Архітектура комп’ютера

Лекція 12

Лектор:

доцент кафедри КІ та КБ,
к.т.н. Шелуха Олексій

Зміст лекції

- Робота з масивами
- Приклади
- Математичний співпроцесор

Інструкція LEA

- Інструкція LEA (Load Effective Address) – використовується для завантаження ефективної **адреси 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-приймач, в який буде завантажена ефективна адреса.
- source: Операнд-джерело, адреса якого буде завантажена в регістр destination.

Приклад 1. Обчислення суми елементів з використанням ефективної адреси елемента масиву myArray1 [Len1]

```
    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;
```

Приклад 2.1. Обчислення суми всіх елементів масиву myArray2 [Rows][Cols].

```
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

- Інструкція LODS (Load String) – 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

Приклад 2.2. Обчислення суми всіх елементів масиву myArray2 [Rows][Cols].

```
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

- 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). Також 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:
  - **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.

Основні характеристики:

- **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.

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	-99999999999999999 ... 99999999999999999
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	3.362103143112093506 3e4932... 1.1897314953572317 65e+4932

Спеціальні формати результатів математичних операцій

- **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.

Спеціальні формати результатів математичних операцій

- **Невизначеність** – знаковий біт дорівнює 1, 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.

Дякую за увагу!