ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра «Спеціалізовані комп'ютерні системи»

В.В. Нарожний, О.В. Головко

СУЧАСНІ ПЕОМ: АПАРАТНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Курс лекцій

Частина 1

Харків – 2013

УДК 004.41-43(075)

Нарожний В.В., Головко О.В. Сучасні ПЕОМ: апаратне та програмне забезпечення: Курс лекцій. – Харків: УкрДАЗТ, 2013. – Ч.1. – 94 с.

У курсі лекцій викладено теоретичні відомості зі схемотехніки, архітектури, будови та функціонування сучасних обчислювальних систем. Для поліпшення якості знань наведено ряд прикладів мовою програмування Assembler, що дозволяють поглянути на будову та функціонування сучасних обчислювальних систем з максимально глибоким дослідженням.

Курс лекцій призначений для студентів, що навчаються за спеціальностями 092507 «Спеціалізовані комп'ютерні системи», 091503 «Комп'ютерні інформаційно-управляючі системи», спеціалізаціями експлуатаційної спрямованості в рамках цих спеціальностей, а також для інших спеціальностей відповідних напрямків навчання. Курс лекцій містить лекційний та додатковий довідковий матеріал з дисциплін «Мікропроцесорна техніка», «Комп'ютерна схемотехніка», «Архітектура персонального комп'ютера» та «Периферійні пристрої обчислювальної техніки». Курс лекцій може застосовуватись також при виконанні курсових проектів і робіт, розрахунково-графічних робіт, розділів дипломних проектів, присвячених обчислювальній техніці.

Іл. 2, бібліогр.: 7 назв.

Рецензенти:

професори Г.Ф. Кривуля (ХНУРЕ),
С.В. Лістровий (УкрДАЗТ)

В.В. Нарожний, О.В. Головко

СУЧАСНІ ПЕОМ: АПАРАТНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Курс лекцій

Частина 1

Відповідальний за випуск Нарожний В.В.

Редактор Еткало О.О.

Підписано до друку 31.03.11 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 3,5. Тираж 60. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 2874 від 12.06.2007 р.

ЗМІСТ

ВСТУП

За останні 30 років технічний прогрес сягнув великого стрибка. Особливо це стосується значного поширення цифрових електронно-обчислювальних систем майже в усі види діяльності людини. Усе більше пристроїв має у своєму складі цифрові електронно-обчислювальні системи (ЦЕОС). Усе менше людина може жити без обслуговування цифровою електронно-обчислювальною технікою іноді майже не підозрюючи про це. Цифрова електронно-обчислювальна техніка є у багатьох домашніх приладах та на виробництві. Усюди відбувається модернізація залізнично-дорожнього транспорту та станцій, метою якої є заміна пристроїв без ЦЕОС на сучасні комп'ютеризовані системи, і це стосується всіх напрямків розвитку цивілізації. ЦЕОС можна зустріти в багатьох сучасних приладах, а саме: в **телефон**ах, **пральних машин**ах та ін.; вони відповідають за роботу **двигунів** і систем **гальмування** сучасних **автомобілів**, з їх допомогою створюються **системи контролю** і **системи збору інформації**. Шлях всесвітньої комп'ютеризації і далі буде продовжуватись та лише збільшувати темпи. Тому вивчення принципів роботи ЦЕОС є найважливішим напрямком роботи вищої освіти.

У курсі лекцій розглядається робота найпопулярнішої універсальної ЦЕОС – **персонального комп'ютера**. Центральним елементом кожної ЦЕОС є центральний процесор. Уже багато років термін «центральний процесор» є близьким терміну «мікропроцесор». Це сталося, коли майже всі електронні схеми центрального процесора було вшито на кристалі однієї мікросхеми, яку назвали мікропроцесором. Мета курсу лекцій – навчити студентів розуміти роботу всіх вузлів персонального комп'ютера та деяких зовнішніх периферійних пристроїв. Кожний розділ містить опис фрагментів лабораторних робіт, де застосовано мову програмування Assembler за допомогою безкоштовних програмних пакетів MyASM та FAR.

1 Еволюція комп'ютерних технологій

У ході еволюції комп'ютерних технологій були розроблені сотні різних комп'ютерів. Багато з них давно забуті, у той час, як вплив інших на сучасні ідеї виявився дуже значним.

1.1 Нульове покоління комп’ютерів – механічні комп’ютери (1642-1945 роки)

1642 рік – Блез Паскаль створив обчислювальну машину в 19 років для свого батька (збирач податків). Машина могла виконувати операції додавання та віднімання.

1672 рік – Р. В. Лейбніц побудував іншу машину, яка могла ще виконувати й операції множення та ділення.

1822 рік – Чарльз Беббідж побудував різницеву машину для морської навігації, яка могла лише додавати та віднімати, але ця машина видавала результат, видавлений на мідній дощечці сталевим штампом. Пізніше він розробив аналітичну машину, яка містила багато елементів сучасного комп'ютера:

1) пристрій, що запам'ятовує 1000 обчислювальних слів по 50 десяткових розрядів кожне;

2) обчислювальний пристрій, який приймав операнди з пам'яті, потім виконував одну з чотирьох дій і повертав отриманий результат назад у пам'ять;

3) пристрій введення для зчитування перфокарт;

4) пристрій виведення (перфоруючий друкуючий пристрій).

Таким чином, з'явилася можливість проводити не прості операції, а створювати деяку послідовність цих операцій – **алгоритми**.

Для розроблення алгоритмів Беббідж найняв молоду жінку Аду Августу Ловлейс, яка стала першим програмістом у світі, і на честь неї була названа мова програмування – Ада.

На жаль, ідеї Беббіджа випереджали час і можливості. Йому так і не вдалося повністю реалізувати задуману ним машину.

1930 рік – Конрад Зус сконструював декілька автоматичних рахункових машин з використанням електромагнітних реле.

З початком Другої світової війни уряди різних країн почали розробляти обчислювальні машини, усвідомлюючи їхню стратегічну роль під час війни. Збільшення фінансування значною мірою стимулювало розвиток обчислювальної техніки. У 1941 році німецький інженер Конрад Цузе розробив обчислювальну машину Z2, що виконувала розрахунки, необхідні при проектуванні літаків і балістичних снарядів. У 1943 році англійські інженери завершили створення обчислювальної машини «Колос» для дешифрування повідомлень німецької армії. Проте ці пристрої не були універсальними обчислювальними машинами, вони призначалися для вирішення конкретних задач.

У 1944 році американський інженер Говард Ейкен при підтримці фірми IBM сконструював комп'ютер «Марк I» для виконання балістичних розрахунків. Цей комп'ютер за площею займав приблизно половину футбольного поля та містив більше 600 км кабелю. У комп'ютері «Марк I» використовувався принцип електромеханічного реле, що полягав у тому, що електромагнітні сигнали переміщали механічні частини. «Марк I» був досить повільною машиною: для того, щоб виконати одне обчислення, було потрібно 3-5 с. Проте незважаючи на величезні розміри та повільність, «Марк I» став більш універсальним обчислювальним пристроєм, ніж машина Цузе або «Колос». «Марк I» управлявся за допомогою програми, яка вводилася з перфострічки. Це дало можливість, змінюючи програму, що вводиться, розв’язувати досить широкий клас математичних задач.

Таким чином у 1944 році Говард Айкен, прочитавши роботи Беббіджа, створив на реле такий же комп'ютер «МАРК I». Пізніше Він створив ще один комп'ютер «МАРК II», але почалася ера електроніки, а релейні комп'ютери застаріли.

1.2 Перше покоління – електронні лампи (1945-1955 рр.)

У 1946 році американські вчені Джон Мокнулі та Дж. Преспер Еккерт сконструювали електронний обчислювальний інтегратор і калькулятор (ЕНІАК) - комп'ютер, у якому електромеханічні реле були замінені на електронні вакуумні

лампи. Застосування вакуумних ламп дозволило збільшити швидкість роботи ЕНІАК у 1000 разів у порівнянні з «Марк І». ЕНІАК складався з 18000 вакуумних ламп, 70000 резисторів, 5 мільйонів сполучних спайок і споживав 160 кВт електричної енергії, що на ті часи було достатньо для освітлення невеликого міста. Саме ЕНІАК став працюючим прообразом сучасного комп'ютера.

По-перше, ЕНІАК був заснований на повністю цифровому принципі обробки інформації. По-друге, ЕНІАК став дійсно універсальною обчислювальною машиною, він використовувався для розрахунку балістичних таблиць, прогнозу погоди, розрахунків у галузі атомної енергетики, аеродинаміки, вивчення космосу.

Наступний важливий крок у вдосконаленні обчислювальної техніки зробив американський математик Джон фон Нейман. Ранні обчислювальні машини могли виконувати лише команди, що надходять ззовні, причому команди виконувалися по черзі. Хоча використання перфокарт дозволяло спростити процес введення команд, проте часто процес настройки обчислювальної машини і введення команд займав більше часу, ніж власне вирішення поставленої задачі. Фон Нейман запропонував увести до складу комп'ютера для зберігання послідовності команд і даних спеціальний пристрій – пам'ять. Крім того, Джон фон Нейман запропонував реалізувати в комп'ютері спроможність передачі управління від однієї програми до іншої. Спроможність зберігати в пам'яті комп'ютера різні набори команд (програми), припиняти виконання однієї програми та передавати управління іншій, а потім повертатися до початкової значно розширювала можливості програмування для обчислювальних машин. Іншою ключовою ідеєю, запропонованою фон Нейманом, став процесор (центральний обчислювальний пристрій), який повинен був управляти всіма функціями комп'ютера. У 1945 році Джон фон Нейман підготував звіт, у якому визначив такі основні принципи роботи та елементи архітектури комп'ютера:

1) комп'ютер складається з процесора (центрального оброблювального пристрою), пам'яті і зовнішніх пристроїв;

2) єдиним джерелом активності (не беручі до уваги стартове або аварійне втручання людини) в комп'ютері є процесор, що керується програмою, яка знаходиться в пам'яті;

3) пам'ять комп'ютера складається з комірок, кожна з яких має свою унікальну адресу, кожна комірка зберігає команду програми або одиницю обробленої інформації, команда й інформація мають однакове подання;

4) у будь-який момент процесор виконує одну команду програми, адреса якої знаходиться в спеціальному регістрі процесора – лічильнику команд;

5) обробка інформації відбувається лише в регістрах процесора, а інформація у процесор надходить з пам'яті або від зовнішнього пристрою;

6) у кожній команді програми зашифровано такі розпорядження: з яких комірок узяти оброблювану інформацію; які операції зробити з інформацією; у які елементи пам'яті спрямувати результат; як змінити вміст лічильника команд, щоб знати, звідки взяти наступну команду для виконання;

7) процесор виконує програму «команда за командою» відповідно до зміни лічильника команд до тих пір, поки не отримає команду зупинитися.

Надалі архітектура фон Неймана трохи змінювалася і доповнювалася, але початкові принципи управління роботою комп'ютера за допомогою програм, що зберігаються в пам'яті, залишилися незмінними. Переважна більшість сучасних комп'ютерів побудована саме за архітектурою фон Неймана. У 1951 році був створений перший комп'ютер, призначений для комерційного використання, – УНІВАК (універсальний автоматичний комп'ютер), у якому були реалізовані всі принципи архітектури фон Неймана. У 1952 році за допомогою УНІВАК було обчислено результат виборів президента США, що майже демонізувало саме поняття комп’ютера.

Роботи з створення обчислювальних машин велися і в СРСР. Так, у 1950 році в Інституті електроніки Академії наук України під керівництвом академіка Сергія Олексійовича Лебедева була розроблена та введена в експлуатацію МЕРМ (мала електронна рахункова машина). МЕРМ стала першою вітчизняною універсальною ламповою обчислювальною машиною в СРСР. У 1952-1953 роках МЕРМ залишалася найбільш швидкодійною (50 операцій за секунду) обчислювальною машиною в Європі. Принципи побудови МЕРМ

були розроблені С.А. Лебедевим незалежно від аналогічних робіт за кордоном.

У комп'ютерах першого покоління використовувалася машинна мова – спосіб запису програм, що припускає їхнє безпосереднє виконання на комп'ютері. Програма на машинній мові являє собою послідовність машинних команд, допустимих для даного комп'ютера. Процесор безпосередньо сприймає і виконує команди, висловлені у вигляді двійкових кодів. Для кожного комп'ютера існувала своя власна машинна мова. Це також обмежувало галузь застосування комп'ютерів першого покоління.

Поява першого покоління комп'ютерів стала можливою завдяки трьом технічним інноваціям: електронним вакуумним лампам, цифровому кодуванню інформації і створенню пристроїв штучної пам'яті на електростатичних трубках. Комп'ютери першого покоління мали невисоку продуктивність (до декількох тисяч операцій за секунду). У комп'ютерах першого покоління використовувалася архітектура фон Неймана. Засоби програмування і програмного забезпечення ще не були розвинуті, використовувалася низькорівнева машинна мова. Галузь застосування комп'ютерів була обмежена.

1.3 Друге покоління комп'ютерів (1956-1963 роки)

Електронні вакуумні лампи виділяли велику кількість тепла, поглинали багато електричної енергії, були громіздкими, дорогими та ненадійними. Комп'ютери першого покоління, побудовані на вакуумних лампах, мали низьку швидкодію і невисоку надійність.

У 1947 році співробітники американської компанії «Белл» Уїльям Шоклі, Джон Бардін та Уолтер Бреттейн винайшли транзистор. Транзистори виконували ті ж функції, що й електронні лампи, але використовували електричні властивості напівпровідників. У порівнянні з вакуумними трубками транзистори займали у 200 разів менше місця та споживали у 100 разів менше електроенергії. У той же час з'являються нові пристрої для організації пам'яті комп'ютерів – феритові

сердечники. З винаходом транзистора та використанням нових технологій зберігання даних у пам'яті з'явилася можливість значно зменшити розміри комп'ютерів, зробити їх більш швидкими та надійними, а також значно збільшити місткість пам'яті комп'ютерів.

У 1954 році компанія Texas Instruments оголосила про початок серійного виробництва транзисторів, а у 1956 році вчені Массачусетського технологічного інституту створили перший повністю побудований на транзисторах комп'ютер TX-O.

Машинна мова, що застосовувалася у першому поколінні комп'ютерів, була надто незручною для сприйняття людиною. Числове кодування операцій, адреса комірок та оброблюваної інформації, залежність виду програми від її місця в пам'яті не давали можливості стежити за значенням програми. Для подолання цих незручностей була придумана мова – assembler. Для запису кодів операцій та оброблюваної інформації в assembler використовуються стандартні позначення, що дозволяють записувати числа та текст у загальноприйнятій формі, а для кодів команд – прийнято мнемонічні позначення. Для позначення величин, розміщуваних у пам'яті, можна застосовувати будь-які імена, що відповідають значенню програми. Після введення програми assembler сам замінює символічні імена на адреси пам'яті, а символічні коди команд на числові машинні коди. Використання assembler зробило процес написання програм більш наочним.

У кінці 50-х – початку 60-х років комп'ютери другого покоління почали інтенсивно використовуватися державними організаціями та великими компаніями для вирішення різних задач. До 1965 року велика частина великих компаній обробляла фінансову інформацію за допомогою комп'ютерів. Поступово вони набували рис сучасного комп'ютера. Так, у цей період були сконструйовані такі пристрої, як графопобудовник (плотер) і принтер, носії інформації на магнітній стрічці і магнітних дисках та ін.

Розширення галузі застосування комп'ютерів потребувало створення нових технологій програмування. Програмне забезпечення, написане на мові assembler для одного комп'ютера, було непридатне для роботи на іншому комп'ютері. З цієї

причини, зокрема, не вдавалося створити стандартну операційну систему – основну програму, що керує комп'ютером, оскільки кожний виробник комп'ютерів розробляв свою операційну систему на своєму assembler.

Фахівці, що використовують у своїй діяльності комп'ютери, незабаром відчули потребу в більш структурованих мовах, які б спрощували процес програмування, а також дозволяли переносити програми з одного комп'ютера на інший. Подібні мови програмування отримали назву мов високого рівня. Для їхнього використання необхідно мати компілятор (або інтерпретатор), тобто програму, яка перетворить оператори мови в машинну мову даного комп'ютера.

Однією з перших мов програмування високого рівня став Фортран (FORTRAN – FORMULA TRANSLATION), який призначався для математичних алгоритмів і став надзвичайно популярний серед учених. На Фортрані можна писати великі програми, розбиваючи задачу на декілька частин (підпрограм), які програмуються окремо, а потім об'єднуються в єдине ціле. Оскільки Фортран призначений в основному для обчислень, у ньому були відсутні розвинені засоби роботи з структурами даних. Цей недолік був виправлений у мові Кобол (COBOL - Common Business Oriented Language). Кобол спеціально призначався для обробки фінансово-економічних даних. Крім того, розробники постаралися зробити Кобол максимально схожим на природну англійську мову, що дозволило писати програми на цій мові навіть неспеціалістам у програмуванні. З другим поколінням комп'ютерів почався розвиток індустрії програмного забезпечення.

У цілому даний період розвитку обчислювальної техніки характеризується застосуванням для створення комп'ютерів транзисторів і пам'яті на феритових сердечниках, збільшенням швидкодії комп'ютерів до декількох сотень тисяч операцій за секунду, виникненням нових технологій програмування, мов програмування високого рівня, операційних систем. Комп'ютери другого покоління отримали широке розповсюдження, вони використовувалися для наукових, інженерних і фінансових розрахунків, для обробки великих обсягів даних на підприємствах, у банках, державних організаціях.

1.4 Третє покоління комп'ютерів – інтегральні мікросхеми (1964-1971 роки)

У 1958 році інженер компанії Texas Instruments Джек Кілбі запропонував ідею інтегральної мікросхеми – германієвого кристала, на який вмонтовуються мініатюрні транзистори та інші елементи. У тому ж році Кілбі представив перший зразок інтегральної мікросхеми, що містить п'ять транзисторних елементів на кристалі германію. Мікросхема Кілбі займала трохи більше сантиметра площі і була декілька міліметрів товщиною. Рік потому, незалежно від Кілбі, Нойс розробив інтегральну мікросхему на основі кристала кремнію. Надалі Роберт Нойс та Гордон Мур заснували компанію «Intel» з виробництва інтегральних мікросхем. Мікросхеми працювали значно швидше транзисторів і споживали значно менше енергії.

Перші інтегральні мікросхеми складалися всього з декількох елементів. Проте, використовуючи напівпровідникову технологію, учені досить швидко навчилися розміщувати на одній інтегральній мікросхемі спочатку десятки, а потім сотні і більше транзисторних елементів.

У 1964 році компанія IBM випустила комп'ютер IBM System 360, побудований на основі інтегральних мікросхем. Сімейство комп'ютерів IBM System 360 – найчисленніше сімейство комп'ютерів третього покоління та одне з найвдаліших в історії обчислювальної техніки. Випуск цих комп'ютерів можна вважати початком масового виробництва обчислювальної техніки. Всього було випущено більше 20 тисяч екземплярів IBM System 360. IBM System 360 належить до класу так званих мейнфреймів.

Компанія DEC (Digital Equipment Corporation) представила модель мінікомп'ютера PDP-8. Мінікомп'ютери, або комп'ютери середньої продуктивності, характеризуються високою надійністю і порівняно низькою вартістю. Низька порівняно з вартістю суперкомп'ютерів вартість мінікомп'ютерів дозволила почати застосовувати їх у невеликих організаціях – дослідницьких лабораторіях, офісах, на невеликих промислових підприємствах.

У той же час проходило вдосконалення програмного забезпечення. Операційні системи будувалися так, щоб

підтримувати більшу кількість зовнішніх пристроїв, з'явилися перші комерційні операційні системи і нові прикладні програми.

У 1968 році на одній з конференцій Дуглас Енгельбарт із Стенфордського інституту продемонстрував створену ним систему взаємодії комп'ютера з користувачем, що складається з клавіатури, покажчика «миші» і графічного інтерфейсу, а також деякі програми, зокрема текстовий процесор і систему гіпертексту.

У 1964 році з'явилася мова програмування Бейсік (BASIC - Beginner's ALL-PURPOSE Symbolic Instruction Code), призначена для навчання програмістів-початківців. Бейсік забезпечував швидке введення і перевірку програм. Бейсік не дуже підходив для написання серйозних програм, проте він давав загальне уявлення про програмування та дозволяв людям, далеким від комп'ютерів, швидко оволодіти основними навичками програмування.

У 1970 році щвейцарець Ніклас Вірт розробив мову програмування Паскаль, також призначену для навчання принципам програмування. Створюваний як мова для навчання, Паскаль виявився дуже зручним для вирішення багатьох прикладних задач. Він чудово забезпечував застосування методів структурного програмування, що стало необхідним при створенні великих програмних систем.

Основою для комп'ютерів третього покоління послужили інтегральні мікросхеми, що дозволило значно зменшити вартість і розміри комп'ютерів, почалося масове виробництво комп'ютерів. У даний період розвитку обчислювальної техніки продовжувалося збільшення швидкості обробки інформації. Комп'ютери третього покоління працювали з швидкістю до одного мільйона операцій за секунду. З'явилися нові зовнішні пристрої, що полегшують взаємодію людини з комп'ютером. Збільшення швидкодії комп'ютерів та галузей їх застосування потребувало розроблення нових методів створення програмного забезпечення. З'явилися перші комерційні операційні системи реального часу, спеціально розроблені для них мови програмування високого рівня. Галузь застосування комп'ютерів третього покоління надзвичайно широка: системи обробки даних, управління, проектування, вирішення різних комерційних задач.

1.5 Четверте покоління комп'ютерів (з 1971 року)

У 1965 році голова ради директорів компанії «Intel» Гордон Мур передбачив, що кількість елементів на інтегральних мікросхемах повинна подвоюватися кожні 18 місяців. Надалі це правило, відоме як закон, було застосовано до швидкості мікропроцесорів і до цих пір не порушувалося.

У 1969 році компанія «Intel» випустила майже найважливіший для розвитку обчислювальної техніки пристрій – **мікропроцесор**. Мікропроцесор являє собою інтегральну мікросхему, на якій зосереджений оброблювальний пристрій з власною системою команд. Конструкція мікропроцесора дозволяє застосовувати його для вирішення широкого кола задач, створюючи при цьому різні функціональні пристрої. Використання мікропроцесорів значно спростило конструкцію комп'ютерів. Практично відразу мікропроцесори отримали широке застосування в різних системах управління від космічних апаратів до побутових приладів.

Протягом наступних десятиріч, дотримуючись закону Мура, триває збільшення швидкості та інтеграції мікропроцесорів. З'явилися надвеликі інтегральні схеми, що містять сотні тисяч і навіть мільйони елементів на один кристал. Це дозволило продовжити зменшення розмірів, вартості комп'ютерів і підвищити їхню продуктивність і надійність.

Практично одночасно з мікропроцесорами з'явилися мікрокомп'ютери, або персональні комп'ютери, відмітною особливістю яких є невеликі розміри і низька вартість. Завдяки своїм характеристикам персональні комп'ютери дали можливість практично будь-якій людині ознайомитися з обчислювальною технікою. Комп'ютери перестали бути прерогативою великих компаній та державних установ, а перетворилися на товар масового споживання.

Першою у виробництві персональних комп'ютерів була компанія Apple. Її фундатори Стів Джобс і Стів Возняк створили першу модель персонального комп'ютера у 1976 році та назвали її Apple I. У 1977 році вони представили свій комп'ютер членам комп'ютерного клубу в Каліфорнії і наступного дня отримали замовлення на 50 подібних комп'ютерів. Вартість першого

персонального комп'ютера становила всього 500 доларів. У 1977 році компанія Apple представила наступну модель персонального комп'ютера – Apple II. У нової моделі був витончений пластиковий корпус з вбудованою клавіатурою. Уперше комп'ютер набув рис побутового приладу. Продаж персональних комп'ютерів різко зросли. Apple II остаточно зламав уявлення про комп'ютер, як про величезного залізного монстра, у нього був витончений дизайн і доброзичливий інтерфейс взаємодії з користувачем.

ПК не цікавили великі компанії до 1979 року, коли з'явився перший процесор електронних таблиць – VISICALC. Ідея VISICALC була запропонована студентом Гарварду Даном Бріськліном, якому довелося розв’язувати складні фінансові задачі, що вимагають великої кількості обчислень. Зі своїм товаришем Бобом Франкстоном вони написали VISICALC для комп'ютера Apple II. Програма виявилася настільки зручною для фінансових обчислень, що багато компаній почали купувати Apple II з VISICALC для своїх співробітників.

У 1981 році найбільша комп'ютерна компанія IBM представила свій перший персональний комп'ютер – IBM PC. Протягом двох років було продано більше п'яти мільйонів цих комп'ютерів. У той же час компанія Microsoft починає випуск ПЗ для IBM PC. З'являються клони IBM PC, але всі вони, так чи інакше, відображають стандарти, закладені IBM. Поява клонів IBM PC сприяла зростанню промислового виробництва ПК.

У 1984 році компанія Apple представила комп'ютер «Макінтош». Операційна система «Макінтош» включала графічний інтерфейс користувача, що дозволяв уводити команди, вибираючи їх за допомогою покажчика «миша». Самі команди були подані у вигляді невеликих графічних зображень – значків. Простота використання в поєднанні з великим набором текстових і графічних програм зробила цей комп'ютер ідеальним для невеликих офісів, видавництв, шкіл і навіть дитячих садків. З появою «Макінтош» персональний комп'ютер став ще більш доступним. Для роботи з ним більше не вимагалося ніяких спеціальних навичок, а тим більш знання програмування.

У 1984 році компанія Apple показала на телебаченні перший ролик, присвячений рекламі персонального комп'ютера.

Комп'ютер справді перестав бути чимось особливим і перетворився на звичайний побутовий прилад.

Протягом усього 50 років комп'ютери перетворилися з незграбних дивовижних електронних монстрів у потужній, гнучкий, зручний та доступний інструмент. Комп'ютери стали символом прогресу в XX столітті. Людині знадобиться обробляти все більшу кількість інформації, а тому будуть удосконалюватися і засоби її обробки – комп'ютери.

1.6 П’яте покоління комп’ютерів – ?

Неодноразово заявлялося про появу нового п'ятого покоління комп'ютерів, яке ось-ось переверне весь обчислювальний світ. На даний момент можна упевнено говорити, що нової комп'ютерна ера не настала, а всі напрямки, які пропонувалися, так і залишилися одиничними або взагалі теоретичними. Далі розглядається декілька найгучніших, але таких, що так і не стали п'ятим поколінням, комп'ютерів.

1.6.1 Нейрокомп'ютер

Нейрокомп'ютер – пристрій для переробки інформації на основі принципів роботи природних нейронних систем. Ці принципи були формалізовані, що дозволило говорити про теорію мереж на штучних нейронах. Проблема нейрокомп’ютерів полягає у побудові реальних фізичних пристроїв, що дозволить не просто моделювати штучні нейронні мережі на звичайному комп'ютері, але так змінити принципи роботи комп’ютера, що стане можливим говорити про те, що вони працюють відповідно до теорії мереж на штучних нейронах.

На відміну від систем цифр, що являють собою комбінації процесорних і запам’ятовуючих блоків, нейропроцесори містять пам’ять, розподілену у зв'язках між дуже простими процесорами, які часто можуть бути описані як нейрони або блоки з однотипних формальних нейронів. Тим самим основне навантаження на виконання конкретних функцій процесорами лягає на архітектуру системи, деталі якої у свою чергу

визначаються міжнейронними зв'язками. Підхід, заснований на представленні як пам'яті даних, так і алгоритмів системою зв'язків (і їхніми вагами), називається конекціонізмом.

Три основні переваги нейрокомп'ютерів:

1) усі алгоритми нейроінформатики високо паралельні, а це вже наслідок високої швидкодії;

2) нейросистеми можна легко зробити дуже стійкими до перешкод і руйнувань;

3) стійкі та надійні нейросистеми можуть створюватися і з ненадійних елементів, що мають значний розкид параметрів.

Розробники нейрокомп'ютерів прагнуть об'єднати стійкість, швидкодію і паралелізм АВМ — аналогових обчислювальних машин — з універсальністю сучасних комп'ютерів.

1.6.2 Біомолекулярна електроніка

Біомолекулярна електроніка (нанобіоелектроніка) — розділ електроніки та нанотехнологій, у яких використовуються біоматеріали і принципи переробки інформації біологічними об'єктами в обчислювальній техніці для створення електронних пристроїв. У 1974 році А. Авірам і М. Ратнер запропонували використовувати окремі молекули як елементарну базу електронних пристроїв. Потім М. Конрад запропонував концепцію ферментативного нейрона, засновану на безперервних розподілених середовищах, що обробляють інформацію. Ці ідеї дали початок квазібіологічній парадигмі, яка, базуючись на ідеях нейронних мереж Мак Каллоха і Піттса, дозволила практично реалізувати молекулярні нейромережеві пристрої, наприклад, на основі білка бактеріородопсину.

1.6.3 Нечітка логіка

Нечітка логіка й теорія нечітких множин — розділ математики, що є узагальненням класичної логіки та теорії множин. Поняття нечіткої логіки було вперше введено професором Лютфі Заде у 1965 році. У цій роботі поняття множини було розширено допущенням, що функція приналежності елемента до множини може набувати будь-яких значень в інтервалі [0...1], а не тільки 0 або 1. Такі множини були

названі нечіткими. Також автором були запропоновані різні логічні операції над нечіткою множиною та запропоновано поняття лінгвістичної змінної, значенням якої виступає нечітка множина.

Контрольні запитання

1 Чому нумерація поколінь комп'ютерів починається з «нуля»?

2 Які властивості характерні першому поколінню комп'ютерів – електронні лампи?

3 Чому друге покоління комп'ютерів майже миттєво замінило перше?

4 Які особливості надали мікросхеми третьому поколінню комп'ютерів у порівнянні з попередніми?

5 Як почалася ера четвертого покоління комп'ютерів – мікропроцесорного?

6 Чому досі не вважається, що настала ера п'ятого покоління?

2 Місце персонального комп'ютера у сучасних ЦЕОМ

Електронно-обчислювальна машина (скорочено ЕОМ) — загальна назва для обчислювальних машин, що є електронними (починаючи з перших лампових машин, включаючи напівпровідникові тощо), на відміну від електромеханічних (на електричних реле тощо) та механічних обчислювальних машин.

За часів широкого розповсюдження аналогових обчислювальних машин, що теж були у своїй переважній більшості електронними, для уникнення непорозумінь використовувалася назва «цифрова електронна обчислювальна машина» (ЦЕОМ) або «лічильна» (рос. *счётная*) машина (задля підкреслення того, що ЦЕОМ саме реалізує безпосередньо обчислення результату, у той час як аналогова машина фактично реалізує процес фізичного моделювання з отриманням результату вимірювання).

Унікальний винахід XX століття — ЦЕОМ — найчастіше називають англійським словом — комп'ютер (computer), що в перекладі дає те саме значення: обчислювач.

Гордон Мур (засновник корпорації Intel, що першою у світі створила мікропроцесор) зазначив, що є два шляхи розвитку цифрової електронно-обчислювальної техніки:

1) створення комп'ютерів есе більшої потужності при постійній ціні;

2) випуск однієї моделі при постійному зниженні ціни.

Сучасна комп'ютерна промисловість іде двома цими шляхами.

Але за останні 20-30 років з'явилась тенденція щодо розмежування різних за завданнями цифрових електронно-обчислювальних машин.

Одноразові комп'ютери – маленькі, малофункціональні, спеціалізовані пристрої. Функції, які вони можуть виконувати: вбудовані у вітальні листівки, програвачі мелодій, радіопередавачі, які планується імплантувати тваринам, ідентифікатори товарів у супермаркетах, порогові пристрої в детекторній техніці, маркування багажу в аеропортах. Європейський центральний банк планує в найближчі роки налагодити випуск банкнот, які будуть запам'ятовувати всі інстанції проходження та ін.

Такі комп'ютери можуть бути як пасивними (немає джерела живлення), так і активними (знаходяться в постійній дії).

Найбільш відома мікросхема товщиною близько півміліметра – RFID (Radio Frequency Identification – радіочастотна ідентифікація).

Мікроконтролери. Основне завдання таких комп'ютерів – виконання функцій управління пристроями та організація на їх базі інтерфейсів для користувача:

1) побутові прилади (будильники, пральні машини, мікрохвильові печі, охоронні сигналізації);

2) комунікатори (телефони, апарати факсиміле, маршрутизатори);

3) периферійні пристрої (принтери, сканери, приводи та ін.);

4) розважальні пристрої (відеомагнітофони, DVD-програвачі, музичні центри, MP3-плеєри та ін.);

5) формувачі зображень (телевізори, цифрові фото- і відеокамери, копіювальні пристрої);

6) медичне обладнання;

7) військові комплекси;

8) торгове обладнання;

9) іграшки;

10) автомобілі тощо.

На відміну від одноразових комп'ютерів, мікроконтролери є повноцінними комп'ютерами. Усі мікроконтролери поділяють на універсальні і спеціальні. Універсальні – комп'ютери, зменшені у розмірі. Спеціальні – обмежені архітектурою та системою команд і пристосовані для вирішення певного кола завдань.

Мікроконтролери бувають 4-, 8-, 16-, 32-розрядними.

Три основні напрями розвитку мікроконтролерів:

1) ціна;

2) робота в реальному масштабі часу;

3) розмір і енергоспоживання.

Ігрові комп'ютери. По суті це звичайні ПК, у яких розширені можливості графічних і звукових підсистем. При цьому використовуються не останні моделі процесорів: PlayStation-2 (процесор 296 МГц, але 128 розрядів даних, пам'ять 32 Мбайт, графічна мікросхема 160 МГц, 48-канальна звукова плата), XBOX (P3 733 МГц, пам'ять 64 Мбайт, графічна мікросхема 300 МГц, 256-канальна звукова мікросхема).

Персональні комп'ютери. Два варіанти – настільні і портативні (ноутбуки). Особливістю є наявність складної операційної системи.

Сервери. Основна відмінність від ПК – величезні обсяги пам'яті (ОЗУ, вінчестери). Архітектурно вони мало відрізняються від ПК.

Комплекси робочих станцій. На сучасному етапі розвитку – звичайні ПК, зібрані в загальну систему з розпаралелюванням обчислювальних можливостей.

Мейнфрейми. Справді величезні комп'ютери. Працюють не швидше за звичайні, але в них вища швидкість процесів введення-виведення і великі простори на диску. Деякий час тому з'явилася тенденція до витіснення таких машин **серверами**, оскільки такі машини вимагають особливого підходу,

спеціальних програмних засобів і дуже дорогі при купівлі й обслуговуванні. Але, на щастя, дуже скоро стало зрозумілим, що такі машини все ж таки необхідні для обробки величезних масивів даних і в популярних Інтернет-вузлах, де необхідно здійснювати величезну кількість трансакцій за секунду.

Суперкомп'ютер – це загальний термін, який використовується для позначення класу існуючих найпотужніших комп'ютерних систем. Суперкомп'ютери, зазвичай, використовуються при розв'язанні складних наукових та інженерних задач, які вимагають виконання великої кількості математичних операцій та(чи) працюють з великими обсягами даних.

Усі поняття є відносними в часі. Потужність комп'ютерної системи оцінюється в порівнянні з існуючими на певний момент комп'ютерними системами широкого використання та рівнем розвитку технологій.

Контрольні запитання

1 Які властивості виділяють групу обчислювальних пристроїв – одноразові комп'ютери?

2 Які властивості виділяють групу обчислювальних пристроїв – мікроконтролери?

3 Які властивості виділяють групу обчислювальних пристроїв – ігрові комп'ютери?

4 Які властивості виділяють групу обчислювальних пристроїв – персональні комп'ютери та ноутбуки?

5 Які властивості виділяють групу обчислювальних пристроїв – сервери?

6 Які властивості виділяють групу обчислювальних пристроїв – мейнфрейми?

7 Які властивості виділяють групу обчислювальних пристроїв – комплекси робочих станцій?

8 Які властивості виділяють групу обчислювальних пристроїв – суперкомп'ютери?

3 Призначення персональних комп'ютерів

Персональний комп'ютер (ПК) – інформаційно-обчислювальна система, максимально наближена до користувача. ПК має високу швидкодію та завдяки гнучкості підсистеми введення-виведення дозволяє підключати різні зовнішні і периферійні пристрої.

ПК має модульну побудову. Це означає, що він об'єднує навколо центральних пристроїв змінні вузли – модулі, які можна замінювати сумісними, більш сучасними. Подібна концепція модульності забезпечує можливість модернізації ПК, оновлення периферії, підвищує ремонтопридатність і зменшує час простоїв обладнання через відмови системи.

ПК містить такі складові:

1) системний блок;
2) монітор;
3) клавіатуру;
4) комп'ютерну мишу;
5) периферійні пристрої (принтери, сканери і т.д.)

3.1 Системний блок

Системний блок включає:

1) корпус;
2) системну плату;
3) блок живлення;
4) вентилятори;
5) роз'єми з кабелями
6) пристрої зовнішньої пам'яті;
7) інші пристрої.

Центральним елементом будь-якого системного блока ПК є материнська плата (пункт А.1). На материнській платі знаходяться:

1) роз'єм центрального процесора (ЦП);
2) роз'єм системної оперативної пам'яті (оперативний пристрій, що запам'ятовує (ОЗУ));
3) роз'єми для підключення різних пристроїв;

4) контролери різних пристроїв;
5) порти пристроїв введення-виведення;
6) шина адреси;
7) шина даних;
8) шина управління.

Так чи інакше, але всі пристрої, як зовнішні, так і внутрішні, підключаються до материнської плати.

3.2 Зовнішні та периферійні пристрої

У ПК не існує чітко окресленої грані між поняттям периферійних і зовнішніх пристроїв. Багато сучасних пристроїв (монітор, клавіатура, миша, дискова пам'ять) є невід'ємними складовими частинами ПК, але належать до числа зовнішніх по відношенню до центра (системного блока).

Будь-який пристрій, який може бути підключений до ПК через порт, називається периферійним пристроєм. Такими пристроями є принтер, сканер, модем, пристрої мультимедіа.

Монітор відображує на екрані текстову та графічну інформацію, що вводиться з клавіатури або виводиться з ПК.

Клавіатура дозволяє вводити в ПК команди і дані.

Комп'ютерна миша – пристрій введення, що дозволяє вказувати на елементи екрана за допомогою покажчика.

Принтер дозволяє виводити як тверду копію текстову і графічну інформацію.

Модем – це комунікаційний пристрій введення-виведення для підключення до цифрової або аналогової ліній зв'язку і доступу до послуг інформаційної мережі.

Сканер – пристрій введення інформації в графічному вигляді з твердого носія.

Аудіопідсистема – мультимедійний компонент виведення звукової інформації

3.3 Структурна схема ПК

На рисунку 3.1 наведено класичну структурну схему ПК.

Центральний процесор (ЦП) являє собою інтегральну мікросхему (ІМС) з високим ступенем інтеграції, що виконує в ПК основну обчислювальну роботу. ЦП (авіаносець в авіанесучому флоті) керує введенням-виведенням, взаємодіючи зі всіма пристроями та окремими системами комп'ютера. ЦП знаходиться у функціональному центрі комп'ютера та оточений ІМС, у яких міститься системна логіка управління ПК (чипсет).

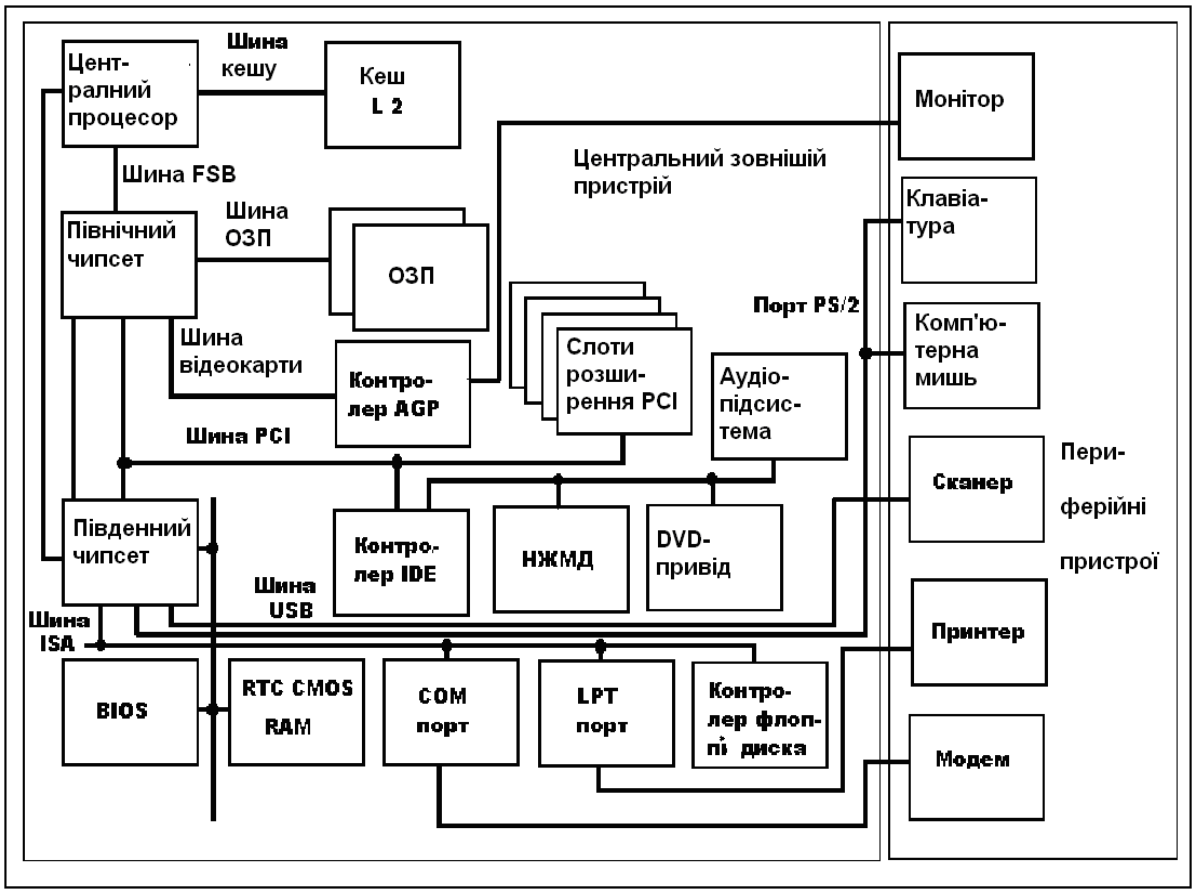


Рисунок 3.1 – Класична структурна схема сучасного ПК

Чипсет – логічні схеми, що допомагають процесору керувати ПК. Всі логічні мікросхеми (чипсет) розташовані на материнській платі. Логіка керування розбита на групи:

1) логіка арбітражу;
2) логіка управління процедурами;
3) логіка управління зовнішніми пристроями;
4) логіка очікування (затримок);
5) логіка управління шинами.

Основна задача чипсета – робота ЦП з постійною продуктивністю.

У мікросхеми чипсетів інтегровані різні контролери, що відстежують правильне використання системних ресурсів. Деякі параметри чипсетів можна будувати за допомогою Setup BIOS.

Контролери дозволяють підключити до материнської плати периферійні пристрої. Контролери – це засоби управління периферійними пристроями. З їх допомогою виконується процедура обміну даними.

Плата або роз'єми, на яких розташовані чипи контролерів, називають адаптерами. Кожний пристрій, що підключається до ПК, має свій адаптер або буде підключений до вбудованого адаптера материнської плати.

Інтерфейс – апаратно-програмна система сполучення об'єктів з різними характеристиками.

Для реалізації логічної взаємодії між ЦП і периферійними пристроями (при шляху контролера) використовуються програми підключення – драйвери пристроїв.

Програмні драйвери інтегровані в BIOS, ОС або окремих носіях, що поставляються.

Контролери периферійних пристроїв підключаються до шини введення-виведення ПК за допомогою інтерфейсу введення-виведення.

Інтерфейси бувають апаратні і програмні.

Апаратні – сукупність ліній зв'язку, логічних елементів і допоміжних схем управління, призначених для перетворення сигналів і з'єднання пристроїв.

Програмний інтерфейс дозволяє сполучати окремі програми з різними параметрами, а також надає користувачу умови роботи з програмними продуктами з тим або іншим ступенем комфорту.

Чипсет – це набір мікросхем материнської плати, який складається з двох основних мікросхем: північний і південний міст.

Північний міст (Northbridge) відповідає за роботу з процесором, пам'яттю і відеоадаптером. Північний міст визначає частоту системної шини, можливий тип оперативної пам'яті, її максимальний обсяг і швидкість обміну інформацією з процесором.

Південний міст (Southbridge) — це мікросхема, яка забезпечує взаємодію між центральним процесором і жорстким диском, картами PCI, PCI Express, інтерфейсами IDE, SATA, USB та ін. На відміну від північного моста, південний міст зазвичай не підключений напряму до процесора (CPU).

Архітектуру чипсета можна подати у вигляді географічної карти. Тоді процесор буде знаходитися на верху, як на півночі. Він буде з'єднаний з чипсетом через швидкий північний міст, а північний міст буде з'єднаний з іншою частиною чипсета через повільний південний міст.

Порти введення-виведення (пристрої введення-виведення УВВ)

Кожний інтерфейс містить регістри, які називають портами введення-виведення. Порт введення-виведення – основний елемент пристрою введення-виведення (ПВВ). Регістр – електронний елемент, що має малу пам'ять (байт, слово), призначений для тимчасового зберігання інформації на будь-якій з шин. Регістри бувають як окремими ІМС, так можуть розташовуватися всередині ІМС контролерів або процесорів. Кожний порт ПК має свою адресу. Для зручності багато портів крім адреси мають власну назву (логічне ім'я).

Деякі порти доступні користувачам. За допомогою BIOS можна змінити настройки (пункт А.2) та навіть адреси портів УВВ.

Порти бувають послідовні та паралельні.

Послідовний порт – порт, інформація в який надходить і виходить у вигляді послідовності імпульсів інформаційних пакетів. Вимірювання – біт за секунду.

Паралельний порт – інформація передається побайтно або послівно. Відповідно швидкість вимірюється одиницях байт або слів за секунду.

Шини (магістралі)

Усі електронні елементи ПК зв'язані між собою за допомогою паралельних доріг, які називають шинами. Кількість доріг шини визначає її розрядність. Так, 32 розрядна шина даних означає, що між процесором і пам'яттю на платі прокладено

32 лінії, по яких одночасно (за один такт) проходить 4 байти інформації.

Шини бувають:

1) адреси – по шині адреси процесор звертається до пам'яті або ПВВ;

2) даних – по шині даних процесор пересилає інформацію за обраною адресою;

3) керування – по шині керування відбувається управління всіма елементами ПК.

Тактовий генератор

Обчислювальна робота в ЦП, а також обробка та пересилання даних між ЦП, ОЗУ, контролерами, ПВВ повинна бути злагоджена в часі – синхронізована. Для синхронізації використовується тактовий генератор. Тактовий генератор формує періодичну послідовність тактових імпульсів, які прямують на шину ЦП, ОЗУ і шини ПВВ. Вимірюється в мегагерцах або гігагерцах.

У ЦП частоту задає тактовий генератор, розташований на системній платі. Необхідна для роботи ЦП частота виходить при множенні частоти синхронізації системи на коефіцієнт у блоці множення, що вбудований у ЦП. Кожна операція на шині триває певний період часу, що називається циклом (рисунок 3.2). Мінімальний цикл обміну триває два цикли тактового генератора – T1+T2.

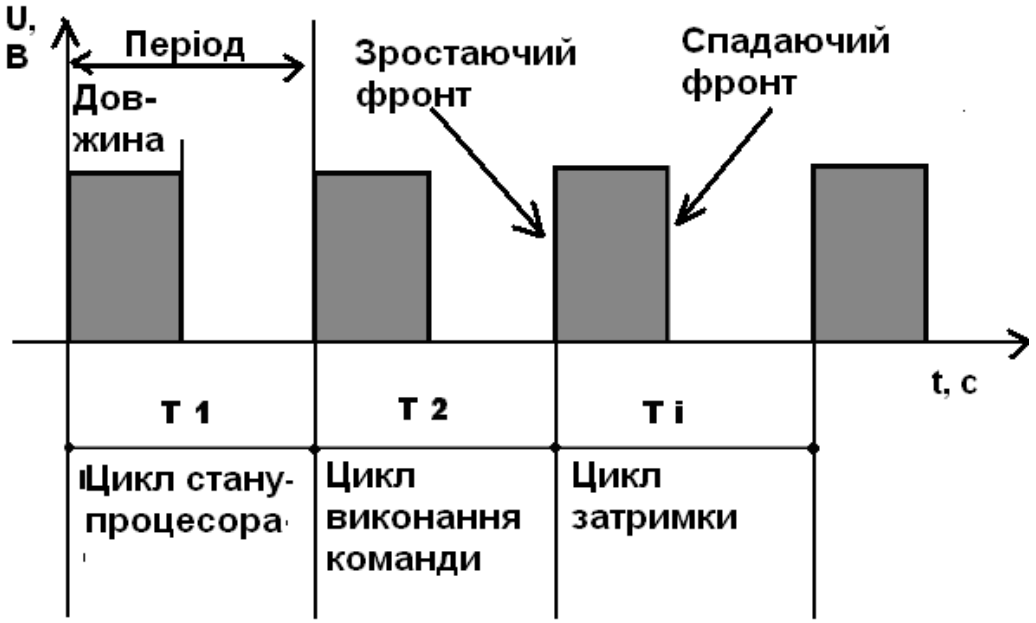


Рисунок 3.2 – Схема імпульсів тактового генератора

ОЗУ потрібно деякий час на завершення внутрішньосхемного процесу доступу до необхідних елементів пам'яті. У період часу дії цих затримок процесор знаходиться у стадії циклів очікування. Потім ОЗУ повідомляє ЦП сигнал готовності до наступної операції.

Інтервальний таймер. Для точного визначення часу виконання процесів на шинах використовуються інтервальні таймери. Лічильники, що розташовані в таймерах, призначені для системного годинника, установлення часу початкової регенерації динамічної пам'яті, визначення затримок та ін., вони дозволяють точно визначити час установлення та зняття сигналів на лініях.

Формфактори системної плати. Визначають не тільки розмір системної плати, але й можливості для установлення тих або інших вузлів, ефективність тепловідведення, використання корпусів і блоків живлення певних типів. Найбільш часто використовують формфактори ATX, розроблені для системних блоків Tower у 1995 році. Найвідоміші – micro-ATX, flex-ATX, ATX-RISER.

Контрольні запитання

1 ПК мають модульну структуру, назвіть складові.

2 Системний блок має модульну структуру, назвіть складові.

3 Які складові системної плати?

4 Які функції чипсетів?

5 Які шини використовуються в ПК та які функції вони виконують?

6 Для чого використовують тактовий генератор та інтервальний таймер?

4 Шини системної плати

Для організації взаємодії пристроїв на системній платі використовуються шини. Шина – це загальний канал зв'язку, вживлений в ПК для передачі даних, адрес і сигналів керування, сигналізації та контролю.

Шини бувають внутрішні та зовнішні.

Внутрішні шини розташовані на материнці. Чипсет може підтримувати відразу декілька протоколів внутрішніх і зовнішніх шин (ISA, PCI, USB, AGP). Шина відображує архітектуру чипсета та з'єднана з ЦП шиною FSB. Таким чином, до середньосистемних відносяться шини FSB, ОЗУ, ISA, PCI, PCI-Express (або AGP).

Зовнішні шини підключають до ПК периферійні пристрої. До зовнішніх належать: SCSI, IDE, USB, SATA та ін.

FSB – Шина процесора, найшвидкісніша на системній платі. Використовується процесором для обміну даними між кеш-пам'яттю та контролером ОЗУ, розташованим у чипсеті.

ОЗУ – підключає системну пам'ять до контролера пам'яті. В ідеалі швидкість на FSB та ОЗУ повинна бути однаковою.

ISA (Industry Standard Architecture - Архітектура промислового стандарту), інша назва AT-BUS. Шина ISA є основною шиною на материнській платі застарілих комп'ютерів типу PC AT. Специфікація PC'99 рекомендує не включати шину ISA до складу нової материнської плати для персональних комп'ютерів, оскільки її застосування значно знижує загальну продуктивність системи. На новій материнській платі цей інтерфейс або відсутній, або представлений усього 1-2 слотами

(роз'ємами) розширення для підключення застарілих компонентів. Конструктивно шина ISA являє собою роз'єм на материнській платі, що складається з двох частин - 62-контактного та примикаючого до нього 36-контактного сегментів. Максимальна пропускна спроможність шини ISA не перевищує 5,55 Мбайт/с і цілком недостатня для сучасних вимог. Через інтерфейс ISA раніше підключалися практично всі компоненти ПК, такі, як відеокарти, контролери введення-виведення, контролери жорстких і гнучких дисків, модеми, звукові карти та інші пристрої.

EISA (Enhanced ISA - Розширена ISA). У роз'єми шини EISA можна вставляти як плати для шини ISA, так і для EISA. Плати для шини EISA мають роз'єм з додатковими рядами контактів, а слот має розташований у глибині ряд додаткових контактів. Максимальна пропускна спроможність – 32 Мбайт/с. Підтримує режим управління шиною збоку будь-якого з пристроїв, установлених в роз'єм (Bus Mastering). На сучасній материнській платі шина EISA уже не зустрічається.

VLB (VESA Local Bus - Локальна шина стандарту VESA). Цей інтерфейс є 32-розрядним розширенням шини ISA. Шина VLB розташовується на материнській платі та конструктивно виглядає як 116-контактний додатковий роз'єм, що продовжує лінійку слотів ISA (разом - три розташовані підряд секції). Тактова частота шини VLB - до 50 МГц, максимальна пропускна спроможність 130 Мбайт/с. Через цей інтерфейс підключалися в основному відеокарти. Кожний компонент, установлений на шині VLB, може обмінюватися даними з процесором напряму, без проміжної буферизації. Це збільшує навантаження на процесор, погіршує проходження сигналів по шині і знижує надійність обміну даними. Тому інтерфейс VLB має жорстке обмеження на кількість установлюваних пристроїв, залежно від тактової частоти шини: при 33 МГц - три, 40 МГц - два, 50 МГц - один. На даний час інтерфейс VLB зустрічається тільки на старих комп'ютерах.

PCI (Peripheral Component Interconnect - з'єднання зовнішніх компонентів). Цей інтерфейс не сумісний з жодним з попередніх. Підтримує тактову частоту до 33 МГц (варіант PCI 2.1 - до 66 МГЦ), має максимальну пропускну спроможність до 132 Мбайт/с на частоті 66 МГЦ для 32-розрядної шини (264

Мбайт/с для 32- розрядних і 528 Мбайт/с для 64-розрядних даних на частоті 66 МГЦ). Конструктивно роз'єм складається з двох розміщених підряд секцій по 64 контакти. Усередині другої секції є пластмасова поперечна перегородка (ключ) для запобігання неправильному установленню карт. Роз'єми PCI та плати до них підтримують рівні сигналів або 5 В, або 3,3 В, або обидва рівні (універсальні). У перших двох випадках плати повинні відповідати рівню сигналу роз'єму, універсальні плати ставляться в будь-який роз'єм. Інтерфейс PCI забезпечує підтримку режимів Bus Mastering і автоматичної конфігурації компонентів при установленні (PLUG-AND-PLAY). Усі слоти PCI на материнській платі згруповані в сегменти, число роз'ємів у сегменті обмежено чотирма. Якщо сегментів декілька, вони з'єднуються шляхом так званих мостів (bridge).

PCMCIA (Personal Computer Memory Card International Association – стандарт міжнародної асоціації виробників плата пам'яті для персональних комп'ютерів). Інтерфейс PCMCIA служить для підключення зовнішніх пристроїв до мобільних комп'ютерів класу NOTEBOOK. Підтримує автоматичну конфігурацію PLUG-AND-PLAY, підключення та відключення пристроїв у процесі роботи комп'ютера («гаряче» підключення). Конструктивно являє собою мініатюрний 68-контактний роз'єм.

USB (Universal Serial Bus – універсальна послідовна шина). До одного USB каналу можна ланцюжком підключити до 127 зовнішніх пристроїв. На сучасній материнській платі звичайно є по два канали USB на контролер. Обмін даними по шині USB проходить у пакетному режимі при максимальній пропускній спроможності до 12 Мбіт/с, у версії 2.0 швидкість збільшена до 450 Мбіт/с, у версії 3.0 швидкість збільшена до 4,8 Гбіт/с.

AGP (Accelerated Graphics Port – прискорений графічний порт). Цей інтерфейс призначений виключно для підключення відеоадаптерів. Шина AGP дозволяє відеоадаптеру зв'язуватися з оперативною пам'яттю безпосередньо, розвантажуючи тим самим системну шину. В оперативній пам'яті розміщуються параметри тривимірних об'єктів, що вимагають швидкого доступу як збоку процесора, так і збоку відеоадаптера. Максимальна пропускна спроможність шини AGP в режимі чотирикратного множення AGP/x4 - до 1066 Мбайт/с. Конструктивно виглядає як окремий

роз'єм на материнській платі. Ніякі інші компоненти, окрім відеоадаптерів, до AGP підключити не можна. Зараз витіснено шиною PCI-Express.

IEEE1394 (Institute of Electrical and Electronic Engineers 1394 – стандарт інституту інженерів з електротехніки та електроніки 1394) має іншу назву **FIREWIRE** (вогненний дріт). За допомогою інтерфейсу IEEE1394 можуть підключатися як внутрішні, так і зовнішні пристрої (всього до 127 пристроїв на один контролер). Максимальна пропускна спроможність досягає 50 Мбайт/с; розробляються модифікації інтерфейсу, здатні передавати дані із швидкістю 200 Мбайт/с і навіть 800 Мбайт/с. Головною перевагою інтерфейсу IEEE1394 є швидкість і простота підключення декількох пристроїв по єдиному шестижильному кабелю: дві жили забезпечують живлення, чотири - служать для обміну даними. Згідно з вимогами специфікації PC'99, новий інтерфейс покликаний замінити IDE/ATA при підключенні жорстких дисків, CD-ROM і DVD приводів, а також рекомендується для з'єднання з високошвидкісними зовнішніми пристроями – цифровими відеокамерами, відеомагнітофонами, різними накопичувачами.

SCSI (Small Computer System Interface – інтерфейс малих комп'ютерних систем). Існує декілька варіантів інтерфейсу SCSI (читається "сказі"), які відрізняються кількістю пристроїв, що підключаються, максимальною пропускною спроможністю і максимальною довжиною шлейфа. Через інтерфейс SCSI частіше всього підключаються високошвидкісні пристрої, а саме: як жорсткі диски, DVD-приводи, сканери. Для забезпечення роботи компонентів з інтерфейсом SCSI потрібна наявність на комп'ютері спеціального SCSI хост-адаптера, що вставляється в слот розширення материнської плати або вбудований у системну плату. Існують такі специфікації SCSI:*SCSI-1;Fast SCSI-2; FastWide SCSI-2; Ultra SCSI-2; UltraWide SCSI-2; Ultra SCSI-3*. Швидкість передачі даних по шині SCSI може складати від 5Мб/с (SCSI-1) до 80 Мб/с (SCSI-3), частота шини від 5 МГц (SCSI-1) до 40 (SCSI-3), кількість пристроїв, що підключаються, від 8 (SCSI-1, Fast SCSI-2, Ultra SCSI-2) до 16 (FASTWIDE SCSI-2, ULTRAWIDE SCSI-2, Ultra SCSI-3), довжина шлейфа від 1,5 (Ultra SCSI-2, ULTRAWIDE SCSI-2) до 12 м (Ultra SCSI-3). Усі

пристрої SCSI підключаються по ланцюжку, причому перший (SCSI хост-адаптер) та останній пристрої в ланцюжку повинні мати так звані термінатори для узгодження електричних характеристик у ланцюжку.

IDE (ATA) (Integrated Drive Electronics – вбудована електроніка накопичувача; AT Attachment – підключення до AT). Цей інтерфейс призначений тільки для підключення жорстких дисків та інших накопичувачів. У більшості випадків контролер IDE/ATA вбудований у системну плату та підтримує два роз'єми IDE (Primary – первинний та Secondary – вторинний), до кожного з яких можна підключати по два пристрої (Master і Slave – ведучий та ведений). Максимальна пропускна спроможність інтерфейсу IDE - до 66 Мбайт/с (по протоколу Ultra DMA-66). Для забезпечення сумісності з накопичувачами, відмінними від жорстких дисків, існує протокол обміну даними **ATAPI** (ATA Packet Interface – пакетний інтерфейс ATA). Згідно з вимогами специфікації PC'99, інтерфейс IDE/ATA буде поступово замінюватися інтерфейсом IEEE 1394.

SATA (Serial ATA) – послідовна шина ATA. У версії 2 має швидкість до 300 Мб/с (швидкість передачі кодованих даних 3 Гбод). Майже повністю витіснила **IDE** (ATA).

SAS (Serial Attached SCSI) – послідовна версія SCSI (частково сумісна з SATA).

PCI Express, або **PCIe**, або **PCI-E** (також відома як **3GIO** for *3rd Generation I/O*; не має відношення до PCI-X та PXI) – комп'ютерна шина, яка використовує програмну модель шини PCI та високошвидкісний фізичний протокол, побудований на послідовній передачі даних. Для підключення пристрою PCI Express використовується двонаправлене послідовне з'єднання типу точка-точка, яке називають – lane; це відрізняється від PCI, у якої всі пристрої підключено до загальної 32-разрядної двонаправленої шини. З'єднання між двома пристроями PCI Express називають link, воно складається з одного (що називається 1x) або декількох (x2, x4, x8, x12, x16 и x32) двонаправлених послідовних з'єднань lane. Кожний пристрій повинен підтримувати з'єднання x1. Висока пікова швидкість шини PCI Express (до 512 Гбит/с) дозволяє використовувати її замість AGP.

Корпорація Intel заявила про готовність до випуску нової технології передачі даних між комп'ютерами та периферійними пристроями – Thunderbolt. Thunderbolt (раніше Light Peak) має можливість забезпечити швидкість передачі інформації між пристроями до 10 Гбіт/с. Для порівняння, стандарт USB 3.0, котрий тільки починають застосовувати у масовому виробництві, забезпечує лише 3,2 Гбіт/с. Intel стверджує, що в майбутньому нові версії зможуть працювати на швидкості 100 Гбіт/с.

Контрольні питання

1 Яку функцію виконують шини?
2 Чим відрізняються послідовні шини від паралельних?
3 Перерахуйте послідовні шини.
4 Перерахуйте паралельні шини.
5 Які сучасні шини використовуються в ПК та які функції вони виконують?

5 Основні відомості про центральний процесор

Центральний процессор (ЦП; CPU — англ. céntral prócessing únit, дослівно — центральний обчислювальний пристрій) — процесор машинних інструкцій, частина апаратного забезпечення комп'ютера або програмованого контролера, що відповідає за виконання арифметичних операцій, заданих програмами операційної системи, і координуючий роботу всіх пристроїв комп'ютера (пункти А.3-А.6).

Сучасні ЦП, які виконуються у вигляді окремих мікросхем (чипів), що реалізують усі особливості, властиві даного роду пристроям, називають мікропроцесорами. Із середини 1980-х років останні практично витіснили інші види ЦП, внаслідок чого термін став усе частіше та частіше сприйматися як звичайний синонім слова «мікропроцесор». Проте це не так: ЦП пристрої деяких суперкомп'ютерів навіть сьогодні являють собою складні комплекси великих і надвеликих інтегральних схем.

Особливості синхронізації ЦП. Швидкодія ЦП залежить від внутрішньої тактової частоти, технології виробництва і структурної організації.

Системний тактовий генератор синхронізує роботу всієї системи. Завдяки вбудованому в ЦП множнику частоти ЦП працює на більш високих швидкостях, ніж уся система. Доступ до осередків повільного ОЗП вимушує ЦП простоювати.

Обробка команди – процес багатостадійний. На кожну стадію витрачаються декілька десятків тактів. Наприклад, для виконання операції з плаваючою комою можуть знадобитися до 400 тактів синхронізації.

Прямим способом підвищення швидкодії є збільшення тактової частоти ЦП та системи в цілому.

ЦП може працювати в декількох режимах: реальному, захищеному і віртуальному.

5.1 Реальний режим роботи ЦП

Реальний режим (або режим реальних адрес) — цю назву було дано колишньому способу адресації пам'яті після появи Intel 80286-го процесора, підтримуючого захищений режим. Але тільки з появою процесора Intel 80386 можна говорити про захищений режим в сучасному розумінні, бо процесор Intel 80286 був 16-бітним і в ньому ще не була створена сторінкова адресація пам'яті.

Технічний опис

Такий спосіб організації пам'яті дозволяв адресувати 1 Мб + 64 Кб — 16 байт пам'яті, але через наявність у ранніх процесорах тільки 20 адресних ліній адресувався тільки 1 Мбайт.

Використання

У цьому режимі процесори працювали тільки в старих версіях операційних систем MS DOS. Адресувати в реальному режимі додаткову пам'ять за межами 1 Мбайт було не можна (хоча можна було використовувати драйвер himem.sys на машинах з процесором 80286 і вище). Не зважаючи на те, що фірма Intel не передбачала повернення із захищеного у реальний режим, вона забезпечила сумісність 16-бітних програм, створивши ще один спеціальний режим віртуальних адрес V86. При цьому програми мали можливість використовувати колишній спосіб обчислення лінійної адреси, не виходячи із

захищеного режиму процесора. Даний режим дозволив організувати роботу колишньої системи MS DOS всередині нових багатозадачних систем Microsoft Windows.

5.2 Режим захищеної віртуальної адреси

Режим захищеної віртуальної адреси (захищений режим) — режим роботи мікропроцесора. Розроблений фірмою Digital Equipments (DEC) для 32-розрядних комп'ютерів VAX-11, а також фірмою Intel починаючи з 32-розрядних процесорів Intel 80386. Не дивлячись на те, що захищений режим частково був реалізований уже в процесорі Intel 80286, але там істотно відрізнявся спосіб роботи з пам'яттю, оскільки процесори ще були 16-бітні та не була реалізована сторінкова організація пам'яті. Застосовується в процесорах інших виробників. Даний режим дозволив створити такі багатозадачні операційні системи, як сімейство MS Windows, Unix та ряд інших.

З появою 32-розрядних процесорів фірми Intel процесори можуть працювати у трьох режимах: реальному, захищеному та віртуального процесора 8086.

У захищеному режимі використовуються повні можливості 32-розрядного процесора – забезпечується безпосередній доступ до 4 Гбайт фізичного адресного простору та богатозадачний з паралельним виконанням декількох програм (процесів). Власне кажучи, багатозадачний режим організує багатозадачна операційна система. Проте мікропроцесор надає необхідний для цього режиму потужний та надійний механізм захисту задач одна від іншої за допомогою системи чотирирівневих привілеїв. Так само в цьому режимі доступна сторінкова організація пам'яті, що підвищує рівень захисту задач одна від іншої та ефективність їх виконання.

У процесорі Intel 80386 компанія Intel урахувала необхідність кращої підтримки реального режиму, тому що програмне забезпечення того часу до його появи ще не було готове повністю працювати в захищеному режимі. Тому, наприклад, у Intel 80386 можливе перемикання із захищеного режиму назад у реальний (при розроблення Intel 80286

вважалося, що це не потрібно, тому на комп'ютерах з процесором Intel 80286 повернення в реальний режим здійснюється схемна — через скид процесора).

При увімкненні мікропроцесора в ньому автоматично встановлюється режим реальної адреси. Перехід у захищений режим здійснюється програмно шляхом виконання відповідної послідовності команд. Програми, призначені для захищеного режиму, повинні бути написані особливим способом. Це означає, що реальний і захищений режим не сумісні.

Реальний адресний простір – це простір, який визначається числом адресної шини. Наприклад, 24-розрядна адресна шина дозволяє звернутися тільки до 16 Мбайт пам'яті (2 у 24 ступені), а 32-розрядна адресна шина дозволяє звернутися тільки до 4 Гбайт (2 у 32 ступені).

Віртуальний адресний простір – у захищеному режимі роботи ЦП, використовуючи свої системні регістри, може звертатися до 64 Тбайт. Ця пам'ять організована не тільки в області фізичних адрес, а і в логічному (віртуальному) адресному просторі. Щоб організувати віртуальний адресний простір, для кожного сегмента адреси створюється його описова частина, де вказується його особливість і рівень привілеїв. Такий формат називають дескриптором. Усі дескриптори зберігаються в спеціальній таблиці (таблиці дескрипторів сегментів). Окрім дескрипторів, може бути увімкнений сторінковий режим управління пам'яттю.

Сторінкова організація пам'яті. Основна думка зводиться до формування таблиць опису пам'яті, які визначають стан її окремих сегментів/сторінок та ін. При браку пам'яті операційна система може вивантажити частину даних з оперативної пам'яті на диск, а в таблицю описів внести вказівку на відсутність цих даних у пам'яті. При спробі звернення до відсутніх даних процесор сформує виключення (різновид переривання) та віддасть управління операційній системі, яка поверне дані в пам'ять, а потім поверне управління програмі. Таким чином, для програм процес підкачування даних з дисків відбувається непомітно.

5.3 Багатозадачність

Багатозадачність (англ. *multitasking*) — властивість операційної системи або середовища програмування забезпечувати можливість паралельної (або псевдопаралельної) обробки декількох процесів. Істинна багатозадачність операційної системи можлива тільки в розподілених обчислювальних системах.

Властивості багатозадачного середовища. Примітивні багатозадачні середовища забезпечують чисте «розподілення ресурсів», коли за кожною задачею закріплюється певна ділянка пам'яті, і задача активізується в строго визначені інтервали часу.

Більш розвинені багатозадачні системи проводять розподіл ресурсів динамічно, коли задача стартує в пам'яті або покидає пам'ять залежно від її пріоритету і від стратегії системи. Таке багатозадачне середовище має такі особливості:

1) кожна задача має свій пріоритет, відповідно до якого одержує час і пам'ять;

2) система організує черги задач так, щоб усі задачі отримали ресурси, залежно від пріоритетів і стратегії системи;

3) система організує обробку переривань, по яких задачі можуть активуватися, деактивуватися та видалятися;

4) після закінчення встановленого кванта часу задача може тимчасово викидатися з пам'яті, віддаючи ресурси іншим задачам, а потім через певний час, що виділений системою, відновлюватися в пам'яті (свопінг);

5) система забезпечує захист пам'яті від несанкціонованого втручання інших задач;

6) система розпізнає збої та зависання окремих задач і припиняє їх;

7) система вирішує конфлікти доступу до ресурсів і пристроїв, не допускаючи тупикові ситуації загального зависання від очікування заблокованих ресурсів;

8) система гарантує кожній задачі, що рано або пізно вона буде активована;

9) система обробляє запити реального часу;

10) система забезпечує комунікацію між процесами.

Труднощі реалізації багатозадачного середовища. Основною трудністю реалізації багатозадачного середовища є її

надійність, що полягає у захисті пам'яті, обробці збоїв і переривань, запобіганні від зависань і тупикових ситуацій.

Окрім надійності, багатозадачне середовище повинне бути ефективним. Витрати ресурсів на її підтримку не повинні: заважати процесам проходити, уповільнювати їхню роботу, різко обмежувати пам'ять.

Псевдопаралельна багатозадачність має декілька типів.

Багатозадачність, що не витискається, – тип багатозадачності, при якому операційна система одночасно завантажує в пам'ять два або більше додатків, але процесорний час надається тільки основному додатку. Для виконання фонового додатку його повинно бути активізовано.

Сумісна або кооперативна багатозадачність – тип багатозадачності, при якому фонові задачі виконуються тільки під час простою основного процесу та лише в тому випадку, якщо на це отриманий дозвіл основного процесу.

Кооперативну багатозадачність можна назвати багатозадачністю «другого рівня» оскільки вона використовує більш передові методи, ніж просте перемикання задач, реалізоване багатьма відомими програмами (наприклад, MS-DOS shell з MS-DOS 5.0). При простому перемиканні активна програма одержує весь процесорний час, а фонові додатки повністю заморожуються. При кооперативній багатозадачності додаток може захопити фактично стільки процесорного часу, скільки він вважає за потрібне. Усі додатки поділяють процесорний час, періодично передаючи керування наступній задачі.

Багатозадачність, що витискає, або пріоритетна багатозадачність (режим реального часу) – тип багатозадачності, у якому операційна система сама передає керування від однієї програми до іншої, що виконується. Розподіл процесорного часу здійснюється планувальником процесів. Цей вид багатозадачності забезпечує більш швидкий відгук на дії користувача.

5.4 Історія багатозадачних операційних систем

Спочатку реалізація багатозадачних операційних систем являла собою серйозну технічну трудність, тому впровадження багатозадачних систем затягувалося, а користувачі довгий час після впровадження вважали за краще однозадачні.

Надалі, після появи декількох вдалих рішень, багатозадачні середовища почали удосконалюватися. Сьогодні використовується всюдно.

Однією з перших багатозадачних систем була OS/360 (1966), використовувана для комп'ютерів фірми IBM та їхніх радянських аналогів ЄС ЕОМ. Розробки системи були дуже затягнуті, і на перших порах фірма IBM випускає однозадачну DOS, щоб задовольнити замовників до повної здачі OS/360 в експлуатацію. Система критикувалася за малу надійність і трудність експлуатації.

У 1969 році з'явилася операційна система UNIX з першим достатньо акуратним алгоритмічним рішенням проблеми багатозадачності. У даний час на базі UNIX створені деякі операційні системи.

На комп'ютерах PDP-11 та їхніх радянських аналогах СМ-4 використовувалася багатозадачна операційна система RSX-11 (радянський аналог ОСРВ) та система розподілу часу TSX-PLUS, що забезпечує обмежені можливості багатозадачності та розрахована на багато користувачів режиму розподілу часу, емулюючи для кожного користувача однозадачну операційну систему RT-11 (радянський аналог РАФОС). Останнє рішення було дуже популярне через низьку ефективність і надійність повноцінної багатозадачної системи.

Акуратним рішенням виявилася операційна система VMS, розроблена спочатку для комп'ютерів VAX (радянський аналог — СМ-1700) як розвиток RSX-11.

Перший у світі мультимедійний персональний комп'ютер Amiga-1000 (1984 р.) спочатку проектувався з розрахунком на повну апаратну підтримку багатозадачності реального часу в операційній системі AMIGA OS, що витісняється. У даному випадку розроблення апаратної та програмної частини велося паралельно, це призвело до того, що за показником квантування багатозадачності (1/50 с на перемикання) AMIGA OS довгий час залишалася неперевершеною.

Основною сучасною багатозадачною операційною системою є Windows фірми Microsoft. При цьому Microsoft вибрала дві лінії розробок — на базі придбаної їй Windows (яка після довгого дороблення системи, спочатку володіючою кооперативною багатозадачністю, аналогічною Mac OS, вилилася у лінійку Windows 95/98/ME) і на основі ідей, закладених у VMS (які привели до створення лінійки операційних систем Windows NT/2000/XP/Vista/7). Використання досвіду VMS забезпечило системам істотно більш високу продуктивність та надійність. За часом перемикання контексту багатозадачності (квантування) тільки ці операційні системи можуть бути порівнянні з AMIGA OS та UNIX (а також з такими його нащадками, як Linux).

5.5 Багатопоточність

Багатопоточність — властивість платформи (наприклад операційної системи) або програми, яка полягає у тому, що процес, породжений в операційній системі, може складатися з декількох потоків, що виконуються «паралельно», тобто без порядку, що вказано в часі. При виконанні деяких задач таке розділення може досягти більш ефективного використання ресурсів обчислювальної машини.

Такі потоки називають також потоками виконання; іноді називають «нитками» (буквальний переклад англ. thread) або неформально «тредами».

Суттю багатопоточності є квазібагатозадачність на рівні одного процесу, що виконується, тобто всі потоки виконуються в адресному просторі процесу. Окрім цього, всі потоки процесу мають не тільки загальний адресний простір, але і загальні дескриптори файлів. Процес, що виконується, має як мінімум один (головний) потік.

Багатопоточність не потрібно плутати ані з багатозадачністю, ані з багатопроцесорністю, не зважаючи на те, що операційні системи, що реалізують багатозадачність, як правило, реалізують і багатопоточність.

До переваг багатопоточності можна віднести таке:

1) спрощення програми в деяких випадках за рахунок використання загального адресного простору;

2) менші, відносно процесу, тимчасові витрати на створення потоку;

3) підвищення продуктивності процесу за рахунок розпаралелювання процесорних обчислень і операцій введення/виведення.

Критика термінології

Переклад англійського терміна thread як «потік» у контексті, пов'язаному з програмуванням, суперечить його ж перекладу «нитка» в загальномовному контексті, а також створює колізії з терміном stream (потік).

Проте термін «потік» пов'язаний з перекладами іноземної технічної літератури, виконаними в 1970-х роках видавництвом «Мир». У наш час в «академічних кругах» (тобто в підручниках, методичних вказівках, ВНЗ-курсах, дисертаціях та ін.) він вважається еталонним. Терміни «нитка», «тред» та інші вважаються технічними жаргонами.

Контрольні питання

1 Яку функцію виконує ЦП?
2 Реальний режим роботи ЦП.
3 Захищений режим роботи ЦП.
4 Навіщо потрібна багатозадачність?
5 Які властивості багатозадачності?
6 Як розвивались багатозадачні операційні системи?
7 Які особливості багатопоточності?

6 Кеш-пам'ять

На сучасному етапі розвитку обчислювальної техніки склалася ситуація, коли швидкість роботи мікропроцесора вища, а іноді значно вища за швидкість роботи відносно дешевої динамічної пам'яті (оперативного пристрою, що запам'ятовує

(ОЗП)). Тому на певному етапі розвитку була задіяна більш дорога й технологічна статична пам'ять (кеш).

Кеш (англ. *cache*) — проміжний буфер з швидким доступом, що містить копію тієї інформації, яка зберігається у пам'яті з менш швидким доступом, але з щонайбільшою вірогідністю може звідти бути запитана. Доступ до даних у кеші де швидше, ніж вибірка початкових даних з повільної пам'яті або їх переобчислювання, що робить середній час доступу коротшим.

Уперше слово «кеш» у комп'ютерному контексті було використано у 1967 році під час підготовки статті для публікації в журналі «IBM Systems Journal». Стаття стосувалася удосконалення пам'яті, що розробляється для комп’ютерів із серії IBM System/360. Редактор журналу Лайл Джонсон попросив придумати більш описовий термін, ніж «високошвидкісний буфер», але за відсутності ідей сам запропонував слово «кеш». Стаття була опублікована на початку 1968 році, автори були премійовані IBM, їхня робота отримала розповсюдження та згодом була поліпшена, а слово «кеш» незабаром стало використовуватися в комп'ютерній літературі як загальноприйнятий термін.

6.1 Функціонування кеша

Кеш – це пам'ять з більшою швидкістю доступу, призначена для прискорення звернення до даних, що містяться постійно в пам'яті з меншою швидкістю доступу (далі «основна пам’ять»). Кешування застосовується ЦПУ, жорсткими дисками, браузерами та веб-серверами.

Кеш складається з набору записів. Кожний запис асоційований з елементом даних або блоком даних (невеликої частини даних), яка є копією елемента даних в основній пам'яті. Кожний запис має ідентифікатор, що визначає відповідність між елементами даних у кеші та їхніми копіями в основній пам'яті.

Коли клієнт кеша (ЦПУ, браузер, операційна система) звертається до даних, перш за все досліджується кеш. Якщо в кеш знайдено запис з ідентифікатором, який збігається з ідентифікатором елемента даних, що запитувався, то використовуються елементи даних з кеша. Такий випадок

називається **попаданням у кеш**. Якщо в кеш не знайдено записи, що містять елемент даних, що запитувався, то він читається з основної пам'яті в кеш та стають доступним для подальших звертань. Такий випадок називається **промахом кеша**. Відсоток звертань до кеша, коли в ньому знайдений результат, називається **рівнем попадань** або **коефіцієнтом попадань** у кеш.

Наприклад, браузер перевіряє локальний кеш на диску на наявність локальної копії веб-сторінки, відповідної запиту URL. У даному прикладі URL – це ідентифікатор, а вміст веб-сторінки – це елементи даних.

Якщо кеш обмежений в обсязі, то при промаху може бути ухвалено рішення відкинути деякий запис для звільнення простору. Для вибору відкинутого запису використовується так званий **алгоритм витіснення.**

При модифікації елементів даних у кеші, виконується їхнє оновлення в основній пам'яті. Затримка в часі між модифікацією даних у кеші та оновленням основної пам'яті керується так званою **політикою запису**.

У кеші з **негайним записом** кожна зміна викликає синхронне оновлення даних в основній пам'яті.

У кеші з **відкладеним записом** (або **зворотним записом**) оновлення відбувається у разі витіснення елемента даних, періодично або за запитом клієнта. Для відстежування модифікованих елементів даних записи кеша зберігають ознаки модифікації (**змінений** або **«брудний»**). Промах у кеші з відкладеним записом може запитати два звернення до основної пам'яті: перше для запису замінюваних даних з кеша, друге для читання необхідного елемента даних.

У випадку, якщо дані в основній пам'яті можуть бути змінені незалежно від кеша, то запис кеша може стати **неактуальним**. Протоколи взаємодії між кешем, які зберігають узгодженість даних, називають **протоколами когерентності кеша**.

6.2 Кеш центрального процесора

Ряд моделей ЦП мають власний кеш, для того щоб мінімізувати доступ до ОЗП, яка повільніша, ніж регістри. Кеш-пам'ять може давати значний виграш у продуктивності у разі, коли тактова частота ОЗП значно менша за тактову частоту ЦП. Тактова частота для кеша звичайно не набагато менше ніж частота ЦП.

Кеш центрального процесора розділений на декілька рівнів. Для універсальних процесорів – до 3. Кеш-пам'ять рівня N+1, як правило, більша за розміром та повільніша за швидкістю звертання та передачі даних, ніж кеш-пам'ять рівня N.

Найшвидшою пам'яттю є кеш першого рівня – L1-кеш. По суті він є невід'ємною частиною процесора, оскільки розташований на одному з ним кристалі та входить до складу функціональних блоків. Складається з кеш-команд і кеш-даних. Деякі процесори без L1-кеша не можуть функціонувати. На інших його можна відключити, але тоді значно знижується продуктивність процесора. L1-кеш працює на частоті процесора та в загальному випадку звернення до нього може проводитися кожний такт (часто є можливим виконувати навіть декілька читань/записів одночасно). Латентність доступу звичайно рівна 2-4 тактам ядра. Обсяг звичайно невеликий – не більше 128 КБ.

Другий за швидкодією є L2-кеш другого рівня. Звичайно він розташований або на кристалі, як L1, або в безпосередній близькості від ядра, наприклад, у процесорному картриджі (тільки у слотових процесорах). У старих процесорах – набір мікросхем на системній платі.

Обсяг L2-кеша від 128 кБ до 128 МБ. У сучасних багатоядерних процесорах кеш другого рівня, знаходячись на тому ж кристалі, є пам'яттю роздільного використання – при загальному обсязі кеша в 8 МБ на кожне ядро доводиться по 2 Мб. Звичайно латентність L2-кеша, розташованого на кристалі ядра, складає від 8 до 20 тактів ядра. На відміну від L1-кеша, його відключення може не вплинути на продуктивність системи. Проте у задачах, що пов'язані з чисельними зверненнями до обмеженої області пам'яті, наприклад, СУБД, продуктивність може знизитися в десятки разів.

Кеш третього рівня якнайменш швидкодійний та звичайно розташований окремо від ядра ЦП, але він може бути дуже

значного розміру – більше 32 МБ. L3-кеш повільніший за попередні кеші, але все одно значно швидший, ніж ОЗП. У багатопроцесорних системах він знаходиться в загальному користуванні.

Відключення кешів другого та третього рівнів звичайно використовується в математичних задачах, наприклад, при розрахунках полігонів, коли обсяг даних менший за розмір кеша. У цьому випадку можна відразу записати всі дані в кеш, а потім проводити їхню обробку.

Одна з фундаментальних характеристик кеша – рівень асоціативності – відображає її логічну сегментацію. Річ у тому, що послідовний перебір усіх рядків кеша у пошуках необхідних даних потребував би десятки тактів та звів би нанівець весь виграш від використання вбудованої в ЦП пам'яті. Тому осередки ОЗП жорстко прив'язуються до рядків кеш-пам'яті (в кожному рядку можуть бути дані з фіксованого набору адрес), що значно скорочує час пошуку. З кожним осередком ОЗП може бути пов'язано більше одного рядка кеш-пам'яті: наприклад, *n-канальна* асоціативна (англ. *associative*) позначає, що інформація за деякою адресою оперативної пам'яті може зберігатися в *n* місцях кеш-пам'яті.

При однаковому обсязі кеша схема з більшою асоціативністю буде якнайменше швидкою, але найефективнішою.

6.3 Кешування, що виконується операційною системою

Кеш оперативної пам'яті складається з таких елементів:

1) набір сторінок оперативної пам'яті, розділених на буфери, рівні по довжині блоку даних відповідного пристрою зовнішньої пам'яті;

2) набір заголовків буферів, що описують стан відповідного буфера;

3) кеш-таблиці, що містять відповідність номера блоку заголовка;

4) список вільних буферів.

Алгоритм роботи кеша з відкладеним записом

Спочатку всі заголовки буферів поміщаються в список вільних буферів. Якщо процес має намір прочитати або модифікувати блок, то він виконує такий алгоритм:

1) намагається знайти в кеш-таблиці заголовок буфера із заданим номером;

2) у випадку, якщо отриманий буфер зайнятий, чекає його звільнення;

3) у випадку, якщо буфер не знайдений у кеш-таблиці, бере перший буфер з хвоста списку вільних;

4) у випадку, якщо список вільних буферів порожній, то виконується алгоритм витіснення (див. нижче);

5) у випадку, якщо отриманий буфер помічений як «брудний», виконує асинхронний запис буфера, що міститься у зовнішній пам'яті;

6) видаляє буфер з кеш-таблиці, якщо він був поміщений у неї;

7) поміщає буфер у кеш-таблицю з новим номером.

Процес читає дані в отриманий буфер і звільняє його. У разі модифікації процес перед звільненням позначає буфер як «брудний». При звільненні буфер поміщається в голову списку вільних буферів.

Таким чином, якщо процес прочитав деякий блок у буфері, то велика вірогідність, що інший процес при читанні цього блоку знайде буфер в оперативній пам'яті. Запис даних у зовнішню пам'ять виконується тільки тоді, коли не вистачає «чистих» буферів, або за запитом.

Алгоритм витіснення

Якщо список вільних буферів порожній, то виконується алгоритм витіснення буфера. Алгоритм витіснення істотно впливає на продуктивність кеша. Існують такі алгоритми:

1) LRU (Least Recently Used) — витісняється буфер, невикористаний довше за всіх;

2) MRU (Most Recently Used) — витісняється останній використаний буфер;

3) LFU (Least Frequently Used) — витісняється буфер, використаний рідше за всіх;

4) ARC (Advanced Replacement Cache) — алгоритм витіснення, що комбінує LRU і LFU, запатентований IBM.

Застосування того або іншого алгоритму залежить від стратегії кешування даних. LRU найбільш ефективний, якщо дані гарантовано будуть повторно використані найближчим часом. MRU найбільш ефективний, якщо дані гарантовано не будуть повторно використані найближчим часом. У випадку, якщо додаток явно вказує стратегію кешування для деякого набору даних, то кеш буде функціонувати найбільш ефективно.

6.4 Інші види кешування

6.4.1 Кешування зовнішніх накопичувачів

Багато периферійних пристроїв зберігання даних використовують кеш для прискорення роботи, зокрема жорсткі диски використовують кеш-пам'ять від 1 до 64 Мб (моделі з підтримкою NCQ/TCQ використовують її для зберігання та обробки запитів), пристрої читання CD/DVD/BD-дисків так само кешують прочитану інформацію для прискорення повторного звертання. Операційна система так само використовує частину оперативної пам'яті як кеш дискових операцій (у тому числі для зовнішніх пристроїв, що не мають власної кеш-пам'яті, у тому числі жорстких дисків, flash-пам'яті і гнучких дисків).

Застосування кешування зовнішніх накопичувачів обумовлено такими чинниками:

1) швидкість доступу процесора до оперативної пам'яті у багато разів більша, ніж до пам'яті зовнішніх накопичувачів;

2) деякі блоки пам'яті зовнішніх накопичувачів використовуються декількома процесами одночасно та має сенс прочитати блок один раз, потім берегти одну копію блоку в оперативній пам'яті для всіх процесів;

3) доступ до деяких блоків оперативної пам'яті відбувається набагато частіше, ніж до інших, тому використання кешування для таких блоків у цілому збільшує продуктивність системи;

4) для деяких блоків пам'яті зовнішніх накопичувачів не вимагається безпосереднього запису після модифікації та

використання кеш для таких блоків оптимізує використання введення-виведення.

6.4.2 Програмне кешування

При читанні даних кеш-пам'ять дає однозначний виграш у продуктивності. При записі даних виграш можна отримати тільки ціною зниження надійності. Тому в різних додатках може бути вибрана та або інша політика запису кеш-пам'яті.

Існують дві основні політики запису кеш-пам'яті — крізний запис (write-through) і відкладений запис (write-back).

Крізний запис має на увазі, що при зміні вмісту елемента пам'яті запис відбувається синхронно в кеш і в основну пам'ять.

Відкладений запис має на увазі, що можна відкласти момент запису даних в основну пам'ять, а записати їх тільки в кеш. При цьому дані будуть вивантажені в оперативну пам'ять тільки у разі звернення до них іншого пристрою (інший ЦП, контролер DMA) або браку місця в кеш для розміщення інших даних. Продуктивність у порівнянні з крізним записом підвищується, але це може поставити під загрозу цілісність даних в основній пам'яті, оскільки програмний або апаратний збій може привести до того, що дані так і не будуть переписані з кеша в основну пам'ять. Крім того, у разі кешування оперативної пам'яті, коли використовуються два та більше процесорів, потрібно забезпечувати узгодженість даних у різних кешах.

Контрольні питання

1 Яку функцію виконує кеш?

2 Політика безпеки кеша.

3 Кешування ЦП та алгоритми кешування.

4 Кешування операційної системи та алгоритми кешування.

5 Кешування зовнішніх накопичувачів та алгоритми кешування.

6 Кешування програм та алгоритми кешування.

7 Технологія мультимедіа

Фактично вся історія розвитку комп'ютерів являє собою безперервну гонку між швидкодією центрального процесора й інших систем – пам'яті та зовнішніх пристроїв. Особливо це помітно в системах мультимедіа, де йде обробка звуку та зображення, цифрове представлення яких займає великі обсяги пам'яті. Для ефективної обробки звуку та відео при низькій пропускній спроможності системної магістралі (шини) все більша кількість функцій переноситься в апаратуру – модеми, відео- та звукові адаптери. Це викликає їхнє помітне дорожчання порівняно із загальною вартістю комп'ютера, що особливо неприємно в обстановці швидкого морального старіння всієї комп'ютерної апаратури.

Ця проблема стала актуальною на початку 1990-х років, коли ПК став доступний широким масам користувачів і все активніше почав перетворюватися на засіб розваг. Першим процесором, що відчув брак ресурсів для мультимедійних додатків по тому часу, став Pentium.

Насправді нездатність ПК з процесором Pentium ефективно обробляти в реальному часі звук і відео без спеціальних карт відбувається уже не стільки від загальної швидкодії процесора або шини, які в більшості випадків цілком достатні, а від характеру його набору команд обробки даних, відомого під назвою CISC. Цей набір, що складається з відносно складних арифметико-логічних команд, орієнтований на типові задачі обробки даних, без спеціального «заточування» під особливі додатки. Ця вигідна для більшості додатків архітектура виявляється цілком неефективною при швидкісній та специфічній обробці великих масивів даних, оскільки складна система команд використовується на лічені відсотки, а невигідні витрати складають десятки та сотні відсотків.

Технологія MMX являє собою компромісне рішення, об'єднуючи шляхи, використовувані в комп'ютерах SPARC і Silicon Graphics (технологія RISC - Reduced Instruction Set Computer, комп'ютер із спрощеним набором команд), а також у комп'ютерах з паралельною архітектурою (технологія SIMD: Single Instruction, Multiple Data - одна команда, багато даних):

класичний процесор Pentium (CISC) з додаванням ряду простих (RISC) команд паралельної обробки даних (SIMD).

Технологія MMX

Абревіатура MMX походить від виразу Multi Media eXtension - розширення для мультимедіа, яке реалізоване фірмою Intel у своїй серії процесорів MMX з тактовою частотою 166 МГц і більше. Історично склалося так, що майже будь-яке нове рішення у галузі персональних комп'ютерів широко рекламується та підноситься як епохальне, обіцяючи небачений досі розквіт комп'ютерним технологіям, проте всі ми пам'ятаємо, скільки разів подібний галас обертався скромним реальним ефектом.

Процесор Pentium MMX відрізняється від «звичайного» Pentium за шістьма основними пунктами:

1) додано 57 нових команд обробки даних;

2) збільшений вдвічі обсяг внутрішнього кеша (16 кб для команд і стільки ж - для даних);

3) збільшений обсяг буфера адрес переходу (Branch Target Buffer - BTB), використовуваного в системі прогнозу переходів (Branch Prediction);

4) оптимізовано роботу конвеєра (Pipeline);

5) збільшено кількість буферів запису (Write Buffers);

6) введено так зване подвійне електроживлення процесора.

Набір з 57 нових команд і є основною відмінністю; інші два - не більше, ніж супутні зміни. Хоча збільшений обсяг кеш і внутрішніх буферів та оптимізований конвеєр дещо прискорюють роботу будь-яких додатків, проте основне збільшення продуктивності (до 60 %) можливе тільки при використанні програм, що правильно застосовують технологію MMX в обробці даних.

Обробка даних в MMX

Як уже говорилося, в Pentium MMX додано 57 нових команд обробки даних і відповідно – чотири нові типи даних. За одну операцію команда MMX обробляє 64-розрядне двійкове слово (так зване квадраслово, або QWORD). Нові типи даних утворюються від упаковування в квадраслово звичайних типів – байтів (по 8), слів (по 4) або подвійних слів (по 2). Четвертий тип являє собою саме квадраслово.

Таким чином, одна елементарна MMX-операція має справу або з одним квадрасловом, що схожа на звичайну операцію великої розрядності, або з двома подвійними словами, чотирма словами або вісьма байтами, причому виконання відбувається одночасно та кожний елемент даних обробляється незалежно від інших. Подібні групові операції переважають під час обробки зображення (групи точок) та звуку (групи значень амплітуди).

Набір MMX-команд складається з команд пересилання даних, упаковування/розпаковування, додавання/віднімання, множення, зсуву, порівняння та порозрядних логічних. Команди упаковування та додавання/віднімання можуть працювати у двох режимах: звичайному, коли переповнювання розрядної сітки викликає «завертання» (wraparound) значення результату, та спеціальному, коли воно приводить до обмеження (clipping) результату до мінімально або максимально допустимого значення. Режим обмеження в термінології Intel називається Saturation (змішування) – у ньому особливо зручно виконувати змішування кольорів зображення або амплітуд звукових сигналів, оскільки при звичайному переповнюванні результат не має ніякого сенсу.

Команда множення представлена трьома видами: перші два виконують попарне множення чотирьох слів з вибором або старшої, або молодшої частини результату, а третій виконує операцію виду (ab + cd) для кожної пари з чотирьох слів операндів, що дуже зручно при обчисленні математичних рядів.

Команди зсуву реалізують логічний та арифметичний зсуви своїх операндів (арифметичний зсув відрізняється від логічного тим, що при зсуві розряди, що вправо звільнилися, заповнюються копією знакового розряду, а не нулями, тому він придатний для множення/розподілу знакових операндів на степені двійки). Логічні порозрядні команди виконують операції «І» (AND), «АБО» (OR), «виключає АБО» (XOR), а також комбіновану команду «І» з інверсією одного з операндів (AND NOT), зручну для реалізації «зворотного вибору» по бітовій масці.

Команди порівняння працюють дещо незвичайно в порівнянні із загальноприйнятою логікою: замість установлення ознак для подальших команд переходу вони генерують одиничні бітові маски для тих операндів, які задовольняють умову, і

нульові – для інших операндів. Подальші логічні порозрядні операції можуть виділити, погасити або якось інакше обробити відзначені таким чином операнди, які в цьому випадку можуть являти собою точки зображення або відліки звукового сигналу.

Особливості реалізації MMX

Для обробки даних і зберігання проміжних результатів у Pentium MMX використовуються вісім 64-розрядних регістрів MM0..MM7, які фізично суміщені із стеком регістрів математичного співпроцесора. При виконанні будь-якої з MMX-команд відбувається установлення «режиму MMX» з відміткою цього в слові стану співпроцесора (FPU Tag Word). З цієї миті стек регістрів співпроцесора розглядається як набір MMX-регістрів; завершує роботу в режимі MMX команда EMMS (End MULTIMEDIA State). З одного боку, така реалізація дозволила забезпечити нормальну роботу додатків, що використовують MMX, у багатозадачних системах, що не підтримують цю технологію, оскільки всі подібні системи створюють власну копію стека співпроцесора, що містить слова його стану для кожного процесу. З іншого боку, перехід між режимами займає значний час та поєднання. Наприклад, в одному циклі команд співпроцесора з командами MMX може не тільки не прискорити, а навіть істотно уповільнити виконання програми. Тому для досягнення якнайкращих результатів рекомендується групувати ці команди окремо одна від іншої, що насправді не складно.

Продуктивність MMX

Оскільки MMX – достатньо вузькоспеціалізоване розширення системи команд процесора, не можна чекати кардинального прискорення роботи тільки від самого факту переходу на процесор MMX. Як уже було сказано, на додатках загального характеру, незнайомих з MMX, реальна продуктивність зростає лише на одиниці відсотків, хоча тести можуть показувати її зростання на 20-30%, це відбувається через циклічність більшості тестів, коли велика частина циклу попадала в збільшений внутрішній кеш.

При використанні «чистого» MMX-коду, вдало відповідного до специфіки вирішуваної задачі, швидкодія переписаної ділянки може зрости в 5-6 разів, проте це

прискорення буде локальним і неминуче компенсується «типовими» ділянками програми, тому не потрібно відразу ж чекати від програм, що використовують MMX, прискорення роботи в рази. За максимальними результатами тестів Intel Media Benchmark і Norton Media Benchmark для Windows 95 обробка зображень з використанням технології MMX відбувається швидше майже в п'ять разів, проте в середньому виходить приблизно 1,5..3-кратне прискорення.

До речі, одним з класів програм, яким використання MMX допомагає, є ігри; проте уже давно не секрет, що зараз комп'ютер, «достатній для ігор», у багатьох випадках істотно перевершує за складністю та вартістю «достатній для роботи», бо сучасні ігри близькі за структурою до складних операційних систем реального часу. Тому всі без винятку ігри, що застосовують анімацію та звук, підтримують (а багато хто вимагає) технологію MMX та її нащадків.

MMX (MultiMedia Extensions — мультимедійні розширення) — комерційна назва додаткового набору інструкцій, що виконують характерні для процесів кодування/декодування аудіо/відео потоків даних та дії за одну машинну інструкцію. Уперше з'явився в процесорах Pentium MMX. Розроблений в лабораторії Intel в Хайфі (Ізраїль) у першій половині 1990-х років.

Регістри MMX. Розширення MMX включає вісім 64-бітних регістрів загального користування MM0-MM7. Фізично ніяких нових регістрів з введенням MMX не з'явилося. MM0-MM7 — це в точності мантиси восьми регістрів FPU (математичний співпроцесор) від R0—R7. Таким чином, не можна одночасно користуватися командами математичного співпроцесора та MMX.

Типи даних MMX. Команди технології MMX працюють з 64-розрядними цілочисловими даними, а також з даними, упакованими в групи (вектори) загальною довжиною 64 біт. Такі дані можуть знаходитися в пам'яті або у восьми MMX-регістрах. Команди технології MMX працюють з такими типами даних:

1) упаковані байти (вісім байтів в одному 64-розрядному регістрі);

2) упаковані слова (чотири 16-розрядні слова в 64-розрядному регістрі) (packed word);

3) упаковані подвійні слова (два 32-розрядні слова в 64-розрядному регістрі) (packed doubleword);

4) 64-розрядні слова (quadword).

3DNow! – додаткове розширення MMX для процесорів AMD, починаючи з AMD K6 3D. Причиною створення 3DNow! послужило прагнення завоювати перевагу над процесорами виробництва компанії Intel у галузі обробки мультимедійних даних. Хоча це розширення є розробкою AMD, його також інтегрували у свої процесори IBM, Cyrix та ін. Технологія 3DNow! запровадила 21 нову команду процесора та можливість оперувати 32-бітними речовинними типами в стандартних MMX-регістрах. Також були додані спеціальні інструкції, що оптимізують перемикання в режим MMX/3DNow! (femms, яка замінювала стандартну інструкцію emms) та роботу з кешем процесора. Таким чином, технологія 3DNow! розширювала можливості технології MMX, не вимагаючи введення нових режимів роботи процесора і нових регістрів.

SSE (англ. Streaming SIMD Extensions, потокове SIMD розширення процесора) — SIMD (англ. Single Instruction, Multiple Data, одна інструкція — множина даних) набір інструкцій, розроблений Intel і вперше представлений у процесорах серії Pentium III як відповідь на аналогічний набір інструкцій 3DNow! від AMD, який був представлений роком раніше. Спочатку назвою цих інструкцій було KNI що розшифровувалося як Katmai New Instructions (Katmai – назва першої версії ядра процесора Pentium III).

Технологія SSE дозволяла подолати дві основні проблеми MMX — при використанні MMX неможливо було одночасно використовувати інструкції співпроцесора (оскільки його регістри використовувалися для MMX) та працювати з дійсними числами.

SSE включає в архітектуру процесора вісім 128-бітних регістрів (xmm0-xmm7), кожний з яких виглядає як 4 послідовні значення з плаваючою точкою одинарної точності. SSE включає набір інструкцій, який проводить операції і скалярними та упакованими типами даних.

Перевага в продуктивності досягається у тому випадку, коли необхідно провести одну й ту саму послідовність дій над різними даними. Реалізація блоків SIMD здійснюється розпаралелюванням обчислювального процесу між даними. Через один блок проходить по черзі множина потоків даних.

SSE2 (англ. Streaming SIMD Extensions 2, SIMD-розширення потоку процесора) – це SIMD (англ. Single Instruction, Multiple Data – одна інструкція – множина даних) набір інструкцій, розроблений Intel та представлений у процесорах Pentium 4.

SSE2 використовує вісім 128-бітних регістрів (xmm0-xmm7), увімкнених в архітектуру x86 з введенням розширення SSE, кожний з яких представляє як 2 послідовні значення з плаваючою комою подвійної точності. SSE2 включає набір інструкцій, який проводить операції зі скалярними та упакованими типами даних. Також SSE2 містить інструкції для обробки потоку цілочислових даних у тих же 128-бітних регістрах xmm, що робить це розширення більш переважним для цілочисельних обчислень, ніж використання набору інструкцій MMX, що з'явився набагато раніше.

Перевага в продуктивності досягається у тому випадку, коли необхідно провести одну й ту саму послідовність дій над великим набором однотипних даних.

SSE3 (**PNI** — Prescott New Instruction) — третя версія SIMD-розширення Intel, нащадок SSE, SSE2 для x86. Вперше представлено 2 лютого 2004 році в ядрі Prescott процесора Pentium 4. У 2005 році AMD запропонувала свою реалізацію SSE3 для процесорів Athlon 64 (ядра Venice та San Diego).

Найпомітніша зміна – можливість горизонтальної роботи з регістрами. Якщо говорити більш конкретно, додані команди додавання та віднімання декількох значень, що зберігаються в одному регістрі. Ці команди спростили ряд DSP і 3D-операцій. Існує також нова команда для перетворення значень з плаваючою комою в ціле без необхідності вносити зміни в глобальному режимі округлення.

Процесори з підтримкою SSE3: Athlon 64; Athlon 64 X2 ; Athlon 64 FX; Opteron; Sempron; Pentium 4 Prescott; Intel Core 2.

SSSE3 (Supplemental Streaming SIMD Extension 3) — це позначення було дано Intel 4-му розширенню системи команд. Попереднє мало позначення SSE3 і Intel додав ще один символ 'S' замість того, щоб збільшити номер розширення, можливо тому, що вони порахували SSSE3 простим доповненням до SSE3. Часто, до того як почало використовуватися офіційне позначення SSSE3, ці нові команди називалися SSE4. Також їх називали кодовими іменами Tejas New Instructions (TNI) і Merom New Instructions (MNI) за назвою процесорів, де вперше Intel мала намір підтримати ці нові команди. З'явившись у Intel Core Microarchitecture, SSSE3 доступно у серіях процесорів Xeon 5100 (Server і Workstation версії), а також у процесорах Intel Core 2 (Notebook і Desktop версії).

Новими в SSSE3, у порівнянні з SSE3, є 16 унікальних команд, що працюють з упакованим цілим. Кожна з них може працювати як з 64-бітними (MMX), так і з 128-бітними (XMM) регістрами, тому Intel у своїх матеріалах посилається на 32 нових команд.

Процесори, що підтримують SSSE3: Xeon 5100 Series; Intel Core 2; Intel Celeron (ядро CONROE-L).

SSE4 це новий набір команд Intel Core мікроархітектури, вперше реалізований у процесорах серії Penryn (не потрібно плутати з SSE4A від AMD). Було анонсовано 27 вересня 2006 року, проте детальний опис став доступний тільки навесні 2007 року, свіжий опис для програмістів можна знайти на сайті Intel. SSE4 складається з 54 інструкцій, 47 з них відносять до SSE4.1 (вони є тільки в процесорах Penryn). Очікується, що повний набір команд (SSE4.1 і SSE4.2, тобто 47 + 7 команд, що залишилися) буде доступний у процесорах Nehalem. Жодна з SSE4 інструкцій не працює з 64-бітними mmx-регістрами (тільки з 128-бітними xmm0-15).

Список літератури

1 Аппаратные средства IBM PC: энциклопедия / М.Ю.Гук. – 3-е изд. – СПб.: Питер, 2006. – 1072 с.

2 Архитектура ЭВМ и вычислительных систем: учебник / Н.В. Максимов, Т.Л. Партыка, И.И. Попов. – М.: Форум, 2006. – 512 с.

3 Модернизация и ремонт ПК: пер. с англ. / С. Мюллер. – М.: Вильямс, 2007. – 1360 с.

4 Практическая сборка и наладка ПК: пер. с англ. / О.С. Степаненко. – М.: Вильямс, 2007. – 336 с.

5 Ремонт и модернизация ПК: пер. с англ. / Б. Томпсон, Р.Томпсон. – М.: BHV, 2007. – 608 с.

6 *BIOS* / А. Трасковский. – М.: BHV, 2006. – 400 с.

7 Асемблер: самоучитель./ А. Крупник. – СПб.: Питер, 2005. –235 с.

Додаток А

А.1 Дослідження структури системної плати

Теоретична підготовка

Системна (system board), або материнська (mother board) плата – основний компонент ПК. На системній платі розташовуються:

1) центральний процесор;

2) BIOS (базова система введення-виведення);

3) оперативна пам'ять – ОЗП (Random Access Memory);

4) інтерфейси накопичувальних пристроїв, послідовних і паралельних портів;

5) роз'єми живлення;

6) контролери (чипсети), призначені для організації обміну інформацією між системною платою та периферійними пристроями (монітором, мишею, клавіатурою та дисковими накопичувачами).

Чипсет (Chipset) – система управління із спеціально розроблених мікросхем, що служать для організації роботи центрального процесора (рисунок А.1). Чипсет – набір логіки, який складається з однієї або декількох мікросхем. Мікросхеми містять у собі контролери прямого доступу до пам'яті, контролери переривань, схеми управління пам'яттю та шинами, контролери зовнішніх пристроїв.

Чипсет визначає функціональні можливості системної плати: типи підтримуваних процесорів, можливі поєднання типів та обсяг модулів пам'яті, кількість і типи слотів розширення, можливість програмної настройки параметрів та ін. На одному й тому ж наборі можуть випускатися декілька моделей системних плат. Функції, підтримувані тим або іншим чипсетом, указані в характеристиках плати, побудованої на його основі.

Корпорації Intel, AMD та VIA розробляють несумісні за технічними характеристиками процесори, що привело до специфікації чипсетів: вони створюються під конкретний тип процесора.

Основні фірми-виробники чипсетів для процесорів Intel: 1) Intel; 2) VIA; 3) SIS; 4) ATI.

Виробники чипсетів для процесорів AMD: 1) VIA; 2) SIS; 3) NVIDIA; 4) ATI; 5) AMD.

Кожний виробник для маркування мікросхем чипсетів розробляє власну буквено-цифрову систему.

Базові мікросхеми сучасного чипсета отримали назви South Bridge (південний міст) та North Bridge (північний міст). Назви умовні та походять від розташування мікросхем на структурних схемах. Основні функції мікросхеми північного моста – обмін даними між процесором і високошвидкісними пристроями (пам'ять, інтегрована графіка, шини AGP, PCI-EXPRESS). Мікросхема південного моста призначена для організації роботи з низькошвидкісними інтерфейсами та пристроями (інтерфейси IDE, Serial ATA, PCI, USB, інтегрований звук, RAID-контролери та ін.).

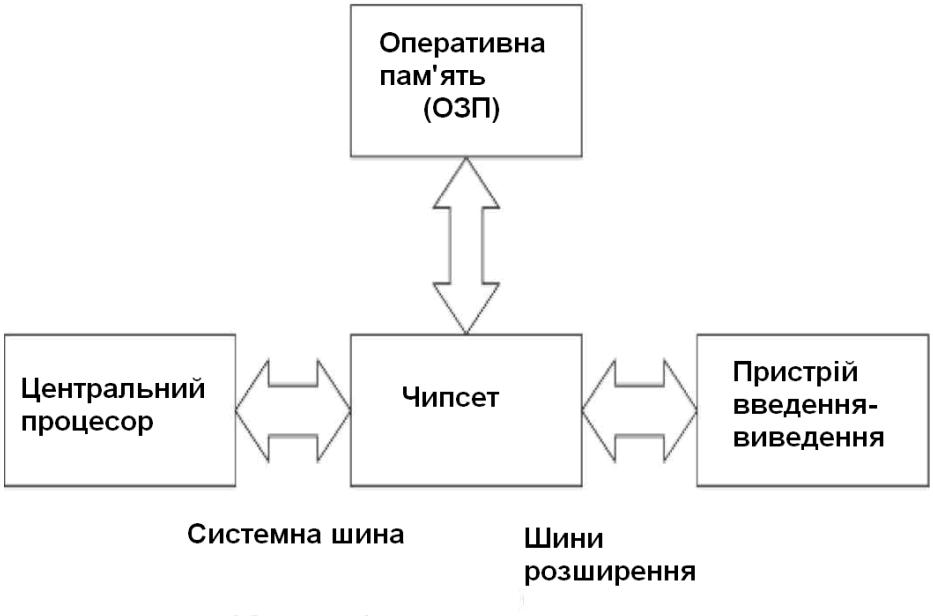


Рисунок А.1 – Узагальнена структурна схема системної плати

Обмін інформацією між північним і південним мостами здійснюється різними типами швидкісних шин. На рисунку А.2 наведено приклад системної плати. Ефективна робота центрального процесора пов'язана з обміном даних між ним, мікросхемою чипсета та ОЗП. Для управління певним типом пам'яті необхідно, щоб чипсет системної плати зміг працювати з нею.

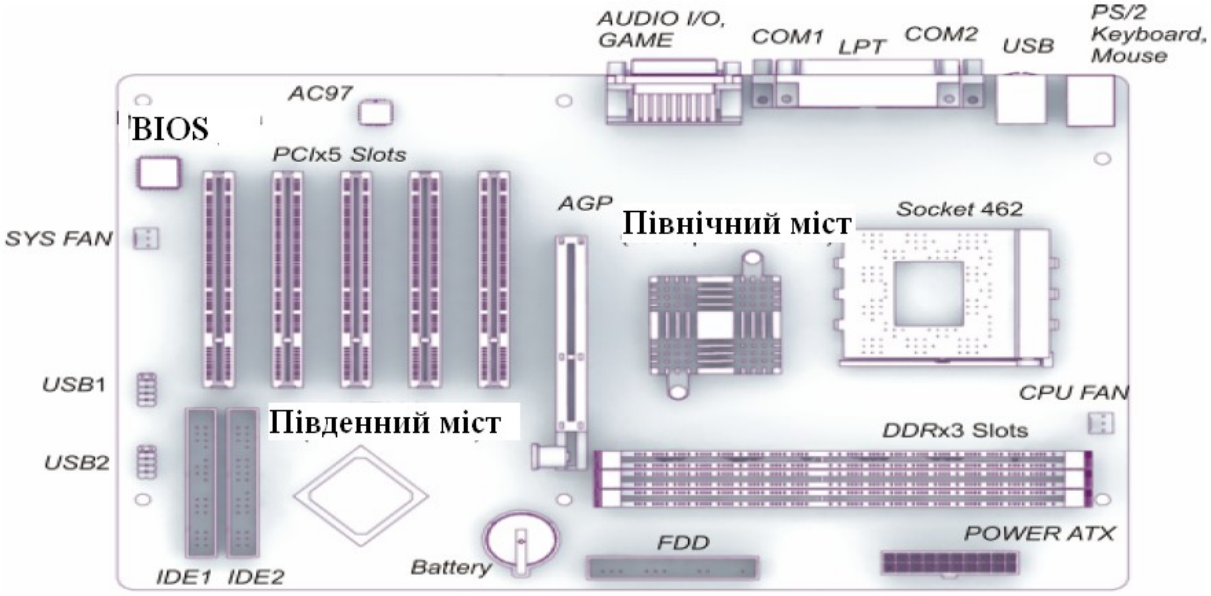


Рисунок А.2 – Розташування елементів на системній платі

Для розширення функцій ПК, вимикання зовнішніх пристроїв введення-виведення на системній платі встановлюються роз'єми, які називають слотами розширення.

У даний час на системній платі для масового використання встановлюються такі типи слотів: *PCI, PCI Express, AGP* та один з варіантів роз'єму для аудіопристроїв. Для слоту *PCI Express* існують декілька варіантів виконання: *x16,* x8 або xl.

BIOS (Basic INPUT-OUTPUT System) – базова система введення-виведення, що розміщується на мікросхемі ПЗП (ROM-Read Only Memory). Вбудоване програмне забезпечення BIOS доступне для ПЕОМ без звернення до дискових накопичувачів. BIOS містить коди, необхідні для управління клавіатурою, відеокартою, дисками, портами та іншими пристроями.

Практична частина

Ознайомитися з основними вузлами та принципами роботи системної плати комп'ютерів на платформі Intel. Ознайомитися з конструктивними особливостями реалізації системної плати, можливостями їхнього використання у складі ПЕОМ.

Вивчити видану системну плату та виконати аналіз за планом: стандарт, основні елементи (набори мікросхем, слоти розширення, додаткові контролери, задаючі генератори, мікросхеми BIOS та ін.), процесорний сокет та його оточення, роз'єми для пимикання джерела живлення, перетворювач напруги.

Виходячи з узагальненої структури системної плати (рисунок А.2), розробити структуру взаємодії між окремими складовими плати.

Визначити базові відмінності системної плати, включаючи конструктивні особливості та результати тестів.

На підставі проведеного аналізу виробити рекомендації щодо використання системної плати у складі ПЕОМ для розв’язання прикладних задач.

А.2 Дослідження BIOS

Теоретична підготовка

Як відомо, до операційної системи в комп'ютері запускається вбудована в мікросхему материнської плати програма BIOS (Base Input/Output System, основна система введення-виведення). Призначення цього невеликого (256 кб) програмного коду – звести до «загального знаменника» апаратні відмінності комп'ютерного обладнання.

Надійна та ефективна робота ПК неможлива без правильно конфігурованого BIOS. Конфлікт між новітнім обладнанням і застарілим кодом чипа – річ досить часта. У такому разі вихід один: перепрошиття. Але якщо заміна BIOS вимагає певних навичок і знань (без яких вона перетворюється на електронний варіант «російської рулетки»), то первинна настройка цілком під силу середньому користувачу. Більше того розуміння правил вмикання комп'ютера необхідне для грамотного його використання.

BIOS являє собою програму, записану в мікросхему за тією або іншою технологією ROM і не вимагає живлення, для того щоб зберігатися там навіть після вимкнення комп'ютера. У сучасних чипах використовується технологія Flash ROM, що дозволяє перезаписувати BIOS без додаткового обладнання. Але, у кожному разі, перезапис BIOS – операція серйозна. Інша справа – настройка (SETUP) BIOS. Настройка дозволяє підігнати стандартну версію BIOS під конфігурацію комп'ютера або ОС. Змінювати настройку BIOS можна хоч по кілька разів за день.

Параметри настройки BIOS зберігаються в енергозалежній CMOS RAM, яка живиться від батареї на материнській платі. Звідси висновок: якщо «злітають» установки комп'ютера або збивається годинник – швидше за все пора змінювати батарею.

Роль BIOS при завантаженні ПК

Після вмикання живлення напруга подається на ЦПП та інші мікросхеми материнської плати. «Прокинувшись», ЦПП запускає з мікросхеми (рисунок А.3) програму BIOS та починається процедура POST (Power On Self Test, ініціалізація при першому вмиканні). Її завдання – просканувати та настроїти всю апаратну частину комп'ютера.

Рисунок А.3 – Мікросхема BIOS

Перш за все формується логічна архітектура комп'ютера. Подається живлення на всі чипсети, у їхніх регістрах установлюються потрібні значення. Потім визначається об'єм ОЗП (цей процес можна спостерігати на екрані), вмикається клавіатура, розпізнаються порти. На наступному етапі визначаються блочні пристрої – жорсткі диски IDE та SCSI, флопі-дисководи та DVD-приводи. Для пристроїв SCSI процедура дещо ускладнюється наявністю власної BIOS, яка бере на себе роботу з відповідним обладнанням і має власну програму настройки. На заключній стадії відбувається відображення підсумкової інформації.

Після закінчення роботи POST BIOS шукає завантажувальний запис. Цей запис, залежно від настройки, знаходиться на першому або другому жорсткому диску, флопі-диску, ZIP або DVD-приводі. Після того, як завантажувальний запис знайдено, він завантажується в пам'ять та управління передається йому.

Якщо в настройках SETUP BIOS є помилки, то вони можуть виявитися вже на цих стадіях, і до запуску ОС справа не дійде. Але можливі інші вияви неправильної настройки BIOS – повільна або нестабільна робота системи, раптові перезавантаження. Тому необхідно запустити програму настройки BIOS і зробити невелику екскурсію по її лабіринтах.

Практична частина

Перш за все, відразу після вмикання живлення, у нижній частині екрана знаходиться ідентифікаційний запис про версію BIOS, наприклад: **Press DEL to enter SETUP 04/19/2001-i815-W83627HF-6169RAB9C-00** .

Це означає, що своєчасно натиснувши клавішу Del, можна зайти у SETUP BIOS (рисунок А.4).

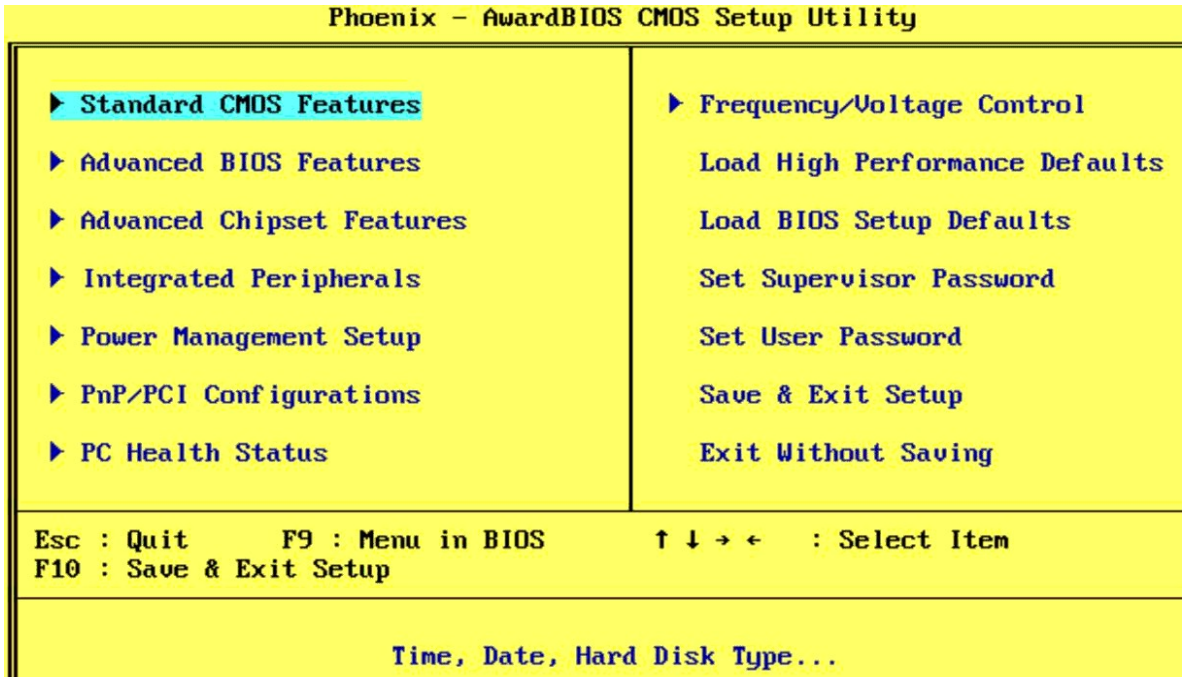


Рисунок А.4 – Вид інтерфейсу настройки BIOS

На жаль, єдиного стандарту інтерфейсу цієї програми не існує. Проте деяка логічна одноманітність – наслідок єдиної задачі, що виконується, – все ж таки є. На сьогоднішній день переважна більшість настільних ПК оснащена AWARD BIOS, тому при описі настройки потрібно спиратися в основному на цього виробника. Втім наведені відомості можна без проблем використовувати для настройки інших SETUP BIOS.

Програма настройки BIOS розділена на функціональні блоки, кожний з яких виконує свій клас задач. Звичайно це такі блоки (в дужках указано, як може називатися цей розділ):

1) загальні параметри (STANDARD CMOS SETUP, MAIN);

2) властивості самої BIOS (BIOS FEATURES SETUP, ADVANCED);

3) властивості інших чипсетів (CHIPSET FEATURES SETUP, Chip Configuration);

4) властивості інтегрованих пристроїв (INTEGRATED PERIPHERALS, I/O Devices Configuration);

5) властивості слотів PCI (PNP/PCI CONFIGURATION, PCI CONFIGURATION);

6) керування живленням (POWER MANAGMENT SETUP, POWER);

7) паролі системи (SUPERVISOR PASSWORD, USER PASSWORD);

8) збереження та відновлення настройок (SAVE SETUP, LOAD BIOS DEFAULT, LOAD SETUP DEFAULTS);

9) вихід та збереження (EXIT).

Потрібно мати на увазі, що цей розподіл досить умовний. У кожному комп'ютері можуть бути свої варіанти розподілу функцій по групах.

Для того, щоб активувати деяку групу, потрібно навести на неї за допомогою клавіш Up та Down курсорну рамку та натиснути Enter. Змінювати властивості можна клавішами Page Up та Page Down, а також «+» та «-». Опис керуючих клавіш звичайно наводиться у нижньому інформаційному рядку. Для виходу з блока використовується клавіша Esc.

Загальні властивості

У цьому розділі встановлюється системний час, настроюється IDE, SATA, DVD-приводи та флоппі-дисководи, вибирається реакція системи на помилки. Тут наводиться розмір інстальованої в комп'ютері RAM. Звичайно вказувати час і дату доводиться тільки при першому вмиканні комп'ютера або під час переходу на зимовий та літній час. Згодом правильне значення підтримується вбудованим годинником, що живиться від акумулятора. У сучасних ОС годинник, що йде неправильно, може дати погану послугу, надаючи помилкову інформацію про час доступу до файла та ін.

LBA MODE. При настройці дисків IDE, крім їхньої геометрії, вимагається вказати тип адресації секторів. Це пов'язано з тим, що файлова система FAT не здатна зайти далі 1023-го сектора, що обмежує обсяг вінчестера розміром 512 Мбайт. Щоб обійти це обмеження, була введена технологія

лінійної адресації блоків – LBA. Вона полягає в тому, що BIOS указує ОС «неправильні» параметри вінчестера, навмисно зменшуючи кількість секторів, так щоб воно не перевищувало 1024 та пропорційно збільшуючи число циліндрів. Сучасна файлова система (NTFS) позбавлена цього недоліку. Але якщо Windows 9x ОС після установлення на новий жорсткий диск не запускається, можливо, варто активувати LBA.

Як відомо, до стандартного комп'ютера можна підключити до чотирьох IDE-пристроїв. Існують два способи їхньої настройки: автоматичне виявлення при кожному вмиканні та жорстке задання параметрів. Наприклад, якщо в комп'ютері встановлений майстер-диск на першому IDE-порту, то знайшовши в SETUP відповідний пункт (Primary Master), можна вибрати для нього один з двох режимів виявлення – Auto або User Default. В останньому випадку передбачається, що користувач сам укаже геометрію вінчестера – кількість циліндрів, головок і секторів. Ця інформація наводиться на етикетці диска або може бути визначена підручними засобами самого SETUP BIOS – IDE HDD AUTO DETECTION. Якщо конфігурація ПК змінюється дуже рідко, то доцільно буде ввести параметри вручну. Тим самим процедура POST швидшатиме на одну-дві секунди. Встановивши для відсутніх IDE-пристроїв режим None, можна ще більше прискорити завантаження комп'ютера.

Настройка флопі-дисководів проводиться за допомогою параметрів Drive A та Drive B. Для них потрібно вибрати тип – звичайно 1,44 M або None, якщо дисковод відсутній. Інші формати флопі-накопичувачів використовуються дуже рідко. А от якщо не вказати None для відсутнього пристрою, то при початковому завантаженні будуть виникати помилки. Від Floppe 3 Mode Support – наслідки однієї з багатьох невдалих спроб «витягнути за вуха» 3-дюймові диски – рекомендується відмовитися.

Якщо жорсткий диск розбитий на декілька логічних дисків, то підключення додаткового вінчестера може викликати ускладнення. Диски D, E та інші «зсунуться», а місце звичного диска D займе новий диск. Окрім інших незручностей, порушаться зв'язки програм та ярликів. Щоб уникнути цього явища, потрібно встановити в SETUP BIOS для вінчестера, що

підключається, тип None. Тепер BIOS його не «побачить», а диск «з'явиться» тільки після завантаження Windows. ОС виявить вінчестер своїми методами та додасть його в кінець списку дисків.

Завантаження комп'ютера може супроводжуватися помилками. Те, як система повинна на них реагувати, визначає параметр Halt On:

1) All Errors – зупиняти завантаження при будь-якій помилці;

2) No Errors – продовжувати завантаження у кожньому разі;

3) All, But Keyboard – припиняти завантаження при будь-якій помилці, окрім відсутності клавіатури (цей режим часто використовується в серверних ПК, налаштованих на видалене управління);

4) All, But Diskette или All, But Disk/Key – переривати завантаження при будь-яких помилках, окрім відсутності дисковода або дисковода та клавіатури.

Властивості BIOS. У цьому розділі знаходяться різні опції, що так чи інакше відносяться до специфічних настройок BIOS, CPU, кеш і подібного. Вони відповідають за більш тонку настройку роботи комп'ютера, тому необхідно поставитися до них з належною увагою та акуратністю. Тут можна зустріти нижченаведені параметри (у дужках указані різні варіанти назв).

CPU Internal Frequency. Конструкція деяких материнських плат дозволяє вказати тут частоту процесора. Проте будьте обережні, «розгін» процесора може призвести до його пошкодження.

CPU Level 1 Cache, CPU Internal Cache. Вмикання-вимикання кеш першого рівня (внутрішнього). Вимикати цю опцію корисно тільки при пошуці несправності.

CPU Level 2 Cache, CPU External Cache. Вмикання-вимикання кеш другого рівня (зовнішнього). Вимикати цю опцію корисно тільки при пошуці несправності або перевірці її ефективності.

CPU L2 Cache ECC Checking. Спроба корекції помилок у кеш другого рівня. Хоча корисність цієї опції достатньо сумнівна, її активація ніяк не позначається на продуктивності системи.

Processor Number Feature. Починаючи з Pentium III, Intel надала можливість програмного визначення серійного номера процесора. Вимкнути цей режим має сенс, наприклад, з міркувань збереження конфіденційності.

Boot Up NUMLOCK Status. Автоматичне вмикання цифрової клавіатури корисно для індивідуальної настройки.

Typematic Rate Setting. Перехід у режим повторів постійно натиснутої клавіші. Частота повторів визначається параметрами Typematic Rate (Keyboard Auto Repeat Rate) та Typematic Rate Delay (Keyboard Auto Repeat Rate).

BIOS Update. Дозволяє або забороняє перепрошивання Flash BIOS. Після появи вірусу Chine, що руйнує системний BIOS, цю опцію варто вмикати тільки перед самим перезаписом Flash ROM.

Video BIOS Shadow. Сучасні ОС не користуються цією властивістю, віддаючи перевагу роботі з відеокартою напряму.

Gate A20 Option. Якщо присвоїти власнити цьому параметру значення Fast, то Windows буде швидше перемикатися в захищений режим і назад.

Наступні опції можуть бути виділені в окремий розділ BOOT.

Quick Power On Self Test. Прискорює завантаження, пропускаючи деякі тести, у тому числі потрійну перевірку ОЗП.

Virus Warning, Boot Virus Detection. Контролює доступ до завантажувального запису жорсткого диска; потрібно вимикати при інсталяції ОС.

Swap Floppy Drive. Змінює місцями дисководи A та B.

Boot Up Floppy Seek. Проводить пошук дисковода при завантаженні. Цей режим можна вимкнути, прискоривши тим самим виконання POST.

Boot Sequence. Послідовність перегляду дисків для завантаження ОС. Цей режим може бути представлений іншим способом – у вигляді списку з чотирьох пристроїв. Звичайно першим завантажувальним пристроєм зручно ставити диск C. Система, готова на завантаження з дискети, у недосвідчених користувачів викликає іноді труднощі. Якщо після закінчення роботи дискета залишається в дисководі, то при наступному вмиканні система буде безуспішно намагатися з неї

завантажитися. Окрім жорстких дисків і дисководів, сучасні системи можуть завантажуватися з DVD- та USB-дисків.

Властивості інших чипсетів

В основному це властивості чипів пам'яті та відео. Правильна їхня настройка необхідна для підвищення продуктивності системи або стійкості роботи.

В одну велику групу можна виділити параметри, що відносяться до циклів читання/запису в RAM. Процеси звернення до пам'яті прив'язані до тактів процесора та регулюються так званими синхронізуючими імпульсами. За допомогою SETUP BIOS користувач може змінювати величину затримки між імпульсами та довжину циклів.

Нижченаведені параметри цього типу мають таку особливість: чим менше їхнє значення, тим інтенсивніше працює пам'ять. При цьому підвищення продуктивності знижує стабільність роботи електроніки. Тому рекомендується оптимізувати систему поступово, вносячи дрібні зміни та повертаючись до робочих режимів при перших виявах помилок у роботі пам'яті. От ці параметри:

1) SDRAM CAS Latency Time (час затримки SDRAM CAS);

2) SDRAM Cycle Time Tras/TrcTras/Trc (час циклу пам’яті SDRAM);

3) SDRAM RAS-to-CAS Delay (затримка SDRAM RAS-to-CAS);

4) SDRAM RAS Precharge Time (час попереднього зарядження RAS SDRAM).

Якщо ПК оснащено шиною AGP, то серед настройок напевно можна зустріти такі параметри:

1) AGP Aperture Size MB. Технологія AGP дозволяє відеокарті «черпати» для своїх потреб пам'ять із системної RAM. Ці параметри визначають не розмір фізичної пам'яті, а адресний простір. У сучасних комп'ютерах рекомендується встановлювати розмір апертури в проміжку 64-128 Мб. Втім суворих правил відносно цього не існує. Збільшивши розмір пам'яті, що виділяється, продуктивність системи не погіршується;

2) AGP2X Mode. Використовувати режим AGP2X, що дозволяє значно збільшити пропускну спроможність шини,

можна тільки в тому випадку, якщо чип материнської плати та відеокарта підтримують AGP2x;

3) AGP4X Mode. Оскільки цей режим не сумісний з AGP2X та AGP1X, за умовчанням виробники материнської плати його вимикають. Але при необхідності можна його увімкнути.

Схожі настройки будуть у сучасних роз'ємів PCI-RXPRESS.

Властивості інтегрованих пристроїв

Як правило, в материнську плату вбудований ряд контролерів периферійних пристроїв: контролер IDE та SATA, контролер послідовних і паралельних портів, клавіатури, флопі-дисковода та ін. Іноді виникає необхідність вимикання деяких пристроїв – наприклад, для налагодження або звільнення переривань. У розділі INTEGRATED PERIPHERALS звичайно можна зустріти такі пункти:

1) Onboard IDE-1 Controller – перший контролер IDE-дисків; якщо використовуються SCSI пристрої, його можна відключити і тим самим звільнити 14-те переривання;

2) Onboard IDE-2 Controller – якщо в комп'ютері встановлений тільки один IDE-пристрій, а переривань катастрофічно не вистачає, то, відключивши IDE-2 Controller, можна звільнити INT 15;

3) Master/Slave Drive PIO Mode - цей параметр прив'язаний до конкретного IDE-пристрою та відповідає за режим передачі даних; звичайно краще дати можливість BIOS самій дібрати потрібне значення (режим Auto). Пропускна спроможність залежить від вибраного PIO таким чином: Master/Slave Drive ULTRADMA – цей параметр дозволяє\не дозволяє вмикати ULTRADMA, а також прив'язаний до конкретного контролера. Краще задовольнитися значенням, присвоєним йому за умовчанням; рекомендується також прослідити, щоб сама ОС також використовувала потрібний режим для пристрою;

4) USB Controller. Якщо в системі немає пристроїв USB, то для економії переривань можна присвоїти цьому параметру значення Disable;

5) USB Keyboard support**.** Як відомо, шина USB підтримується засобами ОС. Таким чином, до завантаження Windows клавіатура працювати не повинна. Режим USB Keyboard

support дозволяє BIOS самостійно, на етапі завантаження, обробляти події, що надходять від клавіатури;

6) USB Keyboard Support Via BIOS/OS. Протягом попереднього зауваження потрібно звернути увагу на те, що Windows може перемикатися в режим, де пристрої USB не працюють. Увімкнувши підтримку клавіатури через BIOS, користувач дістає можливість працювати з клавіатурою в додатках DOS. Але все-таки рекомендується, за винятком описаного випадку, використовувати підтримку клавіатури засобами ОС – як більш функціональну;

7) Init Display First (AGP, PCI). Якщо в ПК встановлено дві відеокарти, цей режим допомагає BIOS розібратися, яку з них використовувати на стадії завантаження комп'ютера;

8) Onboard FDD Controller – за допомогою цього параметра можна відключити інтегрований в материнську плату контролер флопі-дисковода. Така необхідність виникає у випадку, якщо для цього використовується окрема плата MIO або дисковода взагалі немає;

9) Report No FDD For Win95. Навіть якщо немає дисковода і відключено за допомогою попередньої опції контролер FDD, Windows все одно буде відображувати дисковод у списку пристроїв. Для усунення цієї проблеми потрібно встановите перемикач Report No FDD For Win95 в положення Enabled;

10) Onboard Serial Port 1/2. Цей параметр дозволяє відключити порти COM1 та COM2, а також добрати відповідні поєднання номерів порту введення-виведення та переривання. Якщо в комп'ютері не використовуються послідовні порти COM1 або COM2, то їх можна відключити, звільняючи відповідні переривання; інакше рекомендується використовувати режим Auto для автоматичної настройки портів;

11) Onboard Parallel Port – цей параметр має таке ж призначення, що й попередній, але відноситься до порту принтера;

12) Parallel Port Mode – дозволяє настроїти режим роботи паралельного порту. Якщо використовуються сучасні принтери або сканери, пимикані до порту принтера, треба вибрати режим ECP+EPP (або той, який рекомендований виробником периферії),

що реалізує двонаправлений обмін даними. Режим Normal призначений для старіших моделей принтерів.

Властивості слотів PCI

Про функції цього розділу звичайно згадують тоді, коли виникають конфлікти по перериваннях між пристроями ISA та PCI. Річ у тому, що однією із задач BIOS при завантаженні комп'ютера є правильний розподіл системних ресурсів. Згідно з цією технологією карта PCI може бути настроєна на роботу з певним перериванням і з певним портом введення-виведення. Більше того, одне й те саме переривання може спільно використовуватися декількома пристроями PCI.

Інформація про розподіл ресурсів зберігається в спеціальній таблиці – ESCD (Extended System Configuration Data). Але це ще не все. ОС, підтримуюча PNP, пізніше може перерозподілити ресурси на свій розсуд. Вважається, що Windows справляється з цією задачею ефективніше, ніж BIOS комп'ютера. Проте, ідилію псуватимуть карти ISA, що не підтримують PNP. вони настроюються за допомогою перемичок або спеціальних утиліт. Тому може виникнути необхідність закріпити за ISA-слотом певне переривання. Для цього служать такі параметри:

1) PNP OS Installed. Це складний параметр. Для Windows 95/98 рекомендується встановити значення Yes. Windows 2000/XP/7 використовує новітню технологію ACPI, тому для неї Microsoft рекомендує значення No. Linux не є повністю PnP-системою, але за наявності PnP-карт ISA значення Yes може знадобитися для ISAPNPTOOLS. Тут порада одна: поки все у порядку, не чіпати цей параметр. Якщо ж виникли проблеми, треба звіряти таблиці переривань – ту, яку виводить BIOS після процедури POST, та ту, яку використовує Windows. Якщо існують відмінності в непрацюючій платі – доведеться «длубатися» в настройках BIOS і Windows;

2) Reset Configuration Data, Force Update ESCD. Буває, що комп'ютер не розпізнає плату, встановлену замість старої. Присвоївши параметру Reset Configuration Data значення Enabled, можна примусити BIOS забути колишні установки та наново проаналізувати конфігурацію;

3) Resource Controlled By. Як учинити з розподілом ресурсів? Залишити це функціям BIOS (режим Auto) або ж

зробити вручну (Manual)? Якщо вибрати режим Manual, то активуються пункти, описані нижче;

4) IRQ-X assigned to. Цей параметр дозволяє перериванню X призначити тип пристрою. Режим Legacy ISA вимагає окремих IRQ та DMA. Режим PCI/Pnp ISA дозволяє використовувати ці ресурси спільно з іншою платою. Наприклад, для старої плати ISA, працюючої, скажімо, на перериванні 9 IRQ, можна, щоб уникнути конфліктів, вибрати режим Legacy ISA;

5) Delayed Transaction та PCI 2.1 Compliance. Обидва параметри відповідають за узгодженість роботи шин PCI та ISA. Якщо їх активувати, то дані між цими шинами будуть передаватися через буфер. Поки дані нагромаджуються в буфері, більш швидка PCI отримає можливість обробляти трансакції.

Управління живленням

Сучасні BIOS дозволяють оперувати чотирма станами енергоспоживання комп'ютера: робота на «повних обертах», режим, що понижує (Doze) частоти центрального процесора, режим очікування Standby (звичайно полягає у вимиканні відео та жорстких дисків), «сплячий» режим Suspend (максимально низьке енергоспоживання, вимикання пристроїв).

Система контролюється за допомогою лічильника простою певних пристроїв. Якщо ці пристрої не діють протягом певного часу, система переходить у той або інший стан зниженого енергоспоживання.

На початку розділу BIOS, керівника режимами живлення, користувачу пропонується вибрати схеми енергозбереження: дві стандартні (Min saving та Max Saving) або настроїти. Можливо, підійде одна з готових схем. Інакше необхідно вибрати режим User define та ввести вручну такі уточнюючі значення:

1) PM Control by APM. Advanced Power Management дозволяє управляти живленням пристроїв засобами ОС;

2) Video off Method. У режимі DPMS монітор вимикається сигналом від відеокарти. Якщо остання не підтримує протокол DPMS, то після чергового «засинання» комп'ютер уже не «прокинеться». У кожному разі, для сучасних моніторів краще вибирати режим V/H SYNC + Blanc;

3) Video off After. Тут потрібно вибрати стадію енергозбереження, на якій буде вимикатися монітор - Doze, Suspend або Standby;

4) Doze mode, Standby і Suspend. Уводяться тимчасові інтервали, після закінчення яких комп'ютер буде переходити в режими Doze, Standby та Suspend;

5) HDD Power Down – якщо до жорсткого диска давно не зверталися, його також можна відключити.

На закінчення слід зазначити, що оперувати описаними параметрами потрібно акуратно. Іноді неправильна настройка може призвести до «зависання» комп'ютера та втрати даних.

Паролі. Для перекриття стороннім доступу до настройок SETUP можна використовувати такі параметри SETUP:

1) Security. У режимі System вимагається ввести пароль для запуску комп'ютера, в режимі Setup – тільки для входу в SETUP BIOS;

2) User Password – тут можна встановити пароль для користувача, якому дозволені деякі дії щодо найпростішої настройки, а саме: запуск комп'ютера, установлення системного часу та деякі інші. Уведення «порожнього» пароля вимикає перевірку пароля;

3) Supervisor Password - цей пароль призначений для адміністрування SETUP, він дозволяє змінити пароль або скасувати призначений для користувача.

Вихід і збереження параметрів

Внесені зміни потрібно зберегти. Для цього існує команда Save and Exit. Якщо користувач не упевнений, що зробив усе правильно та хотів би скасувати зміни, можна використовувати Discard and Exit (Do not save and Exit).

Load Setup Default – відновлює заводські установки. Як правило, це найнадійніша та універсальна комбінація параметрів, якою можна скористатися, якщо по-іншому комп'ютер не запускається. Ще одна команда – Load Bios Default. По ній завантажується сам код BIOS, збережений у спеціальній області пам'яті. Така операція може знадобитися, якщо BIOS видає повідомлення про помилку контрольної суми, що свідчить про порушення структури інформації в Flash BIOS.

А.3 Робота на мові асемблер – створення програми

Теоретична підготовка

ЦП влаштований так, що йому необхідно давати команди, які він виконує. Кількість команд, які може виконувати ЦП, не велика, але змінюючи їхню послідовність можна створювати майже необмежені програмні продукти. Самі команди, які розуміє ЦП, називають машинними кодами. Програмувати людині в машинних кодах дуже складно. Тому для полегшення праці програмістів були придумані мови програмування. Першою мовою програмування була мова низького рівня – асемблер. Її особливість полягає у тому, що кожна команда цієї мови трансформується в один машинний код. Така властивість мови дуже добре дозволяє вивчити особливості комп'ютерної схемотехніки та архітектури ПК. Усі інші мови програмування для цього непридатні, оскільки не можуть відображати всі особливості комп'ютера.

Структурно мова асемблер складається з компілятора, який автоматично перекладає команди, зрозумілі програмісту (інструкції), в команди, зрозумілі комп'ютеру (машинні коди). Таким чином, асемблер є своєрідним посередником між програмістом і комп'ютером. Щоб асемблер міг правильно проводити компіляцію, необхідно ставити спеціальні інструкції, які повідомляють компілятор, що потрібно робити. Такі інструкції називають **директивами**. Після компіляції директиви не входять в основну програму.

У сучасних операційних системах (ОС) величезна кількість функцій (процедур) уже написана програмістами-розробниками операційних систем. Розумно використовувати ці процедури, а не писати їх самостійно. Це скорочує час роботи над програмами. Оскільки програми працюють під управлінням ОС, то для правильного запуску, забезпечення взаємодії із зовнішнім середовищем (виведення на екран, читання та запис з вінчестера та ін.) і правильного завершення необхідні спеціальні процедури ОС – API (Application Programming Interface – Інтерфейс прикладних програм). Кожній процедурі API передаються параметри, тобто відомості, необхідні їй для роботи.

Найпростіша процедура API – EXITPROCESS – процедура завершення програми.

Процедури ОС викликає спеціальна директива – **invoke**.

Приклад

Invoke ExitProcess. 0

Нижче приклад програми.

```
.386
.model  flat, stdcall
includelib \grup\stud\myasm\lib\kernel32.lib
ExitProcess proto :DWORD
.code
start:
mov eax, 2
add eax, 3
invoke ExitProcess, 0
end start
```

У даній програмі всього дві реальні команди, які будуть переведені у машинні коди:

1) mov eax, 2; – помістити 2 в регістр eax;

2) add eax, 3; – додати 3 до значення в регістрі eax.

Записи решти – директиви.

.386 – директива, що вказує мінімальний тип процесора, з яким повинна працювати програма. Тут слід зазначити, що всі процесори сумісні з більш ранніми версіями.

.model flat, stdcall – дана директива вказує, для якої ОС написана програма. У цьому випадку – Windows.

includelib \grup\stud\myasm\lib\kernel32.lib – директива пимикає бібліотеку kernel32.lib – бібліотеку, що містить готові машинні коди, які будуть вставлені в програму при компіляції. Дуже важливо правильно указати шлях до бібліотеки.

EXITPROCESS proto :DWORD – опис використовуваної в програмі процедури для компілятора. Даний рядок означає, що процедура має одну змінну розміром в DWORD (подвійне слово або чотири байти). Асемблер перекладе програму на мову машинних кодів, коли параметри, описані директивою proto, відповідають опису процедури у бібліотеці, а також параметрам,

указаним при виклику процедури директивою invoke.

.code – директива показує, що далі підуть саме команди програми, які потрібно буде перекладає в машинну мову.

start: – мітка (посилання) початку програми.

end start – директива закінчення програми.

Для роботи з мовою асемблер необхідно використовувати програму цієї мови – **MASM** фірми Microsoft. Дана програма є безкоштовною та може бути скачана з офіційного сайту фірми. Для набору програм, компіляції та відображення результатів їхньої роботи буде використовуватися програма **FAR.**

Практична частина

Інсталювати програму FAR.

Для написання програми необхідно створити умови роботи. Необхідно встановити програму для зручної роботи з асемблером. Для установлення програми FAR необхідно володіти правами адміністратора. Необхідно встановити програму, якщо вона ще не встановлена. Якщо програма вже встановлена на комп'ютері, то просто необхідно запустити програму. На рисунку А5 показане зображення вікна FAR.

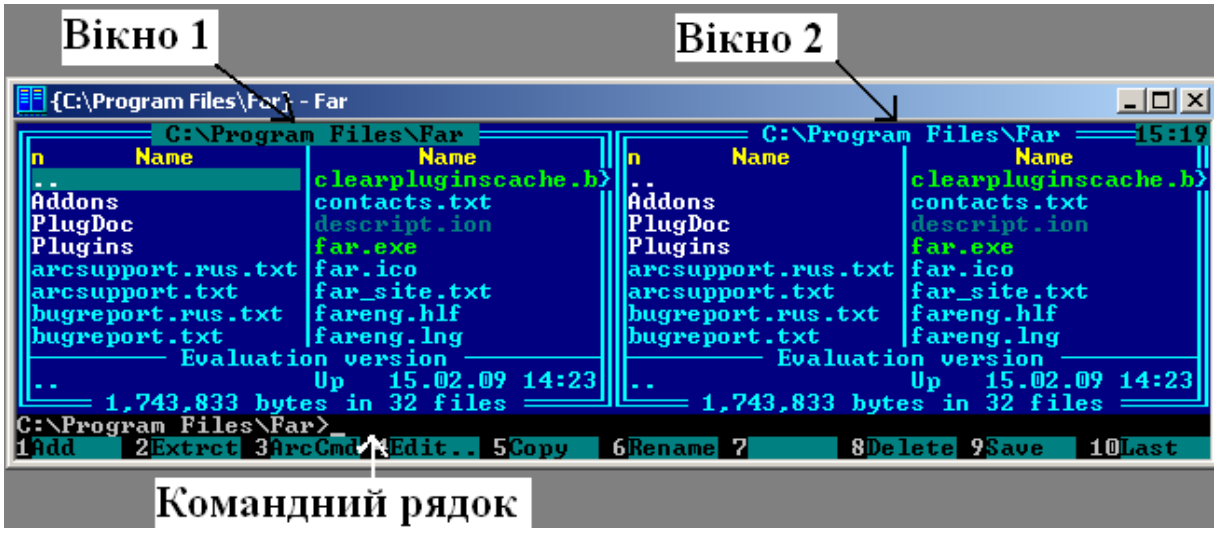


Рисунок А.5 – Зображення вікна програми FAR після запуску

Установлення компілятора асемблера MASM фірми Microsoft.

Для вибору потрібного диска необхідно натиснути комбінацію клавіш Alt+F1 (перше вікно) або Alt+F2 (друге вікно). У вікні (рисунок А.6), що з'явилося, необхідно вибрати диск, на якому буде проводитися робота.

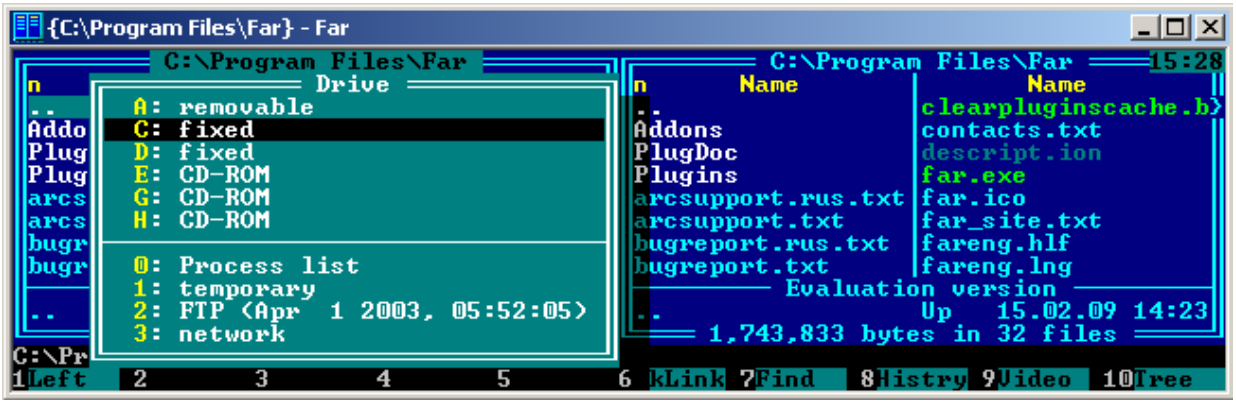


Рисунок А.6 – Зображення вікна програми FAR після натиснення комбінації клавіш Alt+F1

Після вибору диска необхідно створити папку для роботи. Ім'я папки необхідно створити у відповідності до номера групи, а всередині цієї папки створити папку з прізвищем. У наведеному вище прикладі програми stud і grup відповідно. Для створення папки необхідно натиснути клавішу F7 та у вікні, що з'явилося, набрати ім'я папки (рисунок А.7). Після цього натиснути Enter. На диску буде створена папка stud. Необхідно навести на неї курсор і натиснути Enter. Після цього папка відкриється, всередині неї аналогічно необхідно створити другу папку з прізвищем та увійти до цієї папки.

Тепер необхідно скопіювати в цю відкриту папку мову асемблер. Для цього потрібно скопіювати туди папку MYASM зі всім вмістом. Необхідно використовувати друге вікно (перехід по вікнах – клавіша Tab) та знайти цю папку. Потім навести на неї курсор і натиснути клавішу F5 і далі Enter. Після цього буде здійснено копіювання та у створеній папці виявиться папка MYASM.

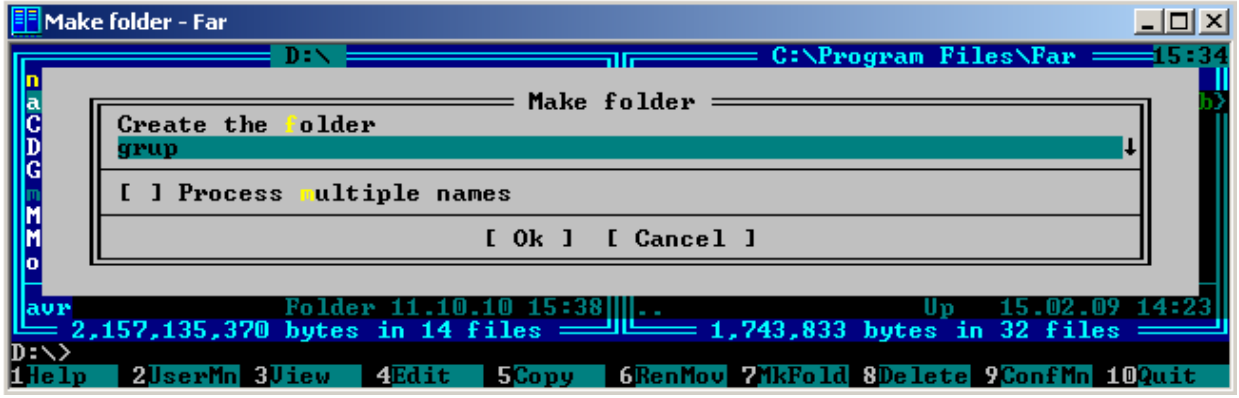


Рисунок А.7 – Зображення вікна програми FAR після натиснення клавіші F7 і введення імені папки

Набір програми

Для набору програми, запропонована в теоретичній частині, необхідно зайти глибше. Відкрити папку MYASM і далі папку BIN. У цій папці необхідно створити текстовий файл для набору програми. Для створення файла необхідно натиснути комбінацію клавіш Shift+F4. У вікні, що відкрилося, необхідно написати будь-яке ім'я програми, але обов'язково в кінці приписати чотири символи «**.asm**» без лапок (рисунок А.8). Після цього натиснути Enter і в результаті з'явиться вікно редактора для набору програми (рисунок А.9). Необхідно набрати програму та натиснути F2 для збереження тексту програми у файлі. Потім натиснути Esc і вийти з редактора.

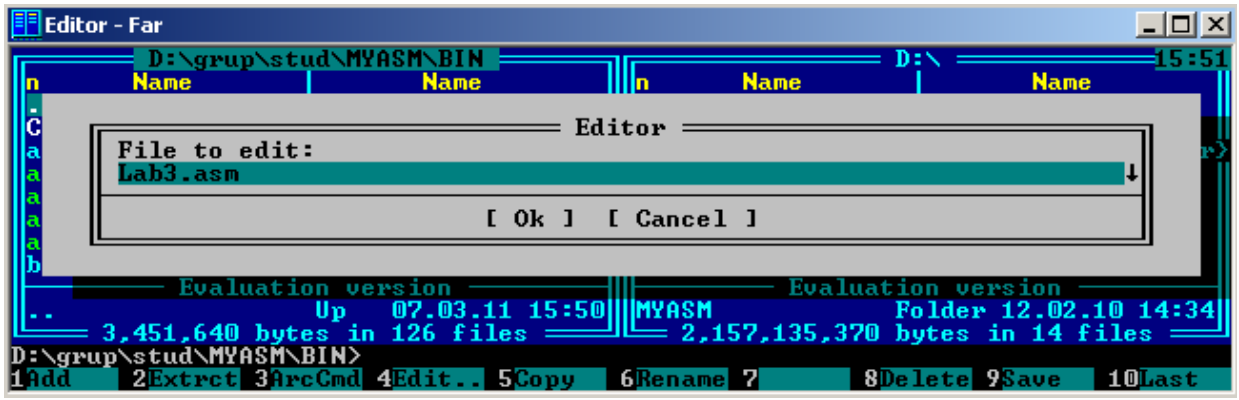


Рисунок А.8 – Приклад створення файла

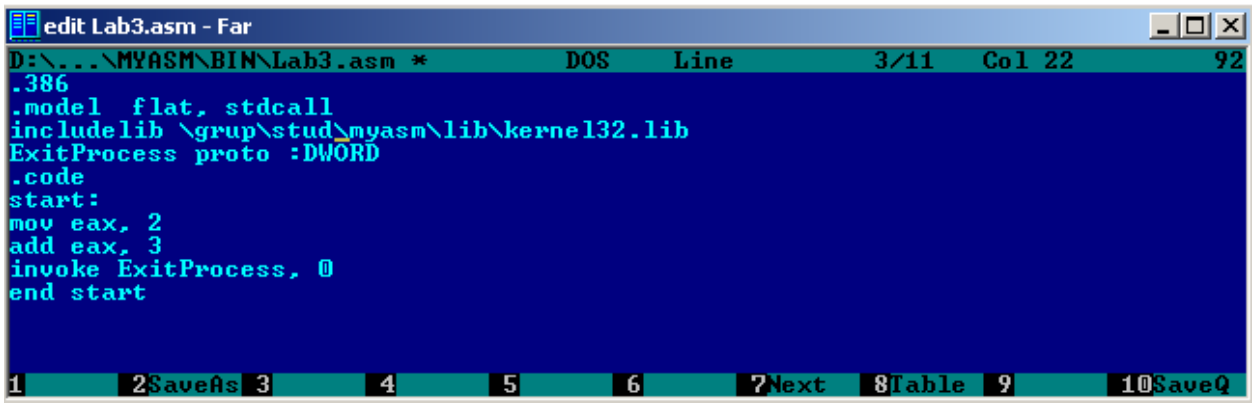


Рисунок А.9 – Набір програми у редакторі FAR

Компіляція програми

Набрати в командному рядку FAR запис «**amake lab3**» **без лапок** і натиснути ENTER. Після цього, якщо все правильно зроблено, в папці з'являться ще два файли: об'єктний (lab3.obj) та додаток або прикладна програма (lab3.exe). Обидва файли вийшли завдяки набору інструкцій, що знаходяться у файлі **amake.bat.**

Якщо цих файлів після компіляції не з'явилося, то необхідно натиснути Ctrl+o. Там можна буде подивитися на помилки допущені при написанні програми та які не дозволили компілятору створити файли додатка. Приклад подано на рисунку А.10.

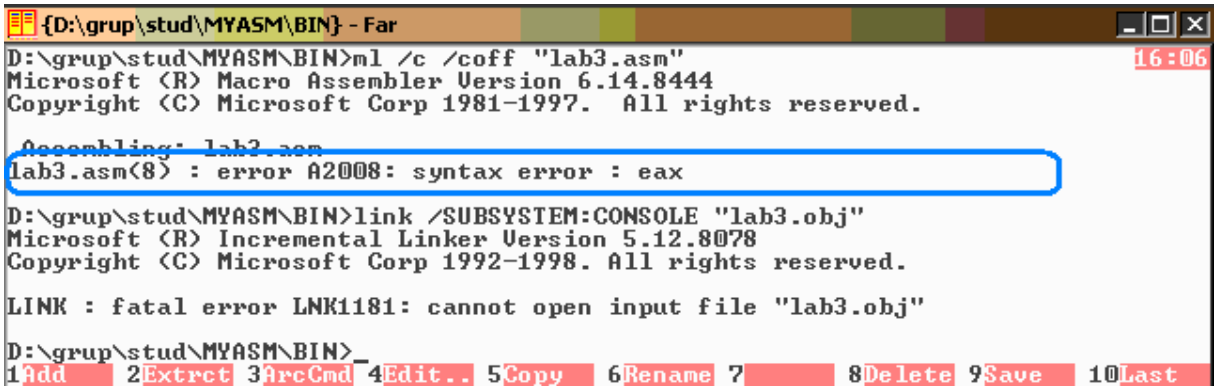


Рисунок А.10 – Приклад помилки, що виявлена компілятором

Необхідно натиснути Ctrl+o (поверне вікна). Увійти до програми (навести на Lab3.asm курсор і натиснути F4). Знайти помилку. Виправити програму та повторити дії з компіляції.

А.4 Робота на мові асемблер – основи структури ЦП

Теоретична підготовка

16-кова система числення

16-кова система числення – це коли після цифри **9** іде латинська буквена цифра **A**, а не **10**, а число **10** іде після буквеної цифри **F**: 0,1,2,3,4,5, 6,7,8,9,A,B, C,D,E,F,10, 11,12,…. Тоді 8+8=10, а 8+7=F.

16-ковая система є своєрідним посередником між комп'ютером і людиною. Комп'ютер може працювати тільки з «0» або «1», але числа, що переведені в такий формат, будуть дуже громіздкими. Зате будь-яке двійкове число легко перетворюється в 16-кове. 0=0000, 1=0001 …8=1000, 9=1001, A=1010, B=1011, C=1100, D=1101, E=1110, F=1111. Тоді 1A=00011010, а CB=11001011. Саме завдяки цій простоті й була введена 16-кова система.

Перенесення. Якщо необхідно провести які-небудь дії в комп'ютері, то необхідно скористатися регістрами, які містять команди та дані. Наприклад, для додавання двох чисел необхідно два регістри – в одному перше число, а в другому друге. Регістри мають різний розмір пам'яті (4,8,16,32,64-го розрядів (біта)). У регістр розміром 4 розряди можна помістити числа від 0000 до 1111 у двійковому коді, від 0 до 15 у десятковому коді, від 0 до F у 16-ковому коді. У регістр (32 – розряду) можна помістити число від 00…0 (32 нулі) до 11…1 (32 одиниці) у двійковому коді, від 0 до 4294967296 у десятковому коді, від 00 00 00 00 до FF FF FF FF в шістнадцятирічному коді.

А що відбудеться, якщо число стане більшим? Це питання повинен вирішувати програміст під час написання програми. А допомогу програміст одержує при використанні спеціального регістра прапорів, вбудованого в ЦП. Для сигналізації того, що число не вмістилося в регістр, установлюється **прапор перенесення «C»**. Наприклад, 1111+0001=0000 (відбулося переповнювання, адже для регістра у 4-му розряді одиницю, що переповнила, нікуди записати) і тоді прапор C=1.

Знак («+» і «-»). Для простоти прикладу потрібно використовувати 4-розрядний регістр (кількість розрядів не змінить суть). У 4-розрядному регістрі поміщаються 16 варіантів значень. Розділивши 16 пополам можна умовно отримати 8 додатних і 8 від’ємних чисел.

0 – 0000	
1 – 0001	-1 – 1111
2 – 0010	-2 – 1110
3 – 0011	-3 – 1101
4 – 0100	-4 – 1100
5 – 0101	-5 – 1011

6 – 0110 -6 – 1010
7 – 0111 -7 – 1001

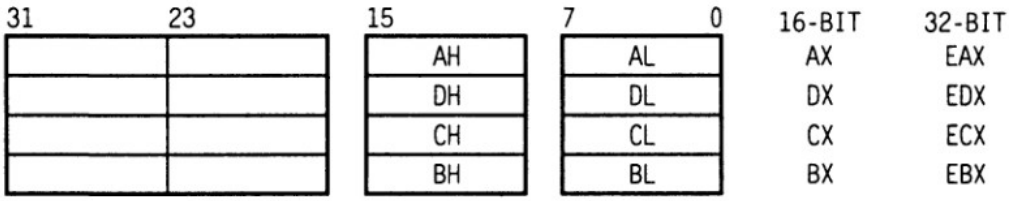
Щоб додатне число можна було відрізнити від від'ємного старший розряд установлюють в «0», якщо додатне число «1» – від'ємне. Але оскільки число «0» в існуючій обчислювальній системі не може бути від'ємним, то доведеться одне від'ємне число ігнорувати (1000)

Як видно з переліків від'ємні числа йдуть у зворотному порядку. Річ у тому, що в ЦП немає від'ємних чисел і віднімання, як є у звичайній алгебрі. У ЦП діє Булева алгебра, яка дозволяє отримати всі алгебраїчні розв'язання шляхом додавання та зсувів. От, наприклад, -1+1=0, а в двійковому коді 1111+0001=10000. Але оскільки п'ятого розряду в 4-розрядному регістрі немає, то 1 переходить у прапор переповнювання C, а результат стає 0000. Таким чином, у чотирьох останніх розрядах нуль. Сума додатного й такого ж по модулю від'ємного числа повинна бути рівна нулю. І так отримують усі від'ємні числа для регістрів будь-якого розміру. Залишилася тільки одна невикористана комбінація 1000. Їй немає пари, оскільки одне додатне число зайняте під нуль. Тому таким числом вважають -8, а такий код додатковим.

Чи є 1111 = 15 або 1111 = -1 знає тільки програміст. При програмуванні у прапорі регістрів використовуються два прапори, які визначають для асемблера, чи відбулося переповнювання або змінився знак: переповнювання (прапор O), знак – старший біт регістра (прапор S).

Байти і слова

Спочатку ЦП були 8-розрядними та робочі регістри були на 8 бітів. Потім у процесі розвитку ЦП стали 16-розрядними. Тоді з'явилися регістрові пари по 8+8 біт кожна. Коли ЦП став 32-розрядним, то він як і раніше повинен був уміти працювати з програмами, що написани для 8-розрядних процесорів (принцип сумісності). Нижче наведена структура основних робочих регістрів загального призначення.



Нижче наведено програму, що використовує різні комбінації роботи з регістрами.

```
.386
.model  flat,stdcall
includelib \grup\stud\myasm\lib\kernel32.lib
ExitProcess proto :DWORD
.code
start:
mov al,   10001000b    ;al = 88h =-120
mov bl, -127           ;bl = 81h
add al, bl             ;al = 09h, O = 1, C = 1, S = 0
mov ax, -120           ;al = 8816
mov bh, FFh            ;bx = ff8116 = -127
add ax, bx             ;ax = ff0916 = -247, O = 0, S = 1
invoke ExitProcess, 0
end start
```

Потрібно запам'ятати. У програмі двійковий код позначають буквою b, десятковий – d (за умовчанням використовується десятковий), шістнадцятковий – h.

```
mov ax,10b ; занести в ax число 2
mov ax,10h ; занести в ax число 16
mov ax,10d ; занести в ax число 10
```

Практична частина

Для створення файла програми потрібно звернутися у пункт А.3.

Для того, щоб можна було подивитися процес роботи програми та розібратися з особливостями двійкових, десяткових і шістнадцятирічних систем числення необхідно скористатися спеціальною програмою – зневаджувач. Програми такого типу дозволяють моделювати (емулювати) виконання програм усередині створеного ними середовища.

Після створення програми необхідно скористатися зневаджувачем OLLYDBG.exe, який знаходиться в папці BIN. Навести курсор на файл OLLYDBG.exe та натиснути Enter. На екрані з'явиться вікно. Необхідно вибрати FILE-OPEN і вибрати створений файл, як на рисунку А.11.

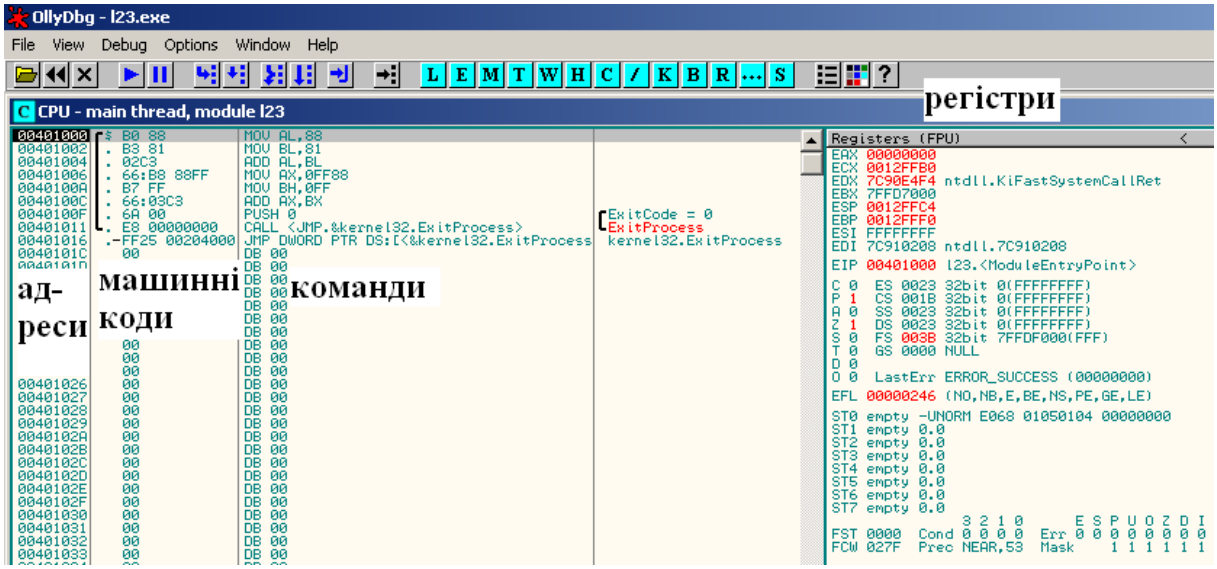


Рисунок А.11 – Приклад роботи зневаджувача OLLYDBG.exe

Дуже важливо попрацювати з даною програмою. Провести обчислення вручну та перевірити їх на комп'ютері. Для перевірки роботи програми необхідно натиснути клавішу F8. Тоді курсор, який указує на поточну позицію, перейде нижче, що буде означати виконання команди та перехід до наступної. А також змінять свої значення регістри, які були використані при виконанні поточної команди. Так, наприклад, після виконання першої команди результат наведений на рисунку А.12.

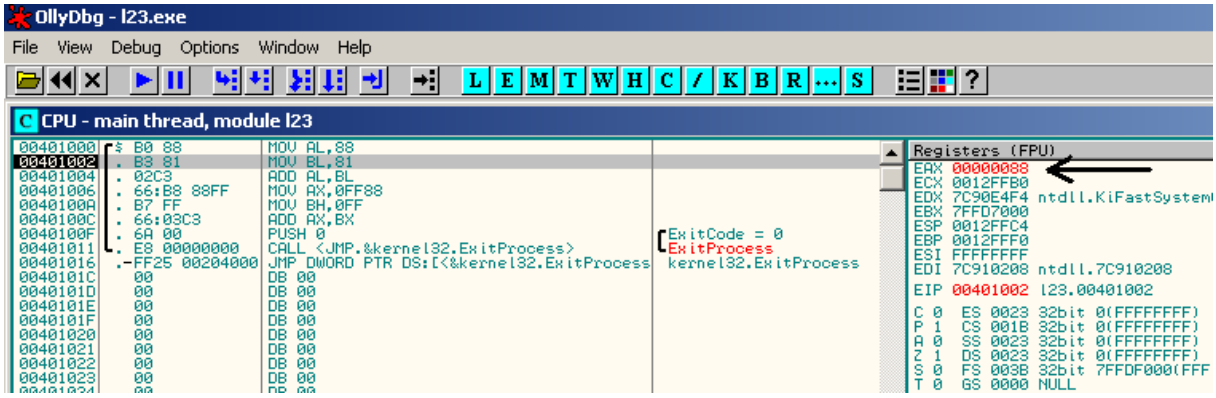


Рисунок А.12 – Приклад роботи зневаджувача OLLYDBG.exe

Продовжуючи змінювати числа в програмі необхідно повністю освоїти матеріал, запропонований у пункті А.4.

А.5 Дослідження пам'яті ПК

Теоретична підготовка

Програми, що виконуються комп'ютером, знаходяться в ОЗП. Процесор вибирає з ОЗП чергову команду, виконує її, потім переходить до наступної команди.

Команди можуть примушувати процесор змінювати вміст регістрів і записувати числа в ОЗП, що складається з окремо пронумерованих послідовно байтів. Найпершому байту присвоєний нульовий номер. Номер байта називають адресою. Адреси команд і даних, що зберігаються у пам'яті, завжди можна переглянути у зневаджувачі. Нижче наведено приклад програми, що ілюструє запис чисел у пам'ять.

```
.386
.model  flat,stdcall
includelib \grup\stud\myasm\lib\kernel32.lib
ExitProcess proto :DWORD
.data
        a1          WORD    -1
        a2          WORD    1
        a3          WORD    ?
        a4          WORD    00FFh
data_32   DWORD   -3
data_16   WORD    -3
data_8    BYTE    -3
.code
start:
mov al, data_8
sub ah, ah
dec ah
```

```
mov data_16, ax
invoke ExitProcess, 0
end start
```

У програмі використана директива **.data**, яка вказує процесору, що слідом за нею йдуть дані – числа, символи, а простіше – все те, що не можна вважати командами процесора. Асемблер буде вважати даними все, що розташовано в початковому тексті програми до директиви **.code**.

Директива **BYTE** задає дані розміром з байт, **WORD** – слово (два підряд байти), якщо необхідно задати дані розміром у 4 байти, то використовують директиву **DWORD**. Знак питання після **WORD** означає, що виділена пам'ять не має попереднього значення і там може зберігатися все, що завгодно від програм, що раніше використали пам'ять.

Рядок асемблера **data_8 BYTE -3** означає, що в змінну data 8 можна помістити будь-яке значення не більше ніж 0…255 або -127…+128 (в даному випадку -3).

Інструкція **mov al, data_8** бере з пам'яті програми байт, помічений раніше, як **data_8** і записує його в регістр **al.** Після цього число -3 залишається в **data_8** і в регістрі **al.**

Інструкція **sub ah,ah** надсилає різницю **ah-ah** у регістр **ah**, тобто нуль.

Інструкція **dec ah** зменшує значення **ah** на 1 (0-1=-1 або FFh). Цю інструкцію можна замінити **sub ah,1**, але така операція займає більше часу роботи процесора та не використовується. Таким чином, у регістрі **ah** тепер містяться всі 1 (пункти А.3, А.4).

Використавши інструкцію **mov data_16, ax** ЦП поміщає обидва регістри **ah, al** у **ax** (пункт А.4) в область пам'яті, виділену в програмі під ім'ям **data_16.**

Практична частина

Використовуючи досвід, отриманий при роботі з пунктами А3,А4, необхідно написати, скомпілювати та запустити отриману програму у зневаджувачі.

В лівому верхньому кутку видно адреси пам'яті команд ЦП. Нижче – область даних (сегмент даних), яка хоча й поміщена раніше області команд у програмі, має більші за величиною

адреси та зберігає числа у міру їхнього надходження з області написаної програми тільки в шістнадцятковому форматі. Оскільки змінна **a1** була заявлена як **WORD** (два байти) з числом -1, то в перших адресах сегмента даних вона записана як FF FF (від'ємне число з 16 біт). Змінні решти також поміщені підряд.

Важливим моментом є той, що в пам'ять записалося число -3 (у шістнадцятковому форматі FF FD), але записано його як FD FF. Це тому, що в процесорах Intel числа розташовуються в пам'яті за правилом: **молодший байт має меншу адресу**. Річ у тому, що латиниця та кирилиця передбачають порядок написання тексту зліва направо, а арабський текст пишеться справа наліво. Тому, використовуючи кириличний правопис та арабські цифри, європейці звикли до цього та не помічають. Приклад. Число 123 вимовляється «сто двадцять три», від старшого до молодшого, а арабочитаючі люди вимовляють його «три двадцять сто». Відповідно, якщо читати правильно, то при письмі зліва направо потрібно записувати 321 (три двадцять сто). Комп'ютер записує в правильному порядку. Тому спочатку записується молодший байт FD, а потім старший FF.

Якщо уважно розглянути ще раз рядок зневаджувача нижче **00401000 A0 0030400E MOV AL,BYTE PTR DS:[40300E],** то A0 – машинний код команди **MOV**, а **0030400E** – 64-бітна адреса змінної **data_8** знаходиться в сегменті даних і вивернута на виворіт, за наведеним правилом (рисунок А.13).

Команда **MOV** може пересилати дані тільки через регістр (так улаштований ЦП). Не можна проводити пересилання зі змінної до змінної. Тому команда **mov data_16, data_8** буде визнана як помилка, але припускає пересилання числа у змінну, тому наступний формат команди **mov data_8,-3** – буде прийнятий.

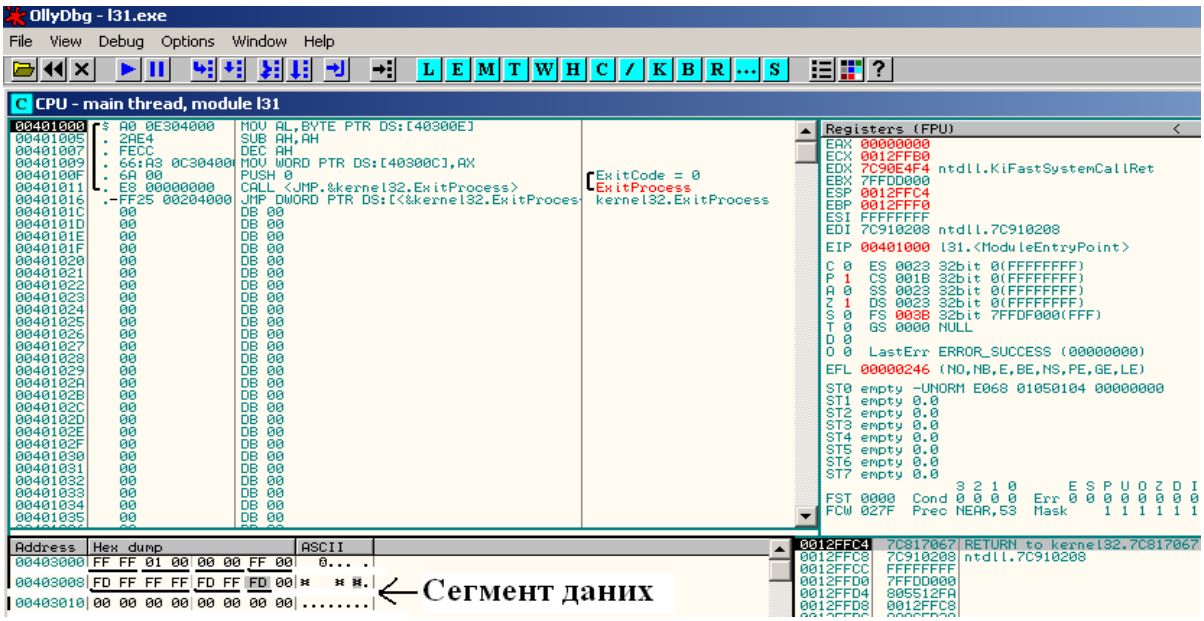


Рисунок А.13 – Сегмент даних ПК у зневаджувачі

Усі змінні, що указані у програмі, не відображувалися у зневаджувачі. Це відбувається тому, що всі змінні при компіляції були замінені на діючі адреси.

Необхідно уважно поставитися до розташування даних у ПК. Для цього потрібно багато разів проробити зміни чисел у змінних і подивитися на розташування їх у зневаджувачі. Провести обчислення вручну та перевірити їх на комп'ютері.

А.6 Дослідження стека ПК

Теоретична підготовка

Стек в інформатиці та програмуванні — різновид лінійного списку, структура даних, яка працює за принципом (дисципліною) «останнім прийшов — першим пішов» (LIFO, англ. last in, first out). Усі операції (наприклад, видалення елемента) в стека можна проводити тільки з одним елементом, який знаходиться на вершині стека та був введений у стек останнім.

Стек можна розглядати як певну аналогію до стосу тарілок, з якої можна взяти верхню, і на яку можна покласти верхню тарілку (інша назва стека — «магазин», за аналогією з принципом роботи магазина в автоматичній зброї).

Важливою властивістю стека є відсутність необхідності проводити повну адресацію кожного байта. Достатньо знати адресу «вершини» стека, а потім додаючи або прибираючи дані зі стека просто зміщувати адресу вершини стека на відповідну кількість байтів. Адреса вершини стека знаходиться у регістрі ESP.

При збереженні даних у пам'яті комп'ютера не обов'язково вказувати адресу. Спеціальна команда процесора **PUSH** поміщає слово або подвійне слово в область пам'яті, що називається стеком, а команда **POP** читає дані зі стека та записує їх у регістр або звичайну область пам'яті. Важливо пам'ятати, що повернення даних зі стека можливе тільки в строго зворотній послідовності поміщених у нього значень (останнім прийшов – першим вийшов). Нижченаведена програма ілюструє сказане.

У програмі використано нову директиву – «;». Після крапки з комою у програмі йдуть коментарі, тому компілятор їх не розглядає. Не потрібно при наборі програми переписувати їх. Тут це зроблено для розшифрування дій команд та розуміння їх користувача.

```
.386
.model  flat,stdcall
includelib \myasm\lib\kernel32.lib
ExitProcess proto :DWORD
.code
start:
mov eax, 2        ;             eax = 2
mov ecx, 3        ;             ecx = 3
push eax          ; відправити значення регістра eax до стека
push ecx          ; відправити значення регістра ecx до стека
mov eax, 4        ;             eax = 4
mov ecx, 5        ;             ecx = 5
pop  ecx          ;             eax = 3
pop  eax          ;             ecx = 2
invoke ExitProcess, 0
end start
```

У даній програмі регістрам спочатку були присвоєні одні

значення. Потім значення, що містяться в цих регістрах, були поміщені в стек. Далі у програмі були виконані дії над цими регістрами, тому їхній вміст змінився. Завдяки команді POP первинні значення були знов поміщені в регістри.

Важливо пам'ятати два правила:

1) поміщене в стек останнім – виходить першим;

2) розмір змінних, що поміщені в стек та що витягнуті звідти, повинен збігатися.

Практична частина

Використовуючи досвід, отриманий у попередніх роботах, необхідно набрати та компілювати програму, наведену в теоретичній частині.

Використовуючи зневаджувач, можна спостерігати зміни, що відбуваються зі стеком, у правому нижньому вікні (рисунок А.14), а вміст регістрів можна спостерігати в правому верхньому вікні. Для покрокового виконання програми необхідно скористатися клавішею F7 або F8.

Важливо пам'ятати, що команда POP збільшує адресу вершини стека, але не стирає самі числа. Після команди POP числа як і раніше лежать у стека та будуть там знаходитися до тих пір, поки чергова команда PUSH їх не зітре. Щоб їх побачити, необхідно натиснути стрілки вікна стека, після чого вершина зміститься, відкривши значення. Якщо є необхідність, то можна вилучати ці значення, але вже не за допомогою команди POP, а за допомогою непрямої адресації (пункт А.7).

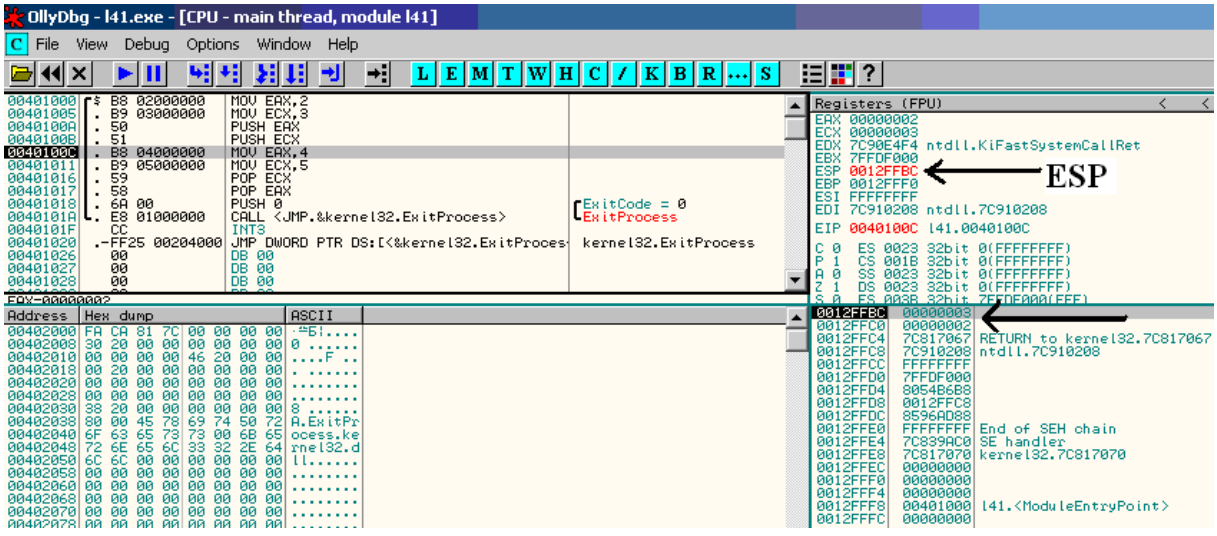


Рисунок А.14 – Зміна стека під дією команди PUSH

Самостійно змініть програму та перевірте правильність обох правил користування стеком.

Для цього необхідно зробити дві програми:

а) щоб перевірити правило перше, необхідно в програмі поміняти місцями регістри EAX та ECX в командах POP;

б) щоб перевірити правило друге, необхідно в програмі (запропоновано у керівництві, а не модифіковано в пункті «а») поставити замість регістра ECX (32-розрядний регістр) його усічену версію CX (16-розрядний регістр).

Обидва отримані варіанти переглянути у зневаджувачі.

Необхідно уважно поставитися до розташування даних у стека. Для цього потрібно багато разів проробити зміни чисел у регістрах і подивитися на розташування їх у зневаджувачі. Провести обчислення уручну та перевірити їх на комп'ютері.

А.7 Дослідження пам'яті ПК – непряма адресація

Теоретична підготовка

В додатку (пункт А.6) був розглянутий матеріал з прямого переміщення даних у стек (команда PUSH) і прямого вилучення даних зі стека (команда POP).

Команда PUSH записує дані в стек і зміщує вершину на кількість байтів, які були записані. При цьому стара інформація затирається. Команда POP вилучає дані з стека та зміщує вершину на кількість байтів, які були вилучені. Відмінність у тому, що після вилучення дані в стека не затираються, а залишаються над вершиною стека. При необхідності ці дані можуть бути вилучені повторно, поки їх не затре команда PUSH. Також очевидно, що команда POP для повторного вилучення непридатна, оскільки вилучає тільки два або чотири байти знизу вершини стека.

Для вилучення даних, які не можуть бути вилучені командою POP, існує непряма адресація (насправді непряма адресація використовується для різних операцій з пам'яттю, а не тільки для операцій із стеком).

Оскільки єдиним орієнтиром для асемблера є вершина стека (адреса вершини стека завжди знаходиться в регістрі ESP), то від вершини стека необхідно відштовхуватися.

Раніше використовувалися команди присвоєння mov, додавання add та ін. Для непрямої адресації не існує спеціальних команд. Використовуються ті ж команди з однією особливістю. Якщо регістр поміщений у квадратні дужки (mov ecx,[esp]), то асемблер вважає, що в регістрі знаходиться не значення, а адреса, за якою знаходиться значення. Приклад програми непрямої адресації наведено нижче.

```
.386
.model  flat,stdcall
option casemap:none
includelib \myasm\lib\kernel32.lib
ExitProcess proto :DWORD
.code
start:
mov eax,12345678h
push eax
mov eax,9abcdef0h
push eax
pop eax
mov ebx,esp      ; у ebx поміщено значення, що знаходиться у
; регістрі esp – адреса вершини стека
mov ecx,[esp]    ; у ecx поміщено значення, що знаходиться
; за адресою вершини стека, яка знаходиться
; у регістрі  esp
invoke ExitProcess, 0
end start
```

Результати роботи програми наведено на рисунку А.15.

Як видно з рисунка А.15, програма змогла вилучати чотири байти нижче за вершину стека та помістити їх у регістр ECX. Але цю ж операцію могла зробити команда POP ECX. Але різниця є (mov ecx,[esp] – не зміщує вершину стека) задача ставилася по вилученню даних над вершиною стека (хоча б 9ABCDEF0). Це

достатньо просто. Приклад таких вилучань із стека наведено в програмі нижче.

```
.386
.model  flat,stdcall
includelib \myasm\lib\kernel32.lib
ExitProcess proto :DWORD
.code
start:
mov ax,     2211h
mov ecx,66554433h
push ax
push ecx
pop  ax
pop  ecx
mov  ecx,[esp-6]                    ; ecx=66554433h
mov  ax, [esp-2]                    ; ax=2211h
invoke ExitProcess, 0
end start
```

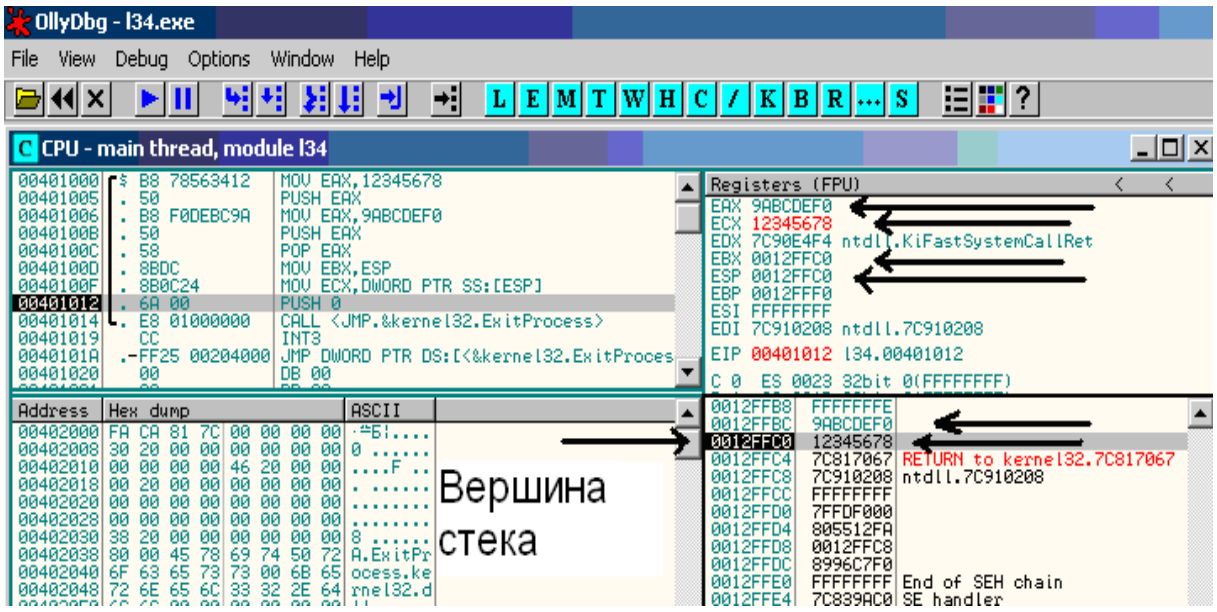


Рисунок А.15 – Результати роботи непрямої адресації

Практична частина

Використовуючи досвід, отриманий у попередніх роботах, необхідно набрати та компілювати програми, наведені у

теоретичній частині.

Використовуючи зневаджувач, можна спостерігати зміни, що відбуваються зі стеком, у правому нижньому вікні (рисунок А.11), а вміст регістрів можна спостерігати в правому верхньому вікні. Для покрокового виконання програми необхідно скористатися клавішею F7 або F8.

Важливо пам'ятати, що команда POP збільшує адресу вершини стека, але не стирає самі числа. Після команди POP числа як і раніше лежать у стека та будуть там знаходитися до тих пір, поки чергова команда PUSH їх не зітре. Щоб їх побачити, необхідно натиснути стрілки вікна стека, після чого вершина зміститься, відкривши значення. Якщо є необхідність, то можна вилучити ці значення, але вже не за допомогою команди POP, а за допомогою непрямої адресації, що розглянуто у додатку А.

Самостійно змініть програми та обидва отримані варіанти необхідно досконало переглянути у зневаджувачі.

Необхідно уважно поставитися до розташування даних у стека. Для цього потрібно багато разів пропрацювати зміни чисел у регістрах і подивитися на розташування їх у зневаджувачу. Провести обчислення вручну та перевірити їх на комп'ютері.

Важливо! Непряма адресація не змінює адресу вершини стека, що знаходиться у регістрі esp.