Архітектура комп’ютера

Лекція 6

Лектор:

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

Зміст лекції

- Синтаксис мови Assembler
- Типи даних в мові Assembler
- Основні арифметичні команди
- Приклад програми в Assembler
- Нестандартні ситуації

Синтаксис мови Assembler

- Команди
- Макрокоманди
- Директиви
- Коментар

- Запис виглядає наступним чином:
  - Команда (Макрокоманда, директива) ;коментар

Структура команди

- Мітка : КОД ОПЕРАЦІЇ {Операнд1, Операнд 2} ; коментар

- Приклад:
  - start1: ADD AX, BX ; Додавання значення регістра BX до значення регістра AX із збереженням результату в регістрі AX

Алфавіт Assembler

- Літери латинського алфавіту (A-Z, a-z)
- Цифри (0-9)
- Спеціальні символи (?, &, $, @ …)
- Роздільники (,), [,], {, }, <,>, +, -, /, * …
- Ключові слова:
  - назви регістрів (AL, AH, AX, EAX, BL, ….)
  - оператори (BYTE, WORD, FAR, NEAR…)
  - назви команд (ADD, NEG, INC, LOOP, JE …)
- Ідентифікатори
- Ланцюжки символів: ‘text’, “text”
- Цілі числа

Операнди

- **Операнди команди** – об’єкти, над якими або за допомогою яких можуть проводитися дії, що задаються відповідними командами.
- Операнди:
  - регістр;
  - пам’ять;
  - безпосередні операнди (як правило, це цілі числа).

Команди

- Команди можуть бути:
- одноопераднними,
- двоопераднними,
- три+ операндними (зазвичай це макрокоманди).

Одноопераднi команди

- Одноопераднi можуть використовувати регiстри або пам’ять:
- Приклад: NEG EAX ⇔ neg eax

Двооперандні команди

- Регістр – Регістр SUB AX, BX
- Регістр – Пам’ять ADD AH, a
- Пам’ять – Регістр ADD a, AH
  - Команди пам’ять – пам’ять немає, ми повинні скористатись проміжним елементом – регістром
- Регістр – Безпосередній операнд ADD BL, 123
- Пам’ять – Безпосередній операнд ADD a, 123

Триопераднні команди

- В більшості випадків вони є макрокомандами, що скадаються з декількох двоопераднних команд
- Приклад:
- ADD AX, BX, CX ⇔ ADD AX, BX
  MOV CX, AX

Типи даних Assembler

- В Assembler типи даних – це значення в пам’яті певної довжини.
- Основні типи даних:
  - Байт, byte (8 розрядів, бітів)
  - Слово, word (2 байти) = (Старший байт, Молодший)
  - Подвійне слово, double word (4 байти) = (Старше, Молодше)
  - Зчетверене слово, quad word (8 байтів)
  - Упакований тип даних, double-quad, xmmword (16 байтів)
  - Десятибайтний, ten bytes (10 байтів)
- Цілі типи даних (8, 16, 32, 64, 128, … біт)
  - знаковий
  - беззнаковий
- Дійсні типи даних. Завжди знакові. (32, 64, 80, 128, … біт)
- Ланцюжки символів – можуть використовуватись для роботи з пам'яттю,
- Дані MMX-розширень тощо.

Основні арифметичні команди мови Assembler

- 1. Для виконання операцій додавання та віднімання операнди повинні мати однакову довжину:
- 8 біт +/- 8 бітів = 8 бітів
- 16 бітів +/- 16 бітів = 16 бітів
- 32 біти +/- 32 біти = 32 біти
- 64 біти +/- 64 біти = 64 біти

- Якщо нам потрібно від 16 бітного відняти 8 бітного – то друге число потрібно перетворити у шістнадцятибітне

Основні арифметичні команди мови Assembler

- 2. При виконанні операції множення операнди повинні мати однакову довжину, а результат довжину в два рази більшу:
- 8 біт * 8 бітів = 16 бітів
- 16 бітів * 16 бітів = 32 бітів
- 32 біти * 32 біти = 64 біти
- 64 біти * 64 біти = 128 біти

Основні арифметичні команди мови Assembler

- 3. При виконанні операції ділення ділене повинне мати довжину в два рази більше ніж дільник та частка:
- 16 бітів : 8 бітів = 8 бітів
- 32 біти : 16 бітів = 16 бітів
- 64 біти : 32 біти = 32 біти
- 128 бітів : 64 біти = 64 біти

Основні арифметичні команди мови Assembler

- Команди перетворення типів
  - cbw – розширює регістр AL до регістра AX
  - cwd – розширює регістр AX до регістра EAX (DX:AX)
  - cdq – розширює регістр EAX до регістра REAX

- Команда пересилки MOV
  - MOV Операнд1, Операнд2

- Команда обміну вмістом операндів
  - XCHG Операнд1, Операнд2

Основні арифметичні команди мови Assembler

- Команда зміни знаку
  - NEG Операнд
- Команди інкременту та декременту
  - INC Операнд
  - DEC Операнд
- Команди додавання та віднімання
  - ADD Операнд1, Операнд2
  - SUB Операнд1, Операнд2

Приклад

- Розрахувати значення виразу x = -(a+b-c)-1, де x, a, b,c, - 8-бітні знакові числа.
  - int main ()
  - { signed char a, b, c, x1, x2;
  - printf("a = "); scanf_s("%d", &a);
  - printf("b = "); scanf_s("%d", &b);
  - printf("c = "); scanf_s("%d", &c);
  - x1 = -(a+b-c)-1;
  - printf("Result C x = %d\n", x1);

Приклад

```
__asm {
        MOV AL, a ; <AL>:=a
        MOV AH, b; <AH>:=b
        MOV BL, c; <BL>:=c
        ADD AL,AH; <AL>:= a+b, <AH>:=b, <BL>:=c
        SUB AL, BL; <AL>:= a+b-c, <AH>:=b, <BL>:=c
        NEG AL; <AL>:= -(a+b-c), <AH>:=b, <BL>:=c
        DEC AL; <AL>:= -(a+b-c)-1, <AH>:=b, <BL>:=c
        MOV x2, AL; x := -(a+b-c)-1, <AL>:= -(a+b-c)-1, <AH>:=b, <BL>:=c
  }
  printf("Result ASM x = %d\n", x2);
  system("Pause");
  return 0;
}
```

Нестандартні ситуації. Приклад 1

- Розрахувати вираз x=a+b, де x,a,b – 8-бітні беззнакові числа.
- 8-бітні беззнакові числа належать до діапазону 0 .. 255.
- a = 10; b = 10; x = 10 + 10 = 20 – **все добре**
- a = 150; b = 200; x = 150 + 200 = 350 – число не потрапляє в діапазон 0 .. 255 **ПЕРЕПОВНЕННЯ !**

- Фізично (побітово) переповнення
- $150_{10} = 10010110_2$; $200_{10} = 11001000_2$
- Додавання
- 10010110
- 11001000
- $101011110_2 = 350_{10}$
- $01011110_2 = 94_{10}$

Правила двійкового додавання
0+0 = 0
1+0 = 1
0+1 = 1
1+1 = 10 = ^{1}0

Нестандартні ситуації. Приклад 2

- Число **a** 8-бітне беззнакове,
- a = 10_{10} = 00001010_2

- Після виконання команди NEG до знач. 00001010_2 отримано результат 10001010_2,
- але число ми використовуємо як 8-бітне беззнакове, тоді результатом буде число 138_{10}

Дякую за увагу!

Список використаних джерел

- 1. Тараракa В.Д. Архітектура комп'ютерних систем: навчальний посібник. Житомир: ЖДТУ, 2018. 383 с.
- 2. David Harris, Sarah Harris. Digital Design and Computer Architecture. 2nd Edition. Morgan Kaufmann. 2012. 720p.
- 3. Assembler. Підручник для ВНЗ. В.І. Юров. 2003. 637 с.