено нижче.

Регістри загального призначення

AX	Регістр-акумулятор – зберігання проміжних даних та результатів
BX	Базовий регістр, використовується для базових типів адресації або як покажчик адреси пам'яті
CX	Регістр-лічильник для керування числом проходів у циклі

<i>DX</i>	Регістр даних для зберігання даних, зберігання адреси порту вводу/виводу
------------------	--

Регістри сегментів та покажчик команд

<i>CS</i>	Регістр сегмента коду програми, вказує сегмент, що містить адресу поточної програми, що виконується
<i>DS</i>	Регістр сегмента даних, вказує адресу початку сегмента даних
<i>ES</i>	Регістр додаткового сегмента, вказує адресу початку додаткового сегмента
<i>SS</i>	Регістр сегмента стека, вказує адресу початку стека

Індексні реєстри та реєстри-покажчики

<i>SI</i>	Індексний реєстр джерела, вказівник адреси рядка (масиву) - джерела із сегмента даних, базово-індексна адресація
<i>DI</i>	Індексний реєстр призначення, індексний реєстр джерела, покажчик адреси рядка (масиву) – приймача з додаткового сегмента, базово-індексна адресація
<i>SP</i>	Регістр-покажчик стека, його вміст вказує адресу елемента на вершині стека
<i>BP</i>	Регістр-покажчик бази, додатковий покажчик під час роботи зі стеком, базово-індексна адресація
<i>IP</i>	Вказівник команд, що вказує адресу наступної команди в сегменті коду програми, при виконанні програми його вміст змінюється автоматично

Регістр ознак (прапорів) має 16 розрядів, причому молодші 8 розрядів відповідають реєстру прапорів МП i8080. У реєстрі ознак зберігаються:

- сформовані в АЛП такі ознаки результату виконаної операції:
 - переповнення OF (при операціях із цілими числами);
 - знаку SF, якщо результат негативний – SF=1;
 - нуля ZF якщо результат нульовий – ZF=1;
 - допоміжного перенесення AF (перенесення з третього або позики до третього розряду);
 - паритету PF (PF=1, якщо число одиниць у молодшому байті результату непарне);
 - перенесення CF (CF=1, якщо було перенесення зі старшого або позику до старшого розряду результату);
- ознаки керування, що встановлюються пристроєм керування:
 - покрокового режиму TF (керує покроковими перериваннями)
 - дозволи переривання IF (дозволяє або забороняє переривання, що маскуються);
 - напрямки DF (вказує напрямки обробки ланцюжка даних, починаючи з елемента з найменшою адресою при DF=0 або з найбільшою адресою при DF=1).

Практична частина

1. На рисунку 4 наведено робочий екран програми- відладчика TD з деякою робочою програмою, відкритою у вікні CPU. Вивчіть робочий екран відладчика.

2. Знайдіть усі вікна TD:

– вікно CPU, де відображено текст завантаженої програми мовою Assembler, а ліворуч – адреси зміщення комірок пам'яті в сегменті коду зі своїм вмістом – H-кодами інструкцій МП;

– вікно регістрів МП, а в ньому знайдіть РОН, ПОКАЖЧИКИ, ІНДЕКСНІ, СЕГМЕНТНІ регістри та регістр ПРАПОРІВ;

– вікно сегмента даних, а в ньому визначити, скільки байт відображається в одному рядку картки пам'яті сегмента;

– вікно сегмента стека.

3. Знайдіть усі ресурси цієї програми (сегменти ОЗУ та їх адреси, а також регістри МП). Письменна виконати завдання 1 – 12.

4. Виконайте завдання 13 – 14. Ці завдання вимагають від вас умінь вводити зміни (нові числові значення) в комірки та регістри. Якщо забули, як це робити, скористайтесь *Інструкцією з роботи з TD*.

5. Складіть **звіт про роботу**.

Завдання до роботи

1. Випишіть адресу поточного сегмента даних, нульового, а потім першого його байтів.

2. Обчисліть фізичну адресу осередку поточного сегмента даних з повним покажчиком DS:0005.

3. Вкажіть, яке значення знаходиться у сегменті даних за адресою зміщення $00010 = 000A_h$. Це значення байта чи слова?

4. Визначте вміст комірок DS:0003_h та DS:0011_h. У якому сегменті вони розміщені?

5. Вкажіть у h-кодах та b-кодах вміст регістрів AH, AL, AX.

6. Знайдіть у сегменті кодів одну з команд. Випишіть у звіт її повний покажчик, мнемокод команди мовою Assembler і h - код її інструкції МП.

7. Знайдіть третій байт і третє слово в сегменті даних, випишіть їх адреси (повний покажчик) і значення в h- і b- кодах.

8. Знайдіть команду мови Assembler за її наступним повним покажчиком: CS:0013_h. Випишіть у звіт цю команду мовою Assembler та її h - код інструкції.

9. Випишіть у звіт h-кодах та b-кодах вміст осередку пам'яті за адресою CS:0005_h.

10. Введіть константу 0F2h у сегмент даних за адресою усунення 0001_h

11. Введіть вміст комірки DS:09A6_h у будь-яку комірку стека. Скільки комірок стеку та з якими повними адресами використано під введену константу?

12. Введіть вміст регістру DX у сегмент даних за адресою зміщення DS:0000_h.

13. Змініть значення прапора Z. Який є сенс цього прапора.

14. Спробуйте записати константу C2_h на адресу: DS:0003_h. Чи змінився вміст осередку пам'яті? Чому? Переведіть константу C2_h у десяткову систему числення, перевірте її формат у h-кодах і виправте вихідний формат і знову спробуйте записати цю константу в той самий осередок. Опишіть у звіті результату своєї **роботи**.

Зміст звіту:

1. Тема та N лаб. роботи
2. Тексти завдань з 1 по 14 та результати їх виконання.
3. Екран програми TD.EXE після виконання завдання 14 лаб. роботи на Вашому комп'ютері (а не) рис. 4 у даних методвказівках.

Примітка: Тексти завдань та питань записувати повністю!

Контрольні питання

1. Назвіть усі сегментні регістри.
2. Який сегмент даних називається поточним?
3. Як називається вміст комірки сегмента кодів?
4. Яка розрядність регістрів загального призначення?
5. Що містить регістр IP?
6. Які регістри за замовчуванням адресують елемент сегмент даних?
7. Який регістр вказує на усунення в сегменті кодів?
8. У яких регістрах не можна зберігати операнди?
9. У яких регістрах можна зберігати адреси зміщення?
10. Наведіть приклад числового значення повного покажчика в h-кодах
11. Наведіть приклади чисел: 4, 35, 335 у форматах байта та слова.
12. Нарисуйте картку пам'яті для наступних сегментів:
сегмента коду розміром 5 слів;
сегмента даних з трьома будь-якими константами у форматі байта, та змінною з ім'ям T1, що вказує на п'яту комірку сегмента;
сегмент додаткових даних з однорозрядним, дворозрядним та трирозрядним десятковими числами в кодах ASCII.

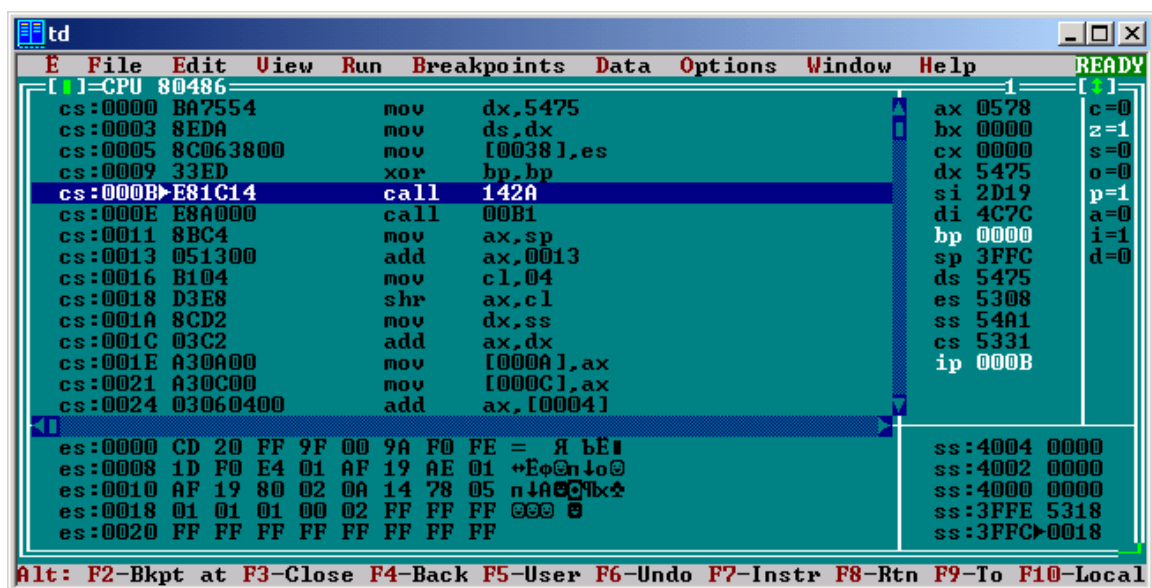


Рисунок 4 – Робочий екран програми TD.

ЛАБОРАТОРНА РОБОТА №11

Вступ в основи програмування на мові Асемблера

Мета роботи: практичне оволодіння навичками складання найпростіших програм на мові Асемблера і роботи з програмами TASM та TLINK.

Теоретичні відомості

Незважаючи на те, що сучасні мови програмування високого рівня забезпечують не тільки зручне, але й ефективне системне програмування, в тих випадках, коли особливо важливо отримати оптимальний об'єктний код, доцільно використовувати Асемблер. Асемблер є машинною мовою в символічній формі, яка досить зрозуміла і зручна програмісту.

Етапи створення програми на Асемблері

Повний цикл створення програми на Асемблері можна представити у вигляді послідовності чотирьох етапів, які показані на рис. 1.

Початковий модуль програми створюється в будь-якому текстовому редакторі, наприклад, в «Блокноті» і зберігається у вигляді файлу з ім'ям, привласненим за правилами **MS DOS**, з обов'язковим розширенням **asm**. Для нашої першої програми це буде **hello_1.asm**.

Для отримання виконуваного модуля, який можна запустити на виконання, потрібно послідовно виконати етапи **трансляції** та **компонування**. Для цього використовуються програми, що входять до складу пакета Асемблера **Turbo Assembler (TASM)** фірми **Borland**.

Трансляція здійснюється за допомогою **компілятора** Турбо Асемблера, який є виконуваною програмою **tasm.exe**, що працює в режимі командного рядка. Він викликається командою **DOS**:

tasm/z/zi/n <ім'я файлу><ім'я файлу>< ім'я файлу>,

де **/z** – ключ, що дозволяє виведення на екран рядків вихідного тексту програми, в яких Асемблер виявив помилки;

/zi – ключ, що керує включенням в результуючий файл повних відомостей про номери рядків і імена вихідного модуля;

/n – ключ, який виключає з лістингу інформацію про символічні позначення в програмі.

Наступні далі **імена трьох файлів** – вихідного (***.asm**), об'єктного (***.obj**) і лістингу (***.lst**) можна використовувати без розширень, наприклад

tasm/z/zi/n hello_1 hello_1 hello_1

Якщо вихідний модуль не містить помилок, то на екран виводиться повідомлення про успішну трансляцію, а в поточному каталозі з'являться нові файли – об'єктний (**hello_1.obj**) і лістингу (**hello_1.lst**).



Рисунок 1 – Етапи створення асемблерної програми

Компонування об'єктних модулів з бібліотечними модулями проводиться викликом компоновщика **tlink.exe** з командного рядка:

tlink/v <им'я файла>

Ключ **/v** передає в завантажувальний файл інформацію, яка використовується при налагодженні програм. Наступне далі **ім'я файлу** позначає ім'я об'єктного модуля. Розширення в цьому імені можна не вказувати.

Для нашого прикладу компоновка буде здійснюватися за допомогою такої команди:

tlink/v hello_1

У разі успішного закінчення компонування в поточному каталозі з'являється виконуваний файл – завантажувальний модуль **hello_1.exe**, і файл карти збірки **hello_1.map**.

Завантажувальний модуль може бути запущений на виконання командою **DOS hello_1.exe**.

Для налагодження створюваних програм використовується програма-відладчик **Turbo Debugger (td.exe)** фірми **Borland**.

Структура початкового модуля

Вихідний модуль програми на Асемблері – послідовність рядків, що містять **командні оператори** та **директиви**, яка надається як функціональна одиниця для подальшої обробки.

Вихідний модуль створюється текстовим редактором і зберігається у вигляді текстового файлу з розширенням ***.asm**.

Алфавіт допустимих символів Асемблера включає в себе великі та малі літери англійського алфавіту, арабські цифри і деякі спеціальні символи, які в Асемблері мають особливий сенс. Крім того, допускається використання деяких недрукованих символів, які керують роботою ЕОМ: **пробіл (20h)**, **перевод рядка (0Ah)**, **повернення каретки (0Dh)** и **табуляція (09h)**.

Великі та малі літери Асемблером не розрізняються, за винятком символічних констант.

Командні оператори або просто команди мають наступний формат:

[мітка:] [префікс] мнемоніка [операнд(и)] [;коментар],

де **мітка** – визначаємо користувачем ім'я команди, що закінчується двокрапкою. Значенням мітки є адреса зазначеної команди. Використовуються для організації команд передачі управління. Мітка складається з послідовності **символів** або **цифр**, проте завжди починається з символів **англійського алфавіту** або з символів @, _, ?;

префікс – елемент, який служить для зміни стандартного дії команди;

мнемоніка – мнемонічне позначення команди, яка представляє собою ключові слова Асемблера і ідентифікує виконувану командою операцію. Зазвичай використовуються скорочені англійські слова, що передають зміст команди;

операнд(и) – об'єкти, які беруть участь у зазначеній операції. Це можуть бути адреси даних або безпосередньо самі дані, необхідні для виконання команди. Команди можуть бути двох, одне і безоперандними. Якщо операндів два, то вони розділяються комою;

коментар – починається з крапки з комою і призначений для пояснення до програми. Може виконуватися російською мовою, тому що не впливає на виконання програми.

Мітка, префікс, мнемоніка і операнд (и) поділяються принаймні одним пробілом один від одного.

Кожна команда при трансляції генерує машинний код команди, розмір якого залежить від способів завдання операндів.

Директиви Асемблера дозволяють управляти процесом асемблювання і формування лістингу. Їх використовують для розподілу пам'яті, забезпечення зв'язку між програмними модулями і роботи з символічними іменами. Часто їх називають **псевдокомандами**. Формат директив схожий на формат команд:

[ім'я] директива [операнд(и)] [;коментар],

де **ім'я** – ім'я директиви. Ніколи не закінчується двокрапкою і має зовсім інший сенс в порівнянні з міткою. Деякі директиви обов'язково повинні мати ім'я, у деяких воно може бути відсутнім;

директива – аналогічна полю мнемоніки команди і містить одне з ключових слів Асемблера, що позначає назву директиви;

операнд(и) – аналогічні операндам команд і конкретизують дії, що виконуються за даною директивою;

коментар – пояснення до директиви.

На відміну від команд директиви діють тільки в процесі асемблювання програми і не генерують машинних кодів.

Вихідний модуль, як правило, складається з декількох фрагментів програми, згрупованих за функціональними ознаками – **логічних сегментів**. Кожен із сегментів починається з директиви **SEGMENT** (початок сегмента), яка обов'язково повинна мати унікальне ім'я, і закінчується директивою **ENDS** (кінець сегмента) *з тим же ім'ям*. Імена сегментів, як правило, несуть смислове навантаження, асоціюючись із призначенням сегмента програми:

ім'я SEGMENT [параметри]

.....

ім'я ENDS

У **сегменті даних** програми визначаються дані (змінні, символні ланцюжки, масиви та ін.) з якими буде працювати програма. Для визначення даних найчастіше використовуються директиви **DB (Define Byte)** (визначить байт), **DW (Define Word)** (визначить слово), **DD (Define Doubleword)** (визначить двійне слово), рідше **DQ (Define Quadword)** (визначить чотири слова) і **DT (Define Tenbyte)** (визначить десять байт). Директиви визначення даних мають наступний формат:

[ім'я змінної] DB/DW/DD <поч. знач.> [, <поч. знач.>] ...

У них визначається: скільки одиниць пам'яті (байт) необхідно зарезервувати для змінної з вказаним ім'ям і як їх формувати, якщо задані початкові значення. В поле операнда потрібно хоча б одне початкове значення, але допустимо і список початкових значень, елементи якого розділяються комами. Зразковий вид типового сегмента даних представлений нижче:

Data	SEGMENT	;початок сегмента з ім'ям Data
var_1	DB 11000110b	;визначити змінну var_1 розміром
		;байт з початковим значенням 11000110b
var_2	DW 9FFh	;визначити змінну var_2 розміром
		;слово з початковим значенням 9FFh
var_3	DW ?	;визначити змінну var_3 розміром
		;слово не ставлячи її початкового значення
string	DB 'Assembler'	;визначити рядок символів string
		;кожен символ якої має розмір
		;байт з початковим значенням Assembler
mas_1	DB 0, -3, 28, 46, 39	;визначити масив з ім'ям mas_1
		;що складається з п'яти числових елементів
		;розміром байт
Data	ENDS	;кінець сегмента з ім'ям Data

Як видно з наведеного прикладу, для завдання початкових значень можуть використовуватися числові константи, представлені в двійковій (Binary – **b**), шістнадцатеричній (Hexadecimal – **h**), восьмеричній (Octal – **q**) або десятковій (Decimal – **d**) системах числення. При цьому за молодшою цифрою числа повинен слідувати однобуквений дескриптор системи числення. **Якщо шістнадцатерична константа починається з букви, то вона обов'язково повинна бути доповнена зліва незначущим нулем.**

Після того як змінна оголошена в сегменті даних, кожна з них пов'язується з трьома атрибутами, які використовуються програмою:

сегмент (SEG) – ідентифікує сегмент, що містить змінну;

зміщення (OFFSET) – являє собою відстань в байтах від початку сегмента до змінної;

тип (TYPE) – ідентифікує одиницю пам'яті, що виділяється для зберігання змінної, тобто байт, слово, подвійне слово і т. д.

У **сегменті стека** програми резервуються осередки пам'яті зазначеного розміру для організації тимчасового сховища даних і результатів проміжних обчислень при виконанні програми. Типовий сегмент стека наводиться нижче:

Stk SEGMENT	;початок сегмента з ім'ям Stk
DB 256 DUP (?)	;відвести під стек 256 байт
Stk ENDS	;кінець сегмента з ім'ям Stk

Для резервування осередків пам'яті для стека використовується директива **DB** з операндом **256 DUP (?)**. Ця конструкція **DUPlicate** (повторювати) мають такий вигляд:

n DUP (<поч. знач.> [, поч. знач., ...]),

тут параметр **n** задає число повторень елементів, що знаходяться в круглих дужках.

У **сегменті коду** програми наводиться послідовність асемблерних команд, відповідно до алгоритму розв'язуваної задачі. Типова побудова сегмента коду наводиться нижче. Як і інші сегменти він починається і закінчується директивами **SEGMENT** і **ENDS** з ім'ям Code. Однак при його написанні слід дотримуватися певних правил:

– перша команда сегмента повинна мати **мітку**, яка є точкою входу в програму;

– дві перші команди сегмента ініціалізують (завантажують) **сегментний регістр даних DS** сегментним (базовим) адресою сегмента даних. Оскільки **сегментні регістри не допускають безпосереднього завантаження**, вона проводиться через **РОН**, в даному випадку через **регістр-акумулятор AX**;

– закінчуватися сегмент повинен **коректним виходом в DOS**, що забезпечується трьома останніми командами.

Типовий вид сегмента коду показаний нижче.

ASSUME DS:Data, SS:Stk, CS:Code	;призначити сегментні регістри
Code SEGMENT	;початок сегмента з ім'ям Code

Start:

MOV AX, Data	;завантажити сегментний регістр DS
MOV DS, AX	;сегментним адресою сегмента даних
.....	;тут розташовуються
.....	;команди Асемблера
.....	;відповідно до алгоритму
.....	;завдання
MOV AL, 0	;завершити програму
MOV AH, 4Ch	;за допомогою
INT 21h	;DOS
Code ENDS	кінець сегмента з ім'ям Code
END Start	;кінець початкового модуля

Перед першим використанням сегментних регістрів і перед кожним місцем у програмі, де їх вміст може змінитися, необхідна директива **ASSUME** (припустити, вважати). Доцільно розташовувати її перед сегментом кодів. Директива має наступний формат:

ASSUME <SR: базова адреса>, [<SR: базова адреса>,...] ,
тут **SR** – сегментний регістр (**DS**, **SS**, **CS** або **ES**), а базова адреса задає сегментна адреса області пам'яті, яка адресується через цей сегментний регістр. Задати базову адресу сегментів можна за допомогою їх імен, що і робиться в наведеному вище прикладі.

Якщо **ASSUME DS:Data, SS:Stk, CS:Code**, то це означає, що базова адреса сегмента даних з ім'ям **Data** повинен знаходитися в регістрі **DS**, сегмента стека з ім'ям **Stk** – в регістрі **SS**, а сегмента коду з ім'ям **Code** – в регістрі **CS**. Проте, ця директива не звільняє програміста від початкової ініціалізації сегментних регістрів (див. Дві перші команди сегмента кодів).

Завершується вихідний модуль директивою **END**, як операнд якої використовується точка входу в програму (**мітка першої команди**). Тим самим повідомляється Асемблеру про досягнення кінця вихідного модуля і дається вказівка почати виконання програми з команди, поміченою міткою **Start**.

Порядок написання сегментів в вихідному модулі значення не має, однак після завантаження програми в пам'ять, розташування сегментів буде таким же, як у вихідному модулі.

Перша програма на Асемблері

Досліджувана в даній роботі програма **Hello_1**, дозволяє вивести на екран монітора рядок тексту з ім'ям **Greet – Hello, My friends!** (см. п. 2 **Завдання на виконання роботи**).

Початковий модуль програми організований у вигляді трьох сегментів, як було описано вище.

Даними є виведений на екран рядок, який і оголошується в сегменті даних. Цифри **13** і **10** в рядку оголошення змінних є керуючими символами і

означають коди перекладу рядка і повернення каретки. Замикає символний рядок перевизначення символ '\$' (долар) це змінна, яка є лічильником поточної адреси після визначення символного рядка. Часто використовується для обчислення довжини символного змінної.

У сегменті стека з ім'ям **Ourstack** резервуються 256 комірок пам'яті розміром байт для організації стека.

У сегменті коду з ім'ям **Code**, після ініціалізації сегментного регістра DS, слідують команди виведення рядка символів на екран.

Для цього:

- в регістр **АН** записується номер функції виведення на екран **09h**;
- в регістр **DX** завантажується початкова адреса рядка, що виводиться символів **OFFSET Greet**. Довжину рядка, що виводиться можна не вказувати, тому що висновок буде відбуватися до символу долара в кінці рядка;
- виконується команда переривання **int 21h**, що викликає переривання з типом **21h**, яке дозволяє звернутися до службових функцій операційної системи **MS DOS**.

В результаті відбувається звернення до службових функцій **MS DOS**, з яких вибирається функція виведення рядка символів і рядок з вказаним ім'ям з'являється на екрані.

Аналогічно проводиться завершення програми:

- в регістр **AL** заноситься код успішного завершення програми **0**;
- в регістр **АН** записується номер функції завершення **4Ch**;
- виконується команда переривання з типом **21h**.

В результаті відбувається звернення до службових функцій **MS DOS** з яких вибирається функція завершення програми і програма коректно завершує свою роботу.

Особливості роботи з Асемблером для MS DOS

На комп'ютерах з 64-х розрядними операційними системами Windows 7, 8, 10 не вдається запустити програми реального режиму **MS DOS** через несумісність. В цьому випадку використовуємо емулятор системи **DOS – DOSBox**. **DOSBox** є вільно розповсюджуваної програмою з відкритим вихідним кодом. Для роботи через **DOSBox** потрібно використовувати деякі прості прийоми, описані нижче.

Припустимо, що Ваші файли зберігаються на диску **D:\UCHEBA\ASM**. Для роботи з використанням **DOSBox** потрібно виконати наступне:

- запустити **DOSBox** з робочого столу комп'ютера. На екрані з'являться два вікна, одне з яких є вікном стану і служить для виведення службових повідомлень, а друге є робочим. Вікно стану можна згорнути.
- створити локальний робочий каталог з ім'ям, що збігається з ім'ям диска, на якому знаходяться Ваші файли. Для цього в командному рядку робочого вікна **DOSBox** набираємо:

z:\>mount D D:

Після повідомлення про те, що робочий каталог створений, працюємо з командного рядка **DOSBox** як зі звичайного командного рядка **MSDOS**:

z:\>D: ; перейти до кореневого каталогу диска D;

D:\>cd UCHEBA\ASM; перейти в папку ASM, зробивши її поточною;

D:\UCHEBA\ASM>tasm/z/zi/n hello_1 hello_1 hello_1; запустить

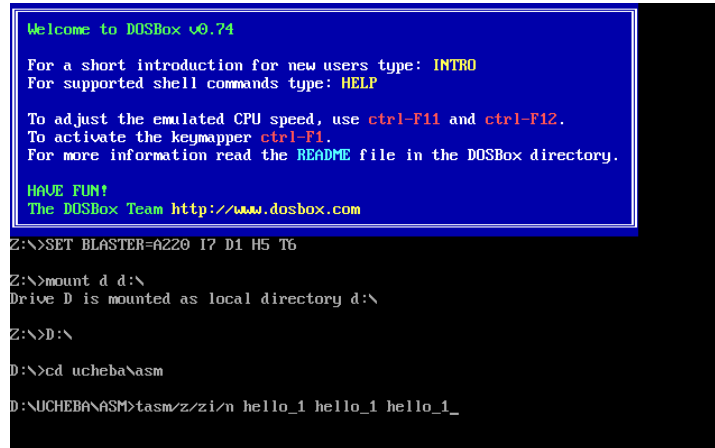


Рисунок 2 – Початкове вікно DOSBox і команди, які відображаються в ньому

; програму tasm.exe

DOSBox володіє чималим набором можливостей, які можна дізнатися набравши в командному рядку

z:>intro

Для отримання відомостей про команди DOS можна набрати

z:\>help

Наступні комбінації натискань клавіш дозволяють управляти режимами **DOSBox**:

ALT+Enter – перемикання повний/зменшений екран;

CTRL+F1 – перепризначення кнопок клавіатури;

CTRL+F5 – зробити знімок екрана як файл *.png;

CTRL+F9 – закрить програму;

Escape – закрити графічне вікно.

Підготовка до роботи

1. Вивчити методичні вказівки та рекомендовану літературу.
2. Підготувати відповіді на контрольні питання.

Завдання на виконання роботи

1. На диску **C** або **D** створити папку (каталог) з ім'ям «**Family name**», наприклад:

D: \USER \Petrov.

Скопіювати в створену папку файли Турбо Асемблера **tasm.exe**, **tlink.exe** і **td.exe** з папки **TASM**.

2. Використовуючи текстовий редактор, створити і відредагувати вихідний модуль програми **hello_1.asm**, текст якого наведено нижче.

```
;Program Hello_1 – Ваша перша програма
Data SEGMENT
    Greet DB 'Hello, My friends!', 13, 10, '$'
Data ENDS
Ourstack SEGMENT Stack
DB 100h DUP (?)
Ourstack ENDS
ASSUME CS:Code, DS:Data, SS:Ourstack

Code SEGMENT
Start:    mov AX, Data
          mov DS, AX
          mov AH, 09h
          mov DX, OFFSET Greet
          int 21h
          mov AL, 0
          mov AH, 4Ch
          int 21h
Code ENDS
          END Start
```

;Відкрити сегмент даних
;Визначити рядок
;символів з ім'ям Greet
;Закрити сегмент даних
;Відкрити сегмент стека
;Відвести під стек 256 байт
;Закрити сегмент стека
;Призначити сегментні
;реєстри
;Відкрити сегмент кодів
;Ініціалізувати
;сегментний реєстр DS
;Вивести рядок Greet
;на екран за допомогою
;DOS
;Завершити програму
;за допомогою
;DOS
;Закрити сегмент кодів
;Кінець вихідного модуля

3. Використовуючи компілятор Турбо Асемблера **tasm.exe** створити файли **hello_1.obj** і **hello_1.lst**. Переглянути на екрані тексти створених файлів **hello_1.obj**, **hello_1.lst** і проаналізувати їх.

4. Використовуючи компоувальник **tlink.exe** створити файли **hello_1.exe** і **hello_1.map**. Вивести на екран файли **hello_1.exe**, **hello_1.map** і проаналізувати їх.

5. Переконалися в працездатності програми **hello_1**, запустивши її з командного рядка.

6. Створити, для прискорення процесу асемблювання і компоування, командний файл з будь-яким ім'ям з розширенням **bat** (***.bat**). Для цього в текстовому редакторі «Блокнот» наберіть текст, що складається з послідовності виконуваних команд **DOS** і збережіть його з розширенням **bat**:

```
tasm /z /zi /n hello_1 hello_1 hello_1
tlink /v hello_1
```

Видаліть з поточного каталогу всі файли **hello_1**, крім вихідного, і перевірте працездатність створеного командного файлу, запустивши його з командного рядка.

7. Створити універсальний командний файл **run.bat**, який можна використовувати для асемблювання і компонування будь створюваної Вами програми. Для цього в командному файлі, створеному в п. 6, імена файлу слід замінити на символи **%1**:

tasm/z/zi/n %1 %1 %1

tlink /v %1

Для запуску універсального командного файлу потрібно в командному рядку після **імені командного файлу** (без розширення) вказати ім'я Вашого **вихідного модуля** без розширення, наприклад:

run hello_1 ;запустити командний файл з ім'ям **run**
;для вихідного модуля **hello_1**.

Видаліть з поточного каталогу всі файли **hello_1**, крім вихідного, і перевірте працездатність створеного командного файлу, запустивши його з командного рядка.

8. Внести зміни в програму **hello_1**, які змусять її виводити на екран ще два рядки символів, наприклад, «**My name is Family name**» і «**My group KS-18**». Для цього створіть новий вихідний модуль **hello_2.asm**, виконайте асемблювання і компоновку, після чого переконайтеся в працездатності програми.

Вимоги до звіту

- мета роботи;
- лістинги вихідного модуля і всіх файлів, створених в процесі асемблювання і компонування для програм **hello_1** і **hello_2** з коментарями: ***.asm, *.obj, *.lst, *.map, *.exe**;
- лістинги створених командних файлів ***.bat** з коментарями.

Контрольні питання

1. Команди директиви Асемблера. Формат і відмінності.
2. Яка мета сегментації пам'яті?
3. Що таке базова адреса сегмента?
4. Які значення може приймати базова адреса сегмента?
5. Який максимальний розмір сегмента і чому?
6. З яких логічних сегментів складається вихідний модуль асемблерной програми?
7. Якими директивами описується сегмент?
8. Як описуються різні типи даних, що використовуються програмою?
9. Яке призначення директиви ASSUME?
10. У чому полягає ініціалізація сегментних регістрів?
11. Що таке асемблювання і компоновка програми?
12. Що являє собою вихідний модуль програми?
13. Опишіть стандартний початок і закінчення сегмента кодів?
14. Який зміст файлів з розширеннями ***.ASM, *.LST, *.OBJ, *.MAP, *.EXE**?

15. Яке призначення і в чому відмінності мітки команди від імені директиви?
16. Для чого потрібно позначати початкову команду програми міткою?
17. Як завершується вихідний модуль програми?
18. Для чого призначена програма DOSBox?
19. Як зробити Вашу робочу папку поточної при використанні програми DOSBox?
20. Яким чином можна зберегти копію екрану в програмі DOSBox?

ЛАБОРАТОРНА РОБОТА №12

Спрощене оформлення програм. Створення виконуваних *.com файлів

Мета роботи: практичне оволодіння навичками спрощеного оформлення найпростіших програм на мові Асемблера і роботи з програмами TASM і TLINK.

Теоретичні відомості

Спрощене оформлення програм

Для створення простих програм, які містять по одному сегменту для даних і коду, а також для програм, модулі яких передбачається пов'язувати з програмами на мовах високого рівня, використовуються спрощені директиви сегментації. При цьому структура вихідного модуля дещо спрощується.

Такі програми починаються з директиви вказівки моделі пам'яті **.MODEL**, яка в найпростішому випадку має формат:

.MODEL<модель пам'яті> [, мова],

де – **< модель пам'яті >** – ім'я обраної моделі пам'яті, є обов'язковим операндом;

– **[мова]** – ім'я мови програмування, на якому написані викликаються з даного модуля процедури. Це може бути C, Pascal, Basic та інші.

Якщо програма написана повністю на Асемблері, то досить вказати тільки модель пам'яті. Можливі моделі пам'яті і їх опис наведено в таблиці 1. Як видно з цієї таблиці вибір директиви **.MODEL** визначає набір сегментів програми, розміри сегментів даних і коду, способи зв'язування сегментів і сегментних регістрів. Директиву **ASSUME**, на відміну від способу традиційного оформлення сегментів, не використовують.

Для програм на Асемблері використовується модель пам'яті Small.

Застосування директиви **.MODEL** зобов'язує програміста використовувати для опису сегментів вихідного модуля спрощені директиви опису сегментів. Кожна з моделей може використовувати директиви, представлені в таблиці 2.

Таблиця 1 – Моделі пам'яті

Ім'я	Тип коду	Тип даних	Визначення	Призначення моделі
Tiny	near	near	CS=DS=SS=dgroup Код і дані об'єднані в одну групу з ім'ям dgroup .	Для створення *.com програм
Small	near	near	CS=_text; Код займає один сегмент. Дані об'єднані в	Для невеликих і середніх програм (*.exe). Для

			одну групу з ім'ям dgroup .	програм на Асемблері.
Medium	far	near	CS=<model>_text Кілька сегментів коду. Дані об'єднані в одну групу з ім'ям dgroup .	Для великих програм з малим об'ємом даних.
Compact	near	far	CS=_text Код в одному сегменті. Дані об'єднані в одну групу з ім'ям dgroup .	Сегмент коду 64К. Обсяг даних не обмежений.
Large	far	far	CS=<model>_text Код в декількох сегментах. Дані об'єднані в одну групу з ім'ям dgroup .	Розмір коду і даних не обмежені. Для великих програм.
Huge	far	far	CS =<model>_text	Теж, що і large. Ведена для сумісності з MBP.
Tchuge	far	far	CS =<model>_text	Використовується при програмуванні на TurboC й BorlandC++.
Flat	near	near	CS =text DS = SS =flat	Використовується в середовищі OS/2.

Для спрощення посилань до імен сегментів введені кілька визначених ідентифікаторів, які можуть використовуватися в програмі. Основні з них представлені в таблиці 3.

Таблиця 2 – Спрощені директиви опису змінних

Директива	Опис
.code	Сегмента коду
.data	Сегмента ініціалізованих даних
.const	Сегмента постійних даних
.data?	Сегмента неініціалізованих даних
.stack [розмір]	Сегмента стека. Параметр задає розмір стека
.fardata [ім'я]	Сегмента ініціалізованих даних типу far
.fardata? [ім'я]	Сегмента неініціалізованих даних типу far

Таблиця 3 – Ідентифікатори, створювані директивою MODEL

Ім'я ідентифікатора	Значення: Фізична адреса сегмента
@code	Кода
@data	Даних типу near

@data?	Неініціалізованих даних типу near
@fardata	Даних типу far
@fardata?	Неініціалізованих даних типу far
@stack	Стека
@code	Коду

Для створення асемблерних програм зі спрощеними директивами сегментації можна використовувати наведений нижче шаблон:

;Forma_1 – Спрощене оформлення програм

```
.MODEL SMALL           ; модель пам'яті ближнього типу
.STACK 100h            ; визначити стек розміром 100h
.DATA                  ; відкрити сегмент даних
```

*У цьому сегменті визначаються дані
з використанням символічних імен*

```
.CODE                  ; відкрити сегмент кодів
Start:
    mov AX, @DATA      ; ініціалізувати
    mov DS, AX         ; сегментний регістр DS
```

*Тут йдуть команди, які визначаються
алгоритмом розв'язуваної задачі*

```
    mov AL, 0          ; завершити програму
    mov AH, 4Ch        ; за допомогою
int 21h                ; DOS
    END Start          ; кінець вихідного модуля.
```

Створення виконуваних модулів типу *.com

За внутрішньої організації все програми, написані для MS DOS, належать до одного з двох типів. Кожному з них відповідають імена з розширеннями ***.EXE** або ***.COM**. Програми, складені в лабораторній роботі №5 за допомогою стандартних директив сегментації, відносяться до найбільш поширеного типу **.EXE додатків**. Для таких програм характерна наявність окремих сегментів даних, стека і команд. Для адресації до цих сегментах використовуються свої сегментні адреси. Такі програми зручно розширювати за рахунок збільшення числа сегментів. Кожен сегмент в пам'яті може займати розмір **64 Кбайта**, проте в сумі цей обсяг може бути значно збільшений.

У багатьох випадках обсяг програми виявляється значно менше **64 Кбайт**. Таку програму немає ніякої необхідності складати з декількох сегментів. В цьому випадку і дані, і стек, і команди можна розмістити в одному сегменті, налаштувавши все сегментні регістри на його початок. Виконувані файли, складені за цими правилами, мають розширення ***.COM**. У вигляді

COM додатків зазвичай пишуться резидентні програми і драйвери, хоча в такому вигляді можна оформити будь-яку прикладну програму.

Для написання вихідних текстів програми типу **COM** можна використовувати наведений нижче шаблон:

```
;Forma_2 – Вихідний модуль для створення *.com додатку
.MODEL TINY                ; модель пам'яті ближнього
                           ; типу
.CODE                      ; відкрити сегмент кодів
ORG 100h                   ; відвести 256 байт під PSP
Begin: jmp Start           ; безумовний перехід на
                           ; першу команду
```

*Тут визначаються дані
з використанням символічних імен*

Start:

*Тут йдуть команди, які визначаються
алгоритмом розв'язуваної задачі*

```
    mov AL, 0              ; завершити програму
    mov AH, 4Ch            ; за допомогою
int 21h                    ; DOS
    END Begin              ; кінець вихідного модуля.
```

З вихідного тексту видно, що програма використовує спрощені директиви сегментації і містить один сегмент – **сегмент коду**. Оператор **ORG 100h** резервує 256 байт в пам'яті для **сегмента префікса програми (Program Segment Prefix – PSP)**. Цей сегмент містить таблиці і поля даних, які заповнюються і використовуються системою в процесі виконання програми.

Дані, необхідні програмі, можна оголосити перед командами, всередині них або в кінці сегмента. Однак при завантаженні програми типу **.COM** регістр лічильника команд **IP** завжди ініціалізується числом **100h**, тому слідом за оператором **ORG 100h** повинна стояти перша виконувана програмою команда. У нашому випадку це команда безумовного переходу на мітку **Start**:

Begin: jmp Start;

Після завантаження програми в пам'ять все сегментні регістри вказують на початок сегмента, в якому розташовується програма, фактично на початок **PSP**. Регістр покажчика стека **SP** автоматично завантажується числом - початковою точкою входу в стек **FFFEh**. Таким чином, незалежно від розміру програми, під неї відводиться **64 Кбайта** адресного простору. Всю нижню частину займає стек, розмір якого заздалегідь не визначений, а залежить від роботи програми. Образ програми в пам'яті показано на рис. 1.

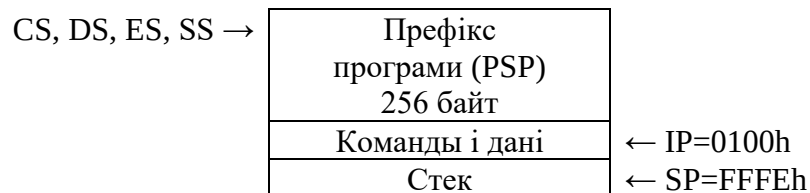


Рисунок 1 – Образ програми COM в пам'яті

З рисунка видно, що програма складається з єдиного сегмента, вміст якого майже точно відображає вміст вихідного модуля. Відмінність полягає в тому, в вихідному модулі відсутня **префікс програми (PSP)**, який з'являється в пам'яті в процесі завантаження програми.

Процес асемблювання вихідного модуля відбувається як завжди, а при виклику компоновщика **TLINK.EXE** в командному рядку слід задавати **ключ /t**, який змушує формувати виконуваний файл типу ***.COM**:
tlink /t <ім'я файлу>.

Підготовка до роботи

1. Вивчити методичні вказівки та рекомендовану літературу.
2. Підготувати відповіді на контрольні питання.

Завдання на виконання роботи

1. Використовуючи текстовий редактор, створити і відредагувати вихідний модуль програми `hello_2.asm`, (див. лаб. роб. №5) з використанням директив спрощеного оформлення програм. Зберегти вихідний текст на диску з ім'ям `hello_2s.asm`. При наборі вихідного тексту використовуйте шаблон `Forma_1`.
2. Використовуючи компілятор Турбо Асемблера `tasm.exe` створити файли `hello_2s.obj` і `hello_2s.lst`.
3. Використовуючи компоновальник `tlink.exe` створити файли `hello_2s.exe` і `hello_2s.map`.
4. Переконатися в працездатності програми `hello_2s`.
5. Переглянути в текстовому редакторі тексти всіх створених файлів і проаналізувати їх. Скласти модель розміщення сегментів програми (образ програми) `hello_2s.exe` в пам'яті ЕОМ.
6. Використовуючи текстовий редактор, створити і відредагувати вихідний модуль попередньої програми для створення виконуваного файлу типу `*.com`. Зберегти вихідний текст на диску з ім'ям `hello_2c.asm`. При наборі вихідного тексту використовуйте шаблон `Forma_2`.
7. Виконати асемблювання і компоновку створеного вихідного модуля для створення виконуваного файлу типу `*.com`.
8. Переконаватися в працездатності створеного файлу.
9. Переглянути в редакторі все створені файли і проаналізувати їх. Скласти модель розміщення сегментів програми (образ програми) `hello_2c.com` в пам'яті ЕОМ.

Вимоги до звіту

Звіт повинен містити:

- мета роботи;
- лістинги вихідного модуля і всіх файлів, створених в процесі асемблювання і компонування для програм **hello_2s** і **hello_2c** з коментарями: ***.asm**, ***.obj**, ***.lst**, ***.map**, ***.exe**, ***.com**;
- список команд, що використовуються для асемблювання і компонування;
- образи створених програм **hello_2s.exe** і **hello_2c.com** в пам'яті.

Контрольні питання

1. Команди директиви Асемблера. Формат і відмінності.
2. Яка мета сегментації пам'яті?
3. Скільки і яких сегментів може мати програма?
4. У яких випадках має сенс використовувати спрощену сегментацію?
5. Які директиви спрощеної сегментації використовуються?
6. Які моделі пам'яті використовуються при спрощеній сегментації?
7. Якими директивами описується сегмент?
8. У чому полягає ініціалізація сегментних регістрів і як вона проводиться при спрощеній сегментації?
9. Як проводиться асемблювання і компоновка програми при спрощеній сегментації?
10. Як виглядає типова форма для створення **.exe** додатків спрощеної сегментацією?
11. Яким чином розташовується програма типу **.exe** після її завантаження в пам'ять?
12. У яких випадках використовуються програми типу **.com**?
13. Як виглядає типова форма для створення **.com** додатків?
14. З яких сегментів складається вихідний модуль програми типу **.com**?
15. Яким чином розташовується програма типу **.com** після її завантаження в пам'ять?
16. Який розмір області сегмента **префікса програми (PSP)**?
17. Який розмір стека в програмах типу **.com**?
18. Як проводиться асемблювання і компоновка програм типу **.com**?
19. Що являє собою образ програми в пам'яті ЕОМ?
20. Що потрібно для того, щоб представити образ програми в пам'яті ЕОМ?

ЛАБОРАТОРНА РОБОТА №13

Програмування арифметичних операцій в Асемблері.

Мета роботи: програмування задач, що виконують арифметичні обчислення і отримання навичок налагодження програм засобами відладчика TURBO DEBUGGER.

Теоретичні відомості

Арифметичні команди в Асемблері

У мову Асемблера входять п'ять груп арифметичних команд: команди перетворення типів, команди двійковій арифметики, десятковим арифметики, допоміжні команди і інші команди з арифметичним принципом дії.

Мікропроцесор 80x86 може працювати з цілими числами **зі знаком** і з цілими **беззнаковими** числами. Цілі беззнакові числа не мають знакового розряду, тому всі його виконавчі біти відводяться під мантису числа. В числах зі знаком під знак відводиться найстарший біт двійкового числа: **0 – позитивне число; 1 – негативне**. Тому діапазон значень двійкового числа залежить від його розміру і трактування старшого біта числа. Необхідно пам'ятати, що числа зі знаком представляються в **додатковому коді**.

Таблиця 1 – Діапазон значень двійкових чисел

Розмірність поля	Ціле число без знаку	Ціле число зі знаком
байт	0...255	-128...+127
слово	0...65 535	-32 768...+32 767
подвійне слово	0...4 294 967 295	-2 147 483 648...+2 147 483 647

ADD – целочисленное сложение

Команда **ADD** здійснює додавання першого і другого операнда, при цьому початкове значення першого операнда (**dst** – приймача) втрачається, замінюючи результатом додавання. Другий операнд (**src** – джерело) не змінюється.

ADD dst, src; dst: = (dst) + (src).

Як перший операнд можна вказувати регістр (окрім сегментного) або елемент пам'яті, як другий – регістр (окрім сегментного), елемент пам'яті або безпосереднє значення, проте не допускається визначати обидва операнди одночасно як елементи пам'яті. Операнди можуть бути байтами або словами і бути числами зі знаком або без знаку. Команда впливає на прапори **OF**, **SF**, **ZF**, **AF**, **PF** і **CF**.

При додаванні беззнакових чисел, коли розмірність результату операції виходить за розрядну сітку операндів, можуть виникнути помилки з визначенням точного результату. Для індикації переповнення призначений

прапор переносу CF, який необхідно проконтролювати при виконанні операції додавання.

При операціях з знаковими числами потрібно враховувати можливе перенесення в старший знаковий розряд, так як при цьому може змінитися знак результату. Для цих цілей може допомогти аналіз **прапора переповнення OF**, так як він встановлюється в 1, якщо відбувається перенесення в старший знаковий розряд (в 7-ий або 15-ий) для позитивних чисел або з старшого знакового розряду для негативних чисел.

Спільне використання прапорів **CF** і **OF** дозволяє контролювати правильність результату при додаванні чисел зі знаком:

якщо **CF = OF**, переповнення немає;

якщо **CF $\neq$ OF**, є переповнення і потрібно коригувати результат.

SUB – віднімання цілих чисел

Команда **SUB** віднімає другий операнд з першого і поміщає результат на місце першого операнда, тобто

SUB dst, src; dst: = (dst) – (src).

Операнди аналогічні операндам команди цілочисельного додавання. Команда впливає на прапори **OF, SF, ZF, AF, PF** и **CF**.

Команди INC й DEC

Команда **INC** (інкремент) додає 1 до операнду (op), в якості якого можна вказувати регістр (окрім сегментного) або елемент пам'яті розміром як в байт, так і в слово:

INC op; op:= (op) + 1.

Не допускається використовувати як операнд безпосереднє значення. Операнд інтерпретується як число без знака. Команда впливає на прапори **OF, SF, ZF, AF** і **PF**. Команда не впливає на прапор **CF**.

Команда **DEC** (декремент) аналогічна команді **INC**, за винятком того, що вона віднімає одиницю з операнда:

DEC op; op:= (op) – 1.

Команди множення

Для множення чисел без знака призначена команда **MUL**, яка має такий вигляд:

MUL src ; **AX:= (AL)*(src)** – при множенні байтів,
 ;DX:AX:= (AX)*src – при множенні слів.

Як видно, другий операнд повинен перебувати або в регістрі-акумуляторі **AL** (у разі множення на байт), або в регістрі-акумуляторі **AX** (в разі множення на слово). Після виконання операції з однобайтовими числами, 16-й бітовий результат записується в регістр-акумулятор **AX**; для двобайтових чисел двір довжиною в 32 біта формується в парі регістрів **DX:AX** (в **DX** – старша частина, в **AX** – молодша). Попереднє вміст регістра **DX** затирається

Якщо вміст регістра **АН** після однобайтового множення або вміст регістра **ДХ** після двухбайтового множення не рівні 0, прапори **CF** і **OF** встановлюються в 1. В іншому випадку обидва прапора скидаються в 0.

Як операнд-співмножник команди **MUL** можна вказувати регістр (окрім сегментного) або елемент пам'яті. Не допускається множення на безпосереднє значення.

Для множення чисел зі знаком призначена команда
IMUL src.

Ця команда виконується так само, як і команда **MUL**. Відмінною особливістю команди **IMUL** є тільки формування знаку. Якщо результат малий і вміщується в одному регістрі (тобто якщо **CF = OF = 0**), то вміст іншого регістра (старшої частини) є розширенням знаку – все його біти рівні старшому біту (знакової розряди) молодшої частини результату. В іншому випадку (якщо **CF = OF = 1**) знаком результату є знаковий біт старшої частини результату, а знаковий біт молодшої частини є значущим бітом двійкового коду результату.

Команди ділення

Команди ділення для знакових і беззнакових операндів **DIV** і **IDIV** виконують цілочисельне ділення, формуючи ціле приватне і цілий залишок. Формат команди:

DIV src ; $AL := \text{quot}((AX)/(src));$ частное при діленні на байт.
 $AH := \text{rem}((AX)/(src));$ залишок при діленні на байт.
 $AX := \text{quot}((DX:AX)/(src));$ частное при розподілі на
 ; слово
 $DX := \text{rem}((DX:AX)/(src));$ залишок при діленні на
 ; слово.

При цьому ділене повинно знаходитися в регістрах **AX** (в разі поділу на байт) або **DX:AX** (в разі поділу на слово). **Розмір діленого повинен бути в два рази більше розмірів дільника і залишку.**

Після виконання операції з однобайтовими дільниками, частное записується в регістр **AL**, залишок – в регістр **AH**; для двобайтових дільників – частное в **AX**, залишок в **DX**.

Якщо дільник дорівнює 0, або якщо частное не поміщається в призначений регістр, збуджується переривання з вектором 0 (ділення на 0).

Команди не впливають на прапори процесора.

При діленні цілих чисел зі знаком (IDIV), і частное, і залишок розглядаються як числа зі знаком, причому знак залишку дорівнює знаку діленого.

Команди узгодження розмірів операндів

При виконанні арифметичних операцій часто використовуються команди перетворення байта в слово або слова в подвійне слово. Ці команди,

що виконують перетворення над операндом, знаходяться в регістрі-акумуляторі:

CBW – перетворити байт в AL в слово в AX. При цьому старший байт **AX** заповнюється значенням старшого розряду регістра **AL**:

CBW; AX:=D₇D₇D₇D₇D₇D₇D₇D₇(AL)
; (AL):= D₇D₆D₅D₄D₃D₂D₁D₀.

CWD – перетворити слово в AX в подвійне слово в DX: AX. При цьому регістр DX заповнюється значенням старшого розряду регістра **AX**:

CWD; DX:=D₁₅D₁₅D₁₅...D₁₅
; (AX):=D₁₅D₁₄D₁₃...D₄D₃D₂D₁D₀.

Робота з TURBO DEBUGGER

Якщо програма скомпонована в режимі /v, то після її завантаження отладчиком, відкривається вікно **Module**. Символ <стрілка> показує на що підлягає виконанню команду. Клавішею **F2** можна розставляти і знімати точки зупинки (пастки) (**Breakpoints**) в тому рядку, де розташований курсор для зупинення виконання програми.

Вікно **Inspect** можна відкрити з локального меню вікна **Module (alt-F10)**. При цьому відладчик запитує ім'я підлягають контролю змінної або регістра. Контролювати стану змінних можна також у вікнах **Variables** і **Watches**, що викликаються з пункту **View** головного меню.

Variables дозволяє спостерігати всі змінні, доступні в місці зупинки програми. У локальному вікні пункт **Inspect** дає доступ до повної інформації про тип, значення та адресу зберігання виділеної змінної. Окремі змінні програміст може задати для аналізу в вікні **Watches**. Для приміщення змінної в це вікно слід підвести курсор до ідентифікатора змінної і натиснути **Ctrl +W**. Для видалення змінної з вікна можна скористатися локальним меню, або клавішею **Delete**.

Підготовка до роботи

1. Вивчити методичні вказівки та рекомендовану літературу.
2. Підготувати відповіді на контрольні питання.

Завдання на виконання роботи

1. Використовуючи текстовий редактор, створити і відредагувати вихідний модуль програми **Prog_4.asm**, яка обчислює значення **X** відповідно до варіанта завдання. Номер функції і значення змінних **A**, **B** і **C** взяти з таблиці
2. Значення змінної **D** береться рівним останній цифрі номера залікової книжки. Після виконання операції ділення, в подальших операціях враховувати тільки частное.

;Program_4 – Арифметичні операції, варіант ...

Data SEGMENT

A DB 1

B DB 2

;Відкрити сегмент даних

;Ініціалізувати

;змінні A, B, C, D, X

C DB 3	
D DB 4	
X DW ?	
Data ENDS	;Закрити сегмент даних
Ourstack SEGMENT Stack	;Відкрити сегмент стека
DB 100h DUP (?)	;Відвести під стек 256 байт
Ourstack ENDS	;Закрити сегмент стека
ASSUME CS:Code, DS:Data, SS:Ourstack	;Призначити сегментні
	;реєстри
Code SEGMENT	;Відкрити сегмент кодів
Start: mov AX, Data	;Ініціалізувати
mov DS, AX	;сегментний реєстр DS
xor AX, AX	;Очистити реєстр AX

*Тут повинні бути команди обчислення
арифметичного виразу*

mov AX, 4C00h	;Завершити програму
int 21h	;за допомогою DOS
Code ENDS	;Закрити сегмент кодів
END Start	;Кінець вихідного модуля.

Таблиця 2 – Варіанти завдань

№ варіанту	Функція	Дані		
		A	B	C
1	$X = \frac{2 * A + B * D}{C - 3}$	64h	14h	-4
2	$X = \frac{D * C}{2 * A + B}$	16h	-50	1Bh
3	$X = \left(1 + \frac{A}{5}\right) * B - C * D$	150	111b	48h
4	$X = \frac{A^2 + D}{C - D}$	15	150h	5
5	$X = (48 + 3 * A) - \frac{B}{C} * D$	5Ah	55h	11h
6	$X = \frac{(B - 25)^2}{A + 1} + (B + D)^2$	-5	31	
7	$X = (A + B) * (C - 4000) * (D + 212)$	A1h	-150	FB0h
8	$X = (A * B - C * D)^2$	Fh	14	10h
9	$X = \frac{A^2 + B^2}{D - C}$	7	12	-15
10	$X = \frac{(B - C) * A^2}{D - 12}$	5	E2h	225
11	$X = \frac{300 - D + B * C}{A}$	8	26h	-10

12	$X = \frac{65528 - A * B}{(D + C)^2}$	BFh	14h	2
13	$X = \frac{A * (B + 1)}{C} - D$	32	Fh	80
14	$X = 3 * (A - B) + \frac{D}{C}$	99h	D9h	155
15	$X = \frac{(-1) * (D + 1)}{A + B * C}$	Ch	4	9
16	$X = \frac{2 * B + A * D}{C^2 - 3}$	32h	5Bh	3
17	$X = \frac{D * A}{2 * B + C}$	111b	-30	1Ch
18	$X = \left(1 + \frac{B}{5}\right) * A + C * D$	100	111b	4Bh
19	$X = \frac{A + D}{C^2 - D}$	2Bh	15	5Fh
20	$X = (128 + 3 * C) - \frac{A}{B} * D$	1Eh	25h	2Ch

2. Використовуючи компілятор Турбо Асемблера **tasm.exe** і компоновальник **link.exe** з відповідними ключами, створити файл **prog_4.exe**.

3. Завантажте програму в відладчик і в вікні **CPU** зробіть трасування програми (послідовне виконання) натисканням клавіші **F8**. На кожному кроці контролюйте вміст регістрів і прапорів. Запишіть обчислене значення приватного і залишку. Переконайтеся в правильності обчислень. Результати покрокового виконання зведіть в таблицю 3 і зробіть по ним відповідні висновки.

4. Визначте найдовшу і найкоротшу команди програми **prog_4.exe**.

5. Визначте початкові і кінцеві адреси сегментів коду, даних і стека складеної програми **prog_4.exe**. Обчисліть довжину сегментів зазначеної програми в байтах. Складіть і намалюйте образ програми в пам'яті ЕОМ.

Таблиця 3 – Результати виконання програми

Варіант ...					
№ рядка	Команда Асемблера	Машинний код	Довжина машинного коду, байт	Логічна адреса в пам'яті	Стан регістрів і прапорів
1					AX=..., BX=..., CX=..., DX=..., SP=..., BP=..., SI=..., DI=..., IP=..., DS=..., SS=..., CS=..., ES=...; CF=..., ZF=..., SF=...,

					OF=..., PF=..., AF=...
2					AX=..., BX=..., CX=..., DX=..., SP=..., BP=..., SI=..., DI=..., IP=..., DS=..., SS=..., CS=..., ES=...; CF=..., ZF=..., SF=..., OF=..., PF=..., AF=...
Значення змінної X: частное=... залишок=...					

Вимоги до звіту

Звіт повинен містити:

- мета роботи;
- лістинги програми **Prog_4** з коментарями;
- таблицю 3 з результатами дослідження налагодження програми;
- рисунок образу програми **Prog_4** в пам'яті ЕОМ (див. п.4.завдання);
- дані файлів ***.map** і обчислені початкові і кінцеві адреси сегментів програми і їх довжини (див. п.4. завдання).

Контрольні питання

1. Формат команди «додати», її операнди.
2. Формат команди «відняти», її операнди.
3. Формат команди «помножити», її операнди.
4. Формат команди «ділити», її операнди.
5. Який діапазон беззнакових чисел допустимий в програмах 16-ті розрядного мікропроцесора?
6. Який діапазон чисел зі знаком допустимо в програмах 16-ті розрядного мікропроцесора?
7. Яку інформацію містять арифметичні прапори операцій?
8. Які прапори встановлюються при виконанні команд «додати» і «відняти».
9. Які прапори встановлюються при виконанні команд «помножити» і «ділити».
10. Як виконати додавання (віднімання) двох операндів, що в пам'яті?
11. Як виконати множення двох операндів, що знаходяться в пам'яті?
12. Як виконати поділ двох операндів, що в пам'яті?

13. З числами якої системи числення може працювати Асемблер?
14. Яким чином можна проконтролювати значення змінних при налагодженні програми?
15. Як можна контролювати область даних програми, що завантажена в пам'ять?
16. Як можна контролювати область стека програми, що завантажена в пам'ять?
17. Знайдіть помилки в наведених нижче командах:
MOV AL, E4h; ADD 64, BL; MUL 3Fh; MOV DS, 3F3Fh.

ЛАБОРАТОРНА РОБОТА №14

Тема: Показники ефективності обчислювальних машин.

Мета: Ознайомитись зі способами оцінки ефективності різних комп'ютерних систем.

Теоретична частина

Обчислювальну машину можна визначити множиною показників, що характеризують окремі її властивості. Виникає завдання введення міри для оцінки ступеня пристосованості ОМ до виконання покладених на неї функцій – міри ефективності.

Ефективність визначає ступінь відповідності ОМ своєму призначенню. Вона вимірюється або кількістю витрат, необхідних для отримання певного результату, або результатом, отриманим при певних витратах. Провести порівняльний аналіз ефективності декількох ОМ, ухвалити рішення на використання конкретної машини дозволяє критерій ефективності.

Критерій ефективності – це правило, що служить для порівняльної оцінки якості варіантів ОМ. Критерій ефективності можна назвати правилом переваги порівнюваних варіантів.

Будуються критерії ефективності на основі окремих показників ефективності (показників якості). Спосіб зв'язку між окремими показниками визначає вид критерію ефективності.

Як основні показники ОМ зазвичай розглядають: ємність пам'яті, швидкодію та продуктивність, вартість і надійність. Зупинимось тільки на показниках швидкодії та продуктивності, що зазвичай представляють основний інтерес для користувачів.

Швидкодія. Доцільно розглядати два види швидкодії: номінальну і середню.

Номінальна швидкодія характеризує можливості ОМ під час виконання стандартної операції. Як стандартну зазвичай вибирають коротку операцію додавання. Якщо позначити через $\tau_{\text{доп}}$ час додавання, то номінальна швидкодія визначиться з виразу

$$V_{\text{ном}} = \frac{1}{\tau_{\text{доп}}} \left[\frac{\text{оп}}{c} \right].$$

Середню швидкодію характеризує швидкість обчислень при виконанні еталонного алгоритму або деякого класу алгоритмів. Величина середньої швидкодії залежить як від параметрів ОМ, так і від параметрів алгоритму і визначається співвідношенням

$$V_{\text{сер}} = \frac{N}{T_e},$$

де T_e – час виконання еталонного алгоритму;

N – кількість операцій, що містяться в еталонному алгоритмі.

Позначимо через n_i число операцій i -го типу; l – кількість типів операцій в ($i = 1, 2, \dots, l$); τ_i – час виконання операції i -го типу.

Час виконання еталонного алгоритму розраховується за формулою

$$T_e = \sum_{i=1}^l \tau_i n_i.$$

Підставивши останнє співвідношення у вираз для $V_{сер}$, отримаємо

$$V_{сер} = \frac{N}{\sum_{i=1}^l \tau_i n_i}.$$

Звідки отримаємо

$$V_{сер} = \frac{1}{\sum_{i=1}^l \frac{n_i}{N} \tau_i}.$$

Позначивши частоту появи операції i -го типу в останньому виразу через $q_i = \frac{n_i}{N}$, запишемо остаточну формулу для розрахунку середньої швидкодії

$$V_{сер} = \frac{1}{\sum_{i=1}^l q_i \tau_i} \left[\frac{on}{c} \right]. \quad (1)$$

У виразі (6.6) вектор $\{\tau_1, \tau_2, \dots, \tau_l\}$ характеризує систему команд ОМ, а вектор $\{q_1, q_2, \dots, q_l\}$, що називається частотним вектором операцій, характеризує алгоритм.

Очевидно, що для ефективної реалізації алгоритму необхідно прагнути до збільшення $V_{сер}$. Якщо $V_{ном}$ головним чином відштовхується від швидкодії елементної бази, то $V_{сер}$ дуже сильно залежить від оптимальності вибору команд ОМ.

Формула (1) дозволяє визначити середню швидкість машини під час реалізації одного алгоритму. Розглянемо більш загальний випадок, коли повний алгоритм складається з декількох окремих, періодично повторюваних алгоритмів. Середня швидкість під час рішення повної задачі розраховується за формулою

$$V_{сер}^n = \frac{1}{\sum_{j=1}^m \sum_{i=1}^l \beta_j q_{ji} \tau_i} \left[\frac{on}{c} \right],$$

де m – кількість окремих алгоритмів;

β_j – частота появи операцій j -го окремого алгоритму в повному алгоритмі;

q_{ji} – частота операцій i -го типу в j -му окремому алгоритмі.

Позначимо через N_j і T_j – кількість операцій і період повторення j -го окремого алгоритму; $T_{\max} = \max_j (T_1, \dots, T_j, \dots, T_n)$ – період повторення повного

алгоритму; $a_j = T_{max}/T_j$ – циклічність включення j -го окремого алгоритму в повному алгоритмі.

Тоді за час T_{max} в ОМ буде виконано $N_{max} = \sum_{j=1}^m a_j N_j$ операцій, а частоту появи операцій j -го окремого алгоритму в повному алгоритмі можна визначити з виразу

$$\beta_j = \frac{\alpha_j N_j}{N_{max}}. \quad (2)$$

Для розрахунку за формулами (1, 2) необхідно знати параметри ОМ, представлені вектором $\{\tau_1, \tau_2, \dots, \tau_l\}$, параметри кожного j -го окремого алгоритму – вектор $\{q_{j1}, q_{j2}, \dots, q_{jl}\}$ і параметри повного алгоритму – вектор $\{\beta_1, \beta_2, \dots, \beta_m\}$.

Продуктивність ОМ оцінюється кількістю еталонних алгоритмів, що виконуються в одиницю часу:

$$P = \frac{1}{T_e} \left[\frac{\text{задач}}{c} \right].$$

Продуктивність під час виконання повного алгоритму оцінюється за формулою

$$P_n = \frac{1}{\sum_{j=1}^m \sum_{i=1}^l \alpha_j N_j q_{ij} \tau_i} \left[\frac{\text{задач}}{c} \right].$$

Продуктивність є більш універсальним показником, ніж середня швидкодія, оскільки в явному вигляді залежить від порядку проходження завдань через ОМ.

На практиці зазвичай використовують простіші вирази. Для оцінки часу виконання програми з динамічною кількістю команд N використовують вираз

$$T = \frac{N \cdot K}{F},$$

де K – середня кількість тактів, що витрачаються на вибірку і виконання однієї команди;

F – тактова частота процесора.

Для оцінки продуктивності використовують *пропускну здатність* процесора – кількість команд, що виконуються за одну секунду. У разі послідовного виконання команд пропускну здатність P_s визначається за формулою

$$P_s = \frac{F}{K}.$$

Для *конвеєрного* процесора пропускну здатність сама по собі ще не є свідомством хорошої продуктивності. Реальною мірою продуктивності комп'ютера є загальний час виконання програми. В загальному випадку n -ступінчастий конвеєр потенційно підвищує продуктивність в n разів.

Виходить, що чим більше значення n , тим вище продуктивність процесора. Проте реальна продуктивність конвеєрного процесора нижча. Кожного разу, коли відбувається зупинка конвеєра, потік команд, що обробляються процесором, різко скорочується. Таким чином, продуктивність конвеєра багато в чому залежить від таких чинників, як накладні витрати переходів і промахів у разі звернення до кеш-пам'яті.

Скорочення накладних витрат можна добитися за рахунок введення вторинного кеша, який розміщується між первинним кешем, інтегрованим у мікро- схему процесора, і основною пам'яттю. Крім того, в комп'ютерах використовується оптимізуючий компілятор, який по можливості віддаляє залежні один від одного команди, поміщаючи між ними інші. А також, якщо в процесорі існує черга команд, промах під час вибірки команди може мати значно менші негативні наслідки, оскільки процесор продовжує виконання команд, що знаходяться в черзі.

Для підвищення продуктивності конвеєрного процесора здавалося б доцільним розділяти процес виконання команд на якомога більшу кількість ступенів. Проте чим більше ступенів, тим вище вірогідність зупинок конвеєра. Це пов'язано з тим, що більшість команд виконуються паралельно, в зв'язку з чим навіть значно віддалені одна від одної команди, між якими є залежності, можуть викликати зупинки конвеєра, що, у свою чергу, приводить до зростання наклад-них витрат переходів. Через перераховані причини підвищення продуктивності за рахунок збільшення кількості ступенів виявляється не таким уже істотним.

Ще один важливий чинник, що впливає на продуктивність, – внутрішні затримки під час виконання процесором базових операцій. Особливої уваги заслуговує затримка в АЛП. У багатьох процесорах тактова частота підбирається так, щоб операція складання в АЛП здійснювалася за один такт. Решта операцій розділяється на кроки, що займають той же час, що і операція складання. При цьому АЛП може мати конвеєрну організацію.

У багатьох конвеєрних процесорах використовується від чотирьох до шести ступенів. У ряді процесорів процедура виконання команди ділиться на менші кроки, використовується більша кількість ступенів конвеєра і вища тактова частота. Наприклад, у процесорі UltraSPARC II застосовується 9-ступінчастий конвеєр, а в процесорі Intel Pentium Pro – 12-ступінчастий, Intel Pentium 4 містить 20-ступінчастий конвеєр і функціонує на тактовій частоті від 1,3 до 1,5 ГГц. Для прискорення роботи на кожному такті виконуються два ступені конвеєра.

Практична частина

Завдання 1. Заповніть таблицю, в якій необхідно вказати відмінності характеристики вибраних 3 (трьох) мікропроцесорів.

Характеристики мікропроцесорів

Мікропроцесор 1		
1	Кількість ядер	

2	Кількість потоків	
3	Розрядність	
4	Тактова частота	
	Мікропроцесор 2	
	...	

Завдання 2. Оцініть час виконання програми T та пропускну здатність процесора P_s .

Завдання 3. Проаналізуйте залежність отриманих параметрів від характеристик мікропроцесорів.

Контрольні питання

1. Що таке *номінальна швидкодія* комп'ютерної системи? Від чого вона залежить?
2. Що таке *середня швидкодія* комп'ютерної системи? Від чого вона залежить?
3. Як оцініть продуктивність комп'ютерної системи?