МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

„ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ”

Лабораторна робота №4

з курсу: «Архітектура обчислювальних систем»

Дослідження адресації даних в «реальному режимі»

Студент ІІ курсу

Група ДА-51

Болобан Олег

Київ, 2017 рік

Частина 1. Адресація в «реальному» режимі.

1. Скласти схему обчислення адресу процесорів архітектури х86.

2. Пояснити призначення регістрів ЦПУ, використовуваних для адресації.

Класичний режим адресації, який використовувався в перших моделях. Використовує сегментну модель пам'яті, яка організована наступним чином: адресний простір в 1 МіВ розбивається на 16-байтові блоки, які називаються параграфи. Всього параграфів в 1 МіВ — 65536, що дозволяє пронумерувати їх 16-розрядними числами. Сегменти пам'яті мають розмір 65536 байт, і завжди починаються на межі параграфу. Адреса комірки пам'яті складається з двох частин: номер параграфу, з якого починається сегмент та зміщення всередині сегменту і зазвичай записується як SSSS:OOOO, де S та O — шістнадцяткові цифри. SSSS називається сегментною компонентою адреси, а OOOO — зміщення. Адреса комірки, яка видається на шину, являє собою сегментну компоненту помножену на 16 плюс зміщення. Сегментна компонента вміщується в спеціальний регістр, який називається сегментний, а зміщення в регістр ІР(регістр інструкцій). Мікропроцесори 8086/8088, 80186/80188 та 80286 мали чотири сегментні регістри, тобто могли працювати одночасно з чотирма сегментами пам'яті.

Сегментні регістри та їх призначення:

CS — Сегмент коду. Використовується для вибірки команд програми;

DS — Сегмент даних. Використовується за замовчуванням для доступу до даних;

ES — Додатковий сегмент. Є отримувачем даних в командах обробки рядків;

SS — Сегмент стеку. Використовується для розміщення програмного стеку;

FS — Додатковий сегментний регістр. Спеціального призначення не має. З'явився в процесорі 80386;

GS — Аналогічно попередньому, але в нових процесорах с 64-бітною архітектурою має особливий статус: може використовуватись для швидкого переключення контекстів.

Незважаючи на те, що сегментні регістри мають спеціальні призначення, архітектура припускає при звернені до даних змінити один сегмент на інший. Сегменти коду, стеку та отримувача рядків завжди використовують регістри CS, SS та ES та не можуть бути змінені. Загальний об'єм пам'яті, яка адресується в реальному режимі становить 1048576 байт(ОООО:ОООО-FOOO:FFFF(OOOOO-FFFFF)-логічна(фізична) адреса в шістнадцятковій системі числення). Сегментний підхід дозволяє розділити всю пам'ять на 16 сегментів, які починаються з адрес, кратних 64 Кбайт. Ці 16 сегментів називають сторінками пам'яті. Зазвичай ділення на сторінки використовується для спільного функціонування пристроїв, інтерфейси яких відображені на адресний простір пам'яті; тоді кожний такий пристрій використовує одну сторінку пам'яті, та адреса комірки в адресному просторі пристрою буде збігатися зі зміщенням у сегменті пам'яті комп'ютера. Також в реальному режимі відсутній захист пам'яті та розмежування прав доступу, тому він вже практично вийшов з використання. Являється режимом за замовчуванням для всіх моделей процесорів x86.

Частина 2. Дослідження способів адресації даних:

1. Визначити склад і пояснити призначення засобів обчислення адресу даних.

2. Перерахувати способи адресації даних і пояснити їх особливості.

3. Скласти схеми обчислення адресу для різних способів адресації даних.

### **Автоматичне виведення операнду(Подразумеваемый операнд):**

У команді може не міститися явних вказівок про операнд; в цьому випадку операнд мається на увазі і фактично задається кодом операції команди.

### **Автоматичне виведення адресу(Подразумеваемый адрес):**

У команді може не міститися явних вказівок на адресу, що бере участь в операції операнда або адреси, за якою повинен бути розміщений результат операції, але ця адреса мається на увазі.

### **Безпосередня адресація:**

У команді міститься не адреса операнду, а безпосередньо сам операнд. При безпосередній адресації не потрібно звернення до пам'яті для вибірки операнда і комірки пам'яті для його зберігання. Це сприяє зменшенню часу виконання програми і займаного нею об'єму пам'яті. Безпосередня адресація зручна для зберігання різного роду констант.

### **Пряма адресація:**

Адреса вказується безпосередньо у вигляді деякого значення, всі комірки пам`яті розташовуються на одній сторінці. Перевага цього способу в тому, що він самий простий, а недолік - в тому, що розрядність регістрів загального призначення процесора повинна бути не менше розрядності шини адреси процесора.

### **Відносна (базова) адресація:**

При цьому способі адресації виконавча адреса визначається як сума адресного коду команди і базової адреси, як правило зберігається в спеціальному регістрі - регістрі бази.

Відносна адресація дозволяє при меншій довжині адресного коду команди забезпечити доступ до будь комірці пам'яті. Для цього число розрядів в базовому регістрі вибирають таким, щоб можна було адресувати будь-яку комірку оперативної пам'яті, а адресний код команди використовують для представлення лише порівняно короткого «зміщення». Зсув визначає положення операнда щодо початку масиву, що задається базовою адресою.

### **Скорочена адресація:**

В адресному полі командного слова містяться лише молодші розряди комірки, що адресується. Додатковий вказівний регістр.

Адресація з регістром сторінки є прикладом скороченою адресації. При цьому вся пам'ять розбивається на блоки-сторінки. Розмір сторінки диктується довжиною адресного поля.

### **Регістрова адресація:**

Регістрова адресація є окремим випадком скороченої. Застосовується, коли проміжні результати зберігаються в одному з робочих регістрів центрального процесора. Оскільки регістрів значно менше ніж комірок пам'яті, то невеликого адресного поля може вистачити для адресації.

### **Непряма адресація:**

Адресний код команди в цьому випадку вказує адресу комірки пам'яті, в якій знаходиться адреса операнда чи команди. Непряма адресація широко використовується в малих і мікроЕОМ, що мають короткий машинне слово, для подолання обмежень короткого формату команди (спільно використовуються регістрова і непряма адресація).

### **Адресація слів змінної довжини:**

Ефективність обчислювальних систем, призначених для обробки даних, підвищується, якщо є можливість виконувати операції зі словами змінної довжини. У цьому випадку в машині може бути передбачена адресація слів змінної довжини, яка зазвичай реалізується шляхом зазначення в команді місцеположення в пам'яті початку слова і його довжини.

### **Стекова адресація:**

Стекова пам'ять, що реалізує безадресне задання операндів, особливо широко використовується в мікропроцесорах і Міні-ЕОМ.

### **Автоінкрементна і автодекрементна адресація:**

Оскільки регістрова непряма адресація вимагає попереднього завантаження регістра непрямою адресою з оперативної пам'яті, що пов'язано з втратою часу, такий тип адресації особливо ефективний при обробці масиву даних, якщо є механізм автоматичного збільшення чи зменшення вмісту регістра при кожному зверненні до нього. Такий механізм називається відповідно Автоінкрементною і автодекрементною адресацією. У цьому випадку достатньо один раз завантажити в регістр адресу першого оброблюваного елементу масиву, а потім при кожному зверненні до регістра в ньому буде формуватися адреса наступного елемента масиву.

### **Індексація:**

Для реалізовуваних на ЕОМ методів розв'язання математичних задач і обробки даних характерна циклічність обчислювальних процесів, коли одні й ті ж процедури виконуються над різними операндами, впорядковано розташованими в пам'яті. Оскільки операнди, що оброблюються при повтореннях циклу, мають різні адреси, без використання індексації потрібно було б для кожного повторення складати свою послідовність команд, що відрізняються адресними частинами.

Програмування циклів істотно спрощується, якщо після кожного виконання циклу забезпечено автоматичну зміну у відповідних командах їх адресних частин згідно розташуванню в пам'яті оброблюваних операндів. Такий процес називається модифікацією команд, і заснований на можливості виконання над кодами команд арифметичних і логічних операцій.

Частина 3. Написати і відлагодити програму, що виконує дії, вказані в таблиці. Для адресації елементів масиву використовувати непрямі режими адресації. Програма має містити процедуру, яка обробляє вихідні параметри зі стеку. Навести приклади використання різних способів адресації даних і команд в програмі, зафіксувавши вміст відповідних регістрів і адрес пам'яті.

Завдання для 1 варіанту. Даний масив байтів. Знайти різницю максимального й мінімального елементів масиву.

**Лістинг програми:**

**title lab4**

**.model small**

**.stack 100h**

**.data**

**massive db 4,10,14**

**middle\_arithmetical dw ?**

**max dw ?**

**summ dw ?**

**.code**

**start:**

**mov ax,@data**

**mov ds,ax**

**lea bx,massive**

**mov cx,3**

**mov ax,[bx]**

**begin:add [bx],ax**

**mov ax,[bx]**

**next: inc bx**

**inc bx**

**loop begin**

**loop next**

**mov max,ax**

**cbw**

**idiv cx**

**mov dx,middle\_arithmetical**

**mov dx,max**

**mov ds,ax**

**lea bx,massive**

**mov cx,3**

**mov ax,[bx]**

**dispertion:add [bx],ax**

**mov ax,[bx]**

**sub ax,middle\_arithmetical**

**mul ax**

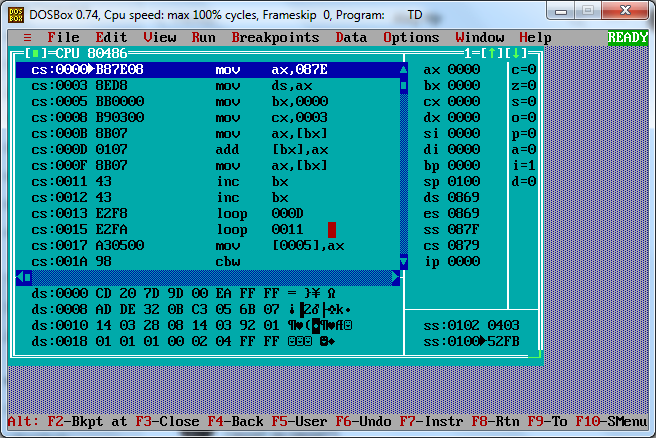
**add summ,ax**

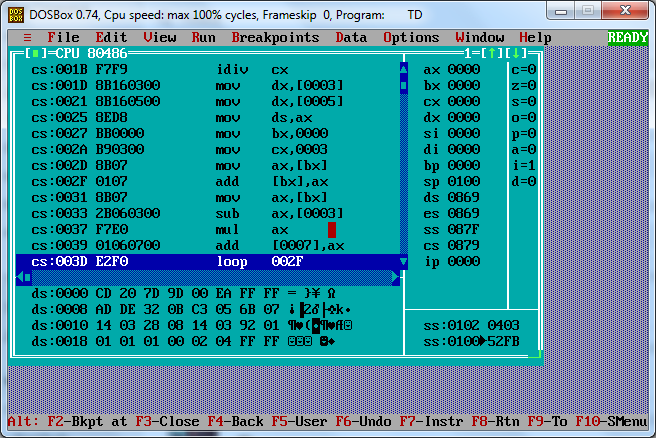
**loop dispertion**

**loop next**

**mov ax,4C00h**

**int 21h**

**end start** 



Висновок: У даній лабораторній роботі я дослідив адресацію пам`яті в реальному режимі, способи адресації даних та виконав програму, яка ілюструю непрямі способи адресації пам`яті.