**Лабораторна робота №2**

**Методи адресації**

**Ціль роботи:** Вивчення методів адресації МП І8086 та організації доступу до пам'яті. В процесі виконання лабораторної роботи планується вивчити організацію пам'яті комп'ютера, методи формування фізичної адреси та одержання доступу до будь-якої точки пам'яті.

**Короткі теоретичні відомості**

Фізичний об'єм пам'яті процесора i8086 містить 1 Мбайт. Але 16-тирозрядна архітектура процесора дозволяє звертатися тільки до 64 Кбайт пам'яті. Для забезпечення доступу до даних об'єм в 1 Мбайт використовується метод сегментації пам'яті.

Сегментація пам'яті – це виділення окремих областей пам'яті з єдиного адресного простору й надання їх у користування. Сегмент може розміщатися в любій точці пам'яті об'ємом в 1 Мбайт, а його адреса повинна містити 20 розрядів та змінюватися від 00000h до FFFF0h. Об'єм сегмента визначається розрядністю регістрів і становить 64 Кбайта (216 = 65565 байт). Сегменти можуть мати наступне розташування відносно один одного: накладатися, стикатися, мати загальні області, бути вкладеними або ізольованими. Фізична адреса елементів у пам'яті має дві складові: адреса сегменту та адреса зміщення в середині сегменту.

Процесор i8086 одночасно може працювати з чотирма сегментами пам'яті, початкові адреси яких визначають чотири сегментних регістри: CS, DS, SS, ES. Сегментні регістри містять різні види інформації виконуваної програми:

• CS – сегмент команд, містить коди виконуваної програми;

• DS – сегмент даних, містить необхідні дані, проміжні та кінцеві результати обчислень;

• SS – сегмент стека, призначений для зберігання адреси тимчасових зупинок програм та необхідних даних для продовження виконання перерваної програми;

• ES – додатковий сегмент даних, обов'язків при виконанні дій з командами обробки строк.

У сегментних регістрах містяться старші 16 розрядів адреси сегментів пам'яті. 20-розрядна фізична адреса сегмента пам'яті в чотирьох молодших розрядах завжди містить нулі. Адреса зміщення в середині сегменту 16-розрядна і дозволяє звернутися до будь-якої адреси в середині 64 Кбайтного сегменту.

**Методи адресації операндів**

Методи адресації – це правила, за якими визначаються безпосередньо операнди, їхнє місцезнаходження, а також адреси розміщення результатів обчислень. Можна виділити кілька основних методів адресації, які застосовуються для знаходження даних у сегменті пам'яті: *регістрова, безпосередня, пряма, непряма*.

У машинній команді з двома операндами можливі наступні взаємодії операндів:

регістр – регістр;

регістр – пам'ять;

пам'ять – регістр;

безпосередній операнд – регістр;

безпосередній операнд – пам'ять.

Тільки для ланцюжкових команд та при роботі зі стеком може виконуватися передача "пам'ять – пам'ять".

При **регістровій** адресації операнди знаходяться безпосередньо в регістрах процесора. У командах використовуються регістри однакового типу: байт або слово. Дана адресація часто застосовується в командах пересилки даних.

В операціях з регістровою адресацією для збереження даних використовуються:

• 16-розрядні регістри АХ, ВХ, СХ, DX;

• 8-розрядні регістри AH, AL, BH, BL, CH, CL, DH, DL.

Наприклад, пересилання вмісту регістра ВХ в АХ і АН в AL буде представлено такими командами: mov ax, bx mov al, ah mov cl, dh

При **безпосередній** адресації операнд перебуває в коді команди і для його зберігання в команді виділяється поле довжиною 8 або 16 біт. Операнд-приймач може перебувати або в пам'яті, або в регістрі, а операнд може бути тільки операндом-джерелом. Наприклад:

mov dx, 0ba64h – команда пересилає в регістр DX шістнадцяткову константу 0BA64h; Пряма адресація вказує безпосередньо на команду, яку необхідно виконати. Пряма адресація використовується в команді безумовного переходу jmp, в якій мітка визначає нову адресу продовження виконання програми. Це переміщення можливе тільки в межах даного сегмента. Непряма адресація напряму не вказує на адресу розміщення інформації, а містить її складові.

Якщо операнд у команді взято у квадратні дужки, то їхній вміст визначає складову адреси розташування операнда. Такий вид адресації називається **непрямим**. Непряма адресація має декілька різновидів: непряма базова (регістрова) адресація; непряма індексна адресація; непряма базова індексна адресація. Ці моделі непрямої адресації різняться між собою вибором регістрів, в яких знаходяться складові частини адреси, наявністю або відсутністю зміщення в формуванні адреси.

**Непряма базова (регістрова)** адресація із зміщенням та без зміщення для доступу до окремих блоків даних використовує регістри загального призначення ВХ та ВР.

**Непряма індексна адресація** із зміщенням та без зміщення Використовують для формування ефективної адреси індексні регістри SI та DI, що значно полегшує роботу з масивами даних

Непряма базово-індексна: ефективна адреса формується як сума вмісту базового й індексного регістрів. Для визначення адреси в командах може використовуватися також зміщення.

**Завдання:**

1) Сформувати масив даних об'ємом в 10 байт.

2) Сформувати порожній масив об'ємом в 10 байт.

3) Виконати перенесення даних з першого масиву в другий, використовуючи методи непрямої адресації (базової, індексної та базово-індексної).

4) За допомогою команди XLAT визначити i-й елемент другого масиву (i-й елемент – остання цифра порядкового номера в списку групи).

**А) Базова:**

Dat Segment

a db 1,2,3,4,5,6,7,8,9,10

b db 10 dup (?)

Dat EndS

Codl Segment

assume ds:dat, cs:codl

start:

mov ax, dat

mov ds, ax

mov bx, offset a

mov bp, offset b

mov ax, [bx]

mov [bp], ax

mov ax, [bx + 1]

mov [bp + 1], ax

mov ax, [bx + 2]

mov [bp + 2], ax

mov ax, [bx + 3]

mov [bp + 3], ax

mov ax, [bx + 4]

mov [bp + 4], ax

mov ax, [bx + 5]

mov [bp + 5], ax

mov ax, [bx + 6]

mov [bp + 6], ax

mov ax, [bx + 7]

mov [bp + 7], ax

mov ax, [bx + 8]

mov [bp + 8], ax

mov ax, [bx + 9]

mov [bp + 9], ax

Codl EndS

End start

**Б) Індексна:**

Dat Segment

a db 1,2,3,4,5,6,7,8,9,10

b db 10 dup (?)

Dat EndS

Codl Segment

assume ds:dat, cs:codl

start:

mov ax, dat

mov ds, ax

mov si, offset a

mov di, offset b

mov ax, [si]

mov [di], ax

mov ax, [si + 1]

mov [di + 1], ax

mov ax, [si + 2]

mov [di + 2], ax

mov ax, [si + 3]

mov [di + 3], ax

mov ax, [si + 4]

mov [di + 4], ax

mov ax, [si + 5]

mov [di + 5], ax

mov ax, [si + 6]

mov [di + 6], ax

mov ax, [si + 7]

mov [di + 7], ax

mov ax, [si + 8]

mov [di + 8], ax

mov ax, [si + 9]

mov [di + 9], ax

Codl EndS

End start

**В) Базово-індексна:**

Dat Segment

a db 1,2,3,4,5,6,7,8,9,10

b db 10 dup (?)

Dat EndS

Codl Segment

assume ds:dat, cs:codl

start:

mov ax, dat

mov ds, ax

mov si, offset a

mov di, offset b

mov bx, 0

mov ax, [si + bx]

mov [di], ax

add ax, 1;

mov ax, [si + bx]

mov [di + bx], ax

add bx, 1;

mov ax, [si + bx]

mov [di + bx], ax

add bx, 1;

mov ax, [si + bx]

mov [di + bx], ax

add bx, 1;

mov ax, [si + bx]

mov [di + bx], ax

add bx, 1;

mov ax, [si + bx]

mov [di + bx], ax

add bx, 1;

mov ax, [si + bx]

mov [di + bx], ax

add bx, 1;

mov ax, [si + bx]

mov [di + bx], ax

add bx, 1;

mov ax, [si + bx]

mov [di + bx], ax

add bx, 1;

mov ax, [si + bx]

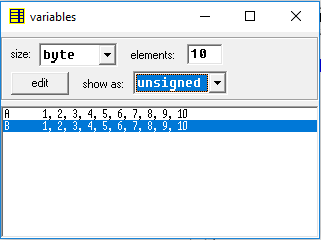
mov [di + bx], ax

add bx, 1;

Codl EndS

End start

**Результат виконання програми:**



**Висновок**: на цій лабораторній роботі я вивчив методи адресації МП І8086 та організації доступу до пам'яті, методи формування фізичної адреси та одержання доступу до будь-якої точки пам'яті.