Тема: Представление целых чисел в памяти ЭВМ

Цель: изучение принципов представления числовой информации в памяти ЭВМ; приобретение практических навыков представления целых чисел со знаком в машинном формате.

Задание:

1. Представить целые числа I1 иI2 в формате DB, DW, DD.
2. Составить и откомпилировать программу, определив число I1 в форматах DB, DW, DD, а число I2 в форматах DW, DD.
3. Задать такие операции пересылки данных:

–загрузить регистр R1 числом I2 из сегмента данных;

–с использованием заданного варианта косвенной адресации записать содержимое R1 в сегмент данных со смещением на I1 байт относительно метки I2 (число I1 предварительно загрузить в соответствующий базовый или индексный регистр, при наличии других операндов в формуле адресации определить их произвольно).

1. Проверить результаты расчетов и пересылок в дампе памяти.
2. Сделать расчет времени выполнения программы.

Ход работы:

1. Полученные числа I1 = 202 и I2=-130 переведём в шестнадцатеричный код, и вычислим их представление в дампе памяти:

|  |  |  |
| --- | --- | --- |
| Операция | Формат DB | Формат DW |
| 1) модуль в двоичной форме | 1100 1010 | 1100 1010 |
| 2) дописываем 0 | 1100 1010 | 00000000 11001010 |
| 3) в 16-ичном дампе | СА | 00 СА |
| 4) в дампе памяти | СА | СА 00 |
| Формат DD | | |
| 1) модуль в двоичной форме | 1100 1010 | |
| 2) дописываем 0 | 00000000 00000000 00000000 11001010 | |
| 3) в 16-ичном дампе | 00 00 00 СА | |
| 4) в дампе памяти | СА 00 00 00 | |

Рис. 1 – перевод числа 202 в 16-ый код.

|  |  |
| --- | --- |
| Операция | Формат DW |
| 1) модуль в двоичной форме | 1000 0010 |
| 2) дописываем 0 | 00000000 10000010 |
| 3) инвертируем и прибавляем 1 | +11111111 01111101  1  11111111 01111110 |
| 4) в 16-ичном дампе | FF 7E |
| 5) в дампе памяти | 7E FF |
| Формат DD | |
| 1) модуль в двоичной форме | 1000101101 |
| 2) дописываем 0 | 00000000 00000000 00000010 00101101 |
| 3) инвертируем и прибавляем 1 | 11111111 11111111 11111101 11010011 |
| 4) в 16-ичном дампе | FF FF FD D3 |
| 5) в дампе памяти | 7E FF FF FF |

Рис.2 – перевод числа -130 в 16-ый код.

2. На основе вычисленных значений составим код на языке ассемблер, и проверим находятся ли вычисленные значения в памяти ЭВМ. Объявляем значения при помощи ключевых слов db, dw, dd. Код программы:

.data

I1\_DB db 202

I1\_DW dw 202

I1\_DD dd 202

I2\_DW dw -130

I2\_DD dd -130

3. Код пересылки данных в регистры R1 и косвенную адресацию:

.code

start:

mov BX,I2\_DW

mov EBP, offset I1\_DW

mov [EBP], BX

exit

end start

4. Скомпилировав и выполнив программу, посмотрим дампы памяти:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок 1 – значения, хранящиеся в памяти до работы программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2 – регистры, хранящиеся до работы программы

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3 - значения, хранящиеся в памяти после работы программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4 - регистры, хранящиеся после работы программы

5. Посчитаем время выполнения данной операции:

1. MOV BX(регистр), I2\_DW(память) – 12(слова)+6 = 18 тактов.

2. MOV EBP(регистр), I1\_DW (память) – 12(слова)+5(база) = 17 тактов.

4. MOV память, регистр – 13(слова)+5(база) = 19 тактов.

Общее время выполнения программы: 54 такта.

1. Результаты:

В пункте 1 задания была проделана работа по переводу чисел из десятичной системы счисления в шестнадцатеричную, и была представлена таблица результатов проделанной работы.

В задании 2 были изучены типы данных, структур программы на языке assembler и способ представления чисел в памяти.

В задании 3 была написана функция для работы с сегментами данных и косвенной адресацией.

Для задания 4 использовался встроенный в RadAsm инструмент Debugger для анализа и представления памяти, результат представлен в виде рисунков.

В задании 5 был проведён расчёт времени выполнения созданного кода.

Вывод: был изучен принцип работы машинной памяти и способ представления информации в ней. Были изучены основы языка ассемблер и написана программа для выполнения задания.