**Лабораторная работа №1**

**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**

Вариант 27

1. Представить целые числа 59 и -1530 (табл. 1.4) в формате DB, DW, DD.
2. Составить и откомпилировать программу, определив число 59 в форматах DB, DW, DD, а число -1530 в форматах DW, DD.
3. Задать такие операции пересылки данных:

* загрузить регистр DX числом -1530 из сегмента данных;
* с использованием заданного варианта косвенной адресации ([BX][DI]) записать содержимое DX в сегмент данных со смещением на 59 байт относительно метки числа -1530 (число 59 предварительно загрузить в соответствующий базовый или индексный регистр, при наличии других операндов в формуле адресации определить их произвольно).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № вар. | I1 | I2 | R1 | Косвенная адресация |
| 22 | 59 | -1530 | DX | [BX][DI] |

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

## Ход работы

* 1. Представление чисел в форматах DB, DW, DD:

Представление числа 59 в форматах DB, DW, DD представлено в таблице 1.

Таблица 1. Представление числа 59 в форматах DB, DW, DD.

|  |  |  |  |
| --- | --- | --- | --- |
| Операция | | Формат DB | Формат DW |
| 1) модуль в двоичной форме | | 11 1011 | 11 1011 |
| 2) дописываем 0 | | 0011 1011 | 0000 0000 0011 1011 |
| 3) в 16-ричном виде | | 3B | 00 3B |
| 4) в дампе памяти | | 3B | 3B 00 |
| Формат DD | | | |
| 1) модуль в двоичной форме | 11 1011 | | |
| 2) дописываем 0 | 0000 0000 0000 0000 0000 0000 0011 1011 | | |
| 3) в 16-ричном виде | 00 00 00 3B | | |
| 4) в дампе памяти | 3B 00 00 00 | | |

Представление числа -1530 в форматах DW, DD представлено в таблице 2.

Таблица 2. Представление числа -1530 в форматах DB, DW, DD.

|  |  |  |  |
| --- | --- | --- | --- |
| Операция | | Формат DW | |
| 1) модуль в двоичной форме | | 101 1111 1010 | |
| 2) дописываем 0 | | 0000 0101 1111 1010 | |
| 3) инвертируем и прибавляем 1 | | 1111 1010 0000 0110 | |
| 4) в 16-ричном виде | | FA 06 | |
| 5) в дампе памяти | | 06 FA | |
| Формат DD | | |
| 1) модуль в двоичной форме | 101 1111 1010 | |
| 2) дописываем 0 | 0000 0000 0000 0000 0000 0101 1111 1010 | |
| 3) инвертируем и прибавляем 1 | 1111 1111 1111 1111 1111 1010 0000 0110 | |
| 4) в 16-ричном виде | FF FF FA 06 | |
| 5) в дампе памяти | 06 FA FF FF | |

* 1. Программа ниже определяет переменные I1B, I1W, I1D (число I1 в форматах DB, DW, DD) и I2W, I2D (число I2 в форматах DW, DD), после чего выполняет пересылки в заданные адреса:

## .data

## I1B DB 59

## I1W DW 59

## I1D DD 59

## 

## I2W DW -1530

## I2D DD -1530

## 

## M dw 1000 dup(0)

## .code

## LStart:

## 

## MOV AX, I2W

## MOV EBX, OFFSET I2W

## 

## 

## MOV EDI, I1D

## MOV [EBX + 0][EDI], AX

## 

## exit

## end LStart

Программа сначала определяет адрес [BX][DI] по правилу:

EBX – адрес переменной I2W (число -1530 в формате DW);

EDI – число 59.

Далее число -1530 в виде слова записывается по адресу [EBX][EDI] через регистр AX.

* 1. В отладчике был открыт дамп сегмента данных, в котором видны адреса объявленных переменных. На рисунке 1 показано состояние дампа памяти до записи значения переменной I2W с использованием косвенной адресации.

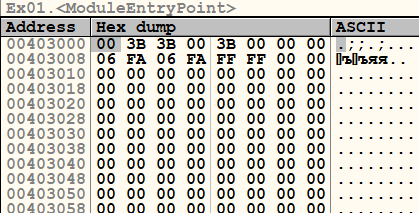


Рисунок 1 – Состояние дампа памяти до выполнения кода программы.

На рисунке 2 показано состояние дампа памяти после выполнения кода программы. На нем видно, что значение переменной I2W было записано со смещением 59 байт относительно позиции переменной I1W.

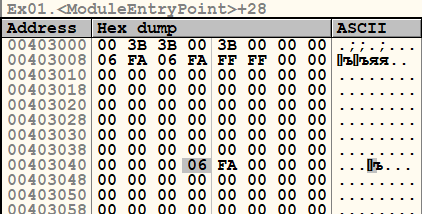


Рисунок 2 – Состояние дампа памяти после выполнения кода программы.

Рисунок 2 – дамп памяти по адресам 0098-00B0

* 1. Расчет времени выполнения программы представлен ниже:

.code

LStart:

MOV AX, I2W ; 12 + 6 = 18

MOV EBX, OFFSET I2W ; 12 + 6 = 18

MOV EDI, I1D ; 12 + 6 = 18

MOV [EBX + 0][EDI], AX ; 13 + 7 = 20

exit

end LStart

## Результат работы

## В ходе выполнения лабораторной работы было вычислено время исполнения написанной программы – 74 такта, или 74\*210 = 15540 наносекунд.

## Вывод

В данной работе была написана и проанализирована программа на языке Ассемблера, в процессе чего было изучено представление числовой информации в памяти компьютера.