Зміст лекції

1. Робота з масивами
2. Приклади
3. Математичний співпроцесор

Інструкція LEA (Load Effective Address) у асемблері x86-32 використовується для завантаження ефективної адреси операнда (не змісту операнда) в регістр. Ця інструкція корисна для обчислення адреси операнда та може бути використана для швидкого обчислення адрес пам'яті, індексації масивів і т.д.

Формат команди LEA виглядає так:
LEA destination, source
Де: destination: Регістр, в який буде завантажена ефективна адреса.
source: Операнд, адреса якого буде завантажена в регістр destination.

Розглянемо приклад, як можна використовувати LEA для обчислення суми елементів з використанням ефективної адреси елемента масиву myArray1 [Len1]:

Приклад 1

```
mov si, 0; //Обнулення регістру SI (індекс зміщення масиву) аналог xor si, si;
mov cx, Len1; //поміщаємо в лічильник довжину масива myArray1

begin:
lea dx, [si + myArray1]; //Ефективна адреса елемента myArray1[si];
//Далі можна використовувати адресу, наприклад, для завантаження значення елемента в регістр:
mov bx, [dx]; //Завантаження значення myArray1[si] в регістр bx
add ax, bx;
add si, 2; // додаємо 2 байти до довжини для посилання на наступний елемент, оскільки у нас масив 16-бітних значень.

loop begin
mov result, ax;
```

Відповідно при роботі з двовимірним масивом слід мати на увазі, що комірки пам’яті розташовані одна за одною.

Розглянемо аналогічний попередньому приклад – обрахуємо суму всіх елементів масиву myArray2 [Rows][Cols].

Приклад 2.1

```
mov si, 0;
mov di, Cols;
mov cx, Rows;

Rows_cycle:
    lea dx, [si+myArray2]
    push cx;
    mov cx, di;

    Cols_cycle:
        mov bx, [dx];
        add ax, bx;
        add si, 2;
        loop Cols_cycle;

    pop cx;
    loop Rows_cycle;

mov result, ax
```

Інструкція LODS (Load String) у асемблерній мові x86 використовується для завантаження значення байта, слова або двійкового слова (в залежності від режиму адресації) з області пам'яті, на яку вказує регістр ESI/SI (у режимі адресації з областю пам'яті, що містить строку або масив байтів). Після завантаження значення LODS збільшує регістр ESI/SI на розмір операції (1 байт, 2 байта або 4 байти).

Синтаксис інструкції виглядає наступним чином:

```
LODSB    ; Завантаження байта в регістр AL
LODSW    ; Завантаження слова (2 байти) в регістр AX
LODSD    ; Завантаження двійкового слова (4 байти) в регістр EAX
```

Основне використання LODS - це операції з рядками чи масивами, коли вам потрібно послідовно обробляти елементи. Ця інструкція може бути корисною при роботі з рядками тексту, обробці масивів або копіюванні даних.

Приклад використання LODSB для зчитування байтових значень з області пам'яті, що починається з адреси, яку вказує ESI/SI:

Розглянемо аналогічний попередньому приклад – обрахуємо суму всіх елементів масиву myArray2 [Rows][Cols].

Приклад 2.2

```
lea si, myArray2; //поміщаємо в лічильник зміщення (положення першого елемента) масива myArray2
    mov cx, len; // поміщаємо в лічильник довжину масива (кількість елементів). На мові C short int len = Rows*Cols;
begin:
    lodsw; // завантажуємо в AX перший елемент та додаємо до регістра SI довжину слова – 2 байти.
    add bx, ax;
loop begin;
    mov result, bx;
```

Математичний співпроцессор (також відомий як FPU - Floating Point Unit) - це частина процесора, яка спеціалізується на виконанні операцій з рухомою комою (операції, які включають десяткові або дробові числа). У контексті архітектури x86 та IA-32, математичний співпроцесор входить в склад процесора і виконує операції, які були спеціально розроблені для обробки чисел з плаваючою комою.

Основні характеристики включають:

Операції з рухомою комою: Виконання операцій додавання, віднімання, множення та ділення для чисел з плаваючою комою.

Реєстри FPU: Математичний співпроцесор має власний набір регістрів, які використовуються для зберігання та виконання операцій над числами з плаваючою комою. Реєстри FPU мають назви від ST(0) до ST(7).

Інструкції FPU: Інструкції, які виконують операції з рухомою комою, такі як FADD (додавання), FSUB (віднімання), FMUL (множення), FDIV (ділення), тощо.

Режими роботи: Підтримка різних режимів роботи, таких як режим реального режиму та захищеного режиму, а також 32-бітний та 64-бітний режими (залежно від архітектури процесора)..

Підтримка стандартів: Підтримка стандартів чисел з плаваючою комою, зокрема IEEE 754.

Векторні розширення (SSE, AVX): У новіших версіях архітектури x86 були введені векторні розширення, такі як SSE (Streaming SIMD Extensions) та AVX (Advanced Vector Extensions), які розширюють можливості обробки чисел з плаваючою комою та інших операцій.

Він може виконувати трансцендентні операції з більшою точністю.

Трансцендентні операції - це операції, які включають в себе математичні функції, що виходять за рамки арифметичних операцій та включають в себе складніші математичні функції, такі як функції експоненти, логарифми, тригонометричні та інші подібні.

Основні трансцендентні функції включають:

Експоненційна функція (exp(x)): Функція, яка підносить число e (приблизно 2.71828) до ступеня x.

Логарифмічні функції (log(x)): Логарифми, які є оберненими функціями експоненцій.

Тригонометричні функції (sin(x), cos(x), tan(x)): Функції, які взаємодіють з кутами трикутників.

Гіперболічні функції (sinh(x), cosh(x), tanh(x)): Функції, які взаємодіють з гіперболами.

Ці функції важливі в численних областях математики, фізики, інженерії та інших наукових галузях. Програмісти та математики використовують трансцендентні операції для вирішення різноманітних завдань, включаючи моделювання, обчислення, обробку сигналів та багато іншого. Трансцендентні операції також широко використовуються в області комп'ютерного програмування та обчислень.

Математичні співпроцесори дозволяють прискорити операції, пов'язані з числами з плаваючою комою, і були особливо важливими в епоху, коли вони були додатковою частиною процесора. З плином часу із зростанням потужності процесорів і появою багатоядерних архітектур, деякі функції математичного співпроцесора були інтегровані безпосередньо в основний процесор.

Співпроцесор може виконувати операції з сімома звичайними типами даних, а також із спеціальними числами. Звичайні дані можуть бути дійсними або цілими числами із знаком. Наявність цілочисельних даних дозволяє виконувати так звані змішані обрахування, коли в якості операндів використовуються і цілі, і дійсні числа. Співпроцесор виконує всі обрахування в 80бітному розширеному дійсному форматі, а 16-, 32- і 64-бітні дані використовуються для обміну даними і можуть застосовуватись в командах співпроцесора як операнди.

Типи звичайних даних співпроцесора

Тип даних	Довжина (біт)	Кількість значимих цифр	Діапазон представлення
Ціле слово	16	4-5	-32768…32767
Коротке ціле	32	9-10	-2147483648 … - 2147483647
Довге ціле	64	18-19	9223372036854775808 … 9223372036854775807
Упаковане двійководесяткове	80	18	-999999999999999999 … 999999999999999999
Коротке дійсне	32	7-8	1.175494351e-38 … 3.402823466e+38
Довге дійсне	64	15-16	2.2250738585072014e-308... 1.7976931348623158e+308
Розширене дійсне	80	19-20	3.3621031431120935063e4932… 1.189731495357231765e+4932

Крім звичайних чисел, специфікація стандарту IEEE передбачає декілька спеціальних форматів, які можуть виникнути в результаті виконання математичних операцій співпроцесора і над якими можна виконувати окремі операції.

- Додатний нуль – всі біти числа скинуті до нуля.
- Від’ємний нуль – знаковий біт дорівнює 1, всі інші біти числа скинуті до нуля.
- Додатна нескінченність – знаковий біт дорівнює 0, всі біти експоненти установлені в 1, а біти мантиси скинуті до нуля.
- Від’ємна нескінченність - знаковий біт дорівнює 1, всі біти експоненти установлені в 1, а біти мантиси скинуті до 0.
- Денормалізовані числа – всі біти експоненти скинуті до 0. Ці числа дозволяють представляти дуже маленькі числа при обрахунках з розширеною точністю.
- Невизначеність – знаковий біт дорівнює1, перший біт мантиси дорівнює 1 (для 80- розрядних чисел перші два біта дорівнюють 11), інші біти мантиси скинуті до нуля, всі біти експоненти встановленні в 1.
- Не-число типу SNAN (сигнальне) – перший біт мантиси дорівнює 0 (для 80-розрядних чисел перші два біти дорівнюють 10), інші біти мантиси можуть містити 0 або 1, всі біти експоненти встановлені в 1.
- Не-число типу QNAN (тихе) – перший біт мантиси дорівнює 1 (для 80-розрядних чисел перші два біти дорівнюють 11), інші біти мантиси можуть містити 0 або 1, всі біти експоненти встановлені в 1.
- Непідтримуване число – всі інші ситуації.