МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

Отчет по лабораторной работе №1

по дисциплине «Цифровые устройства и МЦ»

Вариант 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнил: | студент группы  ИНБб-3301-02-00 |  |  | И.Д. Щукин |
|  |  |  |  |  |
| Проверил: | Преподаватель |  |  | М.А. Земцов |
|  |  |  |  |  |

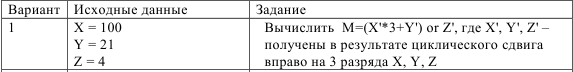
Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_ 2025 г.

г. Киров

2025

1. ЛАБОРАТОРНАЯ РАБОТА № 1 «СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА X86»

**Цель работы**: изучение системы команд и способов адресации микропроцессоров с архитектурой x86.



Код программы:

.686

.model flat, stdcall

.stack 100h

.data

X1 dw 100 ; Исходное значение X

Y1 dw 21 ; Исходное значение Y

Z1 dw 4 ; Исходное значение Z

M dw 0 ; Результат M

overflow\_flag db 0 ; Флаг переполнения

.code

ExitProcess PROTO STDCALL :DWORD

Start:

; Циклический сдвиг вправо на 3 бита для X

mov ax, X1 ; Загружаем X в AX

ror ax, 3 ; Циклический сдвиг вправо на 3 бита

mov bx, ax ; Сохраняем X' в BX

; Циклический сдвиг вправо на 3 бита для Y

mov ax, Y1 ; Загружаем Y в AX

ror ax, 3 ; Циклический сдвиг вправо на 3 бита

mov cx, ax ; Сохраняем Y' в CX

; Циклический сдвиг вправо на 3 бита для Z

mov ax, Z1 ; Загружаем Z в AX

ror ax, 3 ; Циклический сдвиг вправо на 3 бита

mov dx, ax ; Сохраняем Z' в DX

; Вычисляем M = (X' \* 3 + Y') OR Z'

mov ax, bx ; Загружаем X' в AX

shl ax, 1 ; Умножаем на 2 (AX = X' \* 2)

add ax, bx ; AX = X' \* 2 + X' (AX = X' \* 3)

add ax, cx ; AX = (X' \* 3) + Y'

or ax, dx ; AX = ((X' \* 3) + Y') OR Z'

mov M, ax ; Сохраняем результат в M

jmp exit ; Переход к завершению программы

exit:

invoke ExitProcess, 0 ; Завершение программы с кодом 0

End Start

Переменные объявляются в сегменте .data. Типы данных ассемблера приведены в таблице 1.

Таблица 1. Типы данных ассемблера

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Директива** | **Количество байт** |  |
| Байт | DB | 1 | Символ, целое |
| Слово | DW | 2 | Целое, вещественное |
| Двойное слово | DD | 4 | Целое, вещественное |
| 8 байт | DQ | 8 | Целое, вещественное |
| 10 байт | DT | 10 | Вещественное |

Формат объявления переменной:

[имя] [тип] [значение]

Если нужно объявить переменную, но не инициализировать ее, то вместо поля [значение] ставится символ «?».

Например,

X dw 5 ; объявление переменной размером 2 байта X = 5 X dw ? ; объявление переменной без инициализации

M1 dd 0,1,2,3,4,5,6,7,8,9 ; объявление массива M1

a dd 20 dup (0) ; объявление массива a, состоящего из 20 элементов, начальные значения которых равны 0.

Для отладки программы на ассемблере можно использовать стандартные средства Visual Studio.

Чтобы посмотреть значения переменных при выполнении программы, используйте окно «Контрольные значения» (рис. 8).

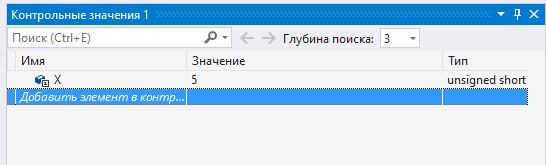
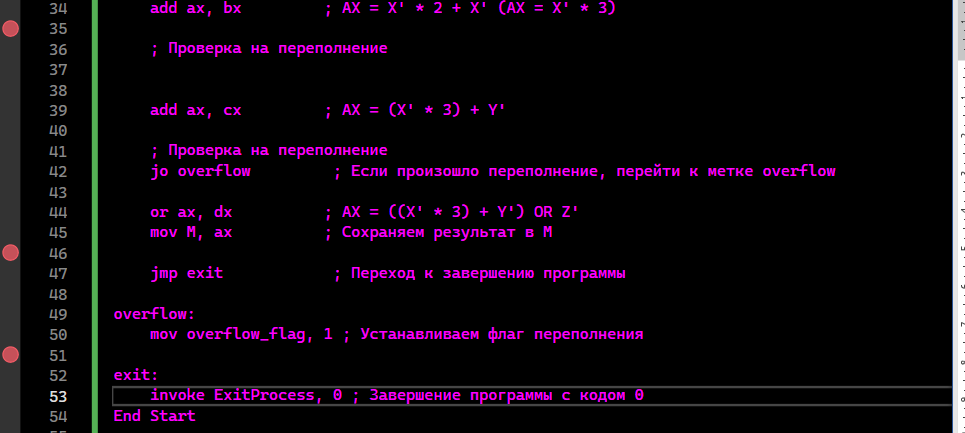
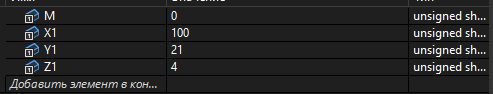


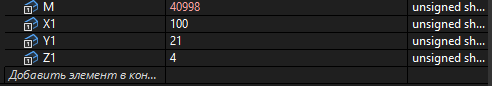
Рис. 8. Просмотр значений переменных

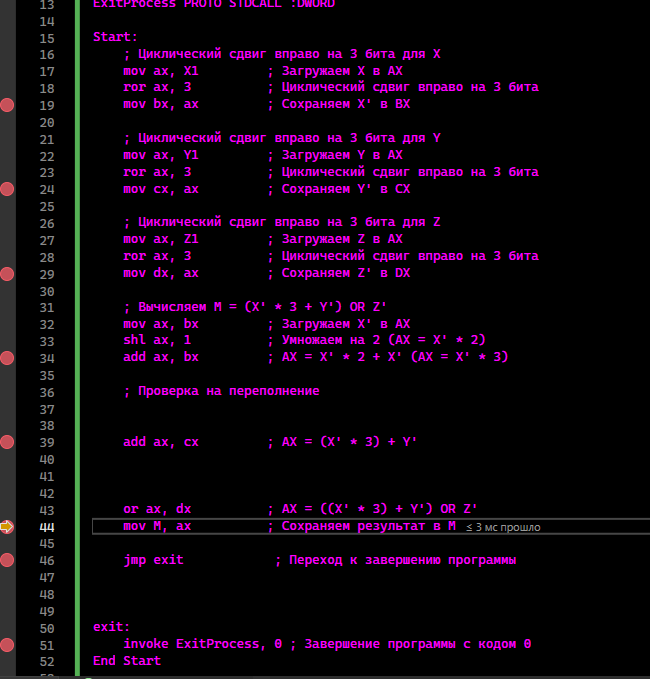
Значения переменных в точке 35



После в строке 42 происходит проверка на переполнение, которая срабатывает из чего можно сделать вывод, что во время работы программы происходит переполнение.

Если убрать проверку то в конце программы получатся такие значения:



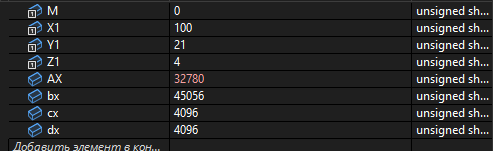


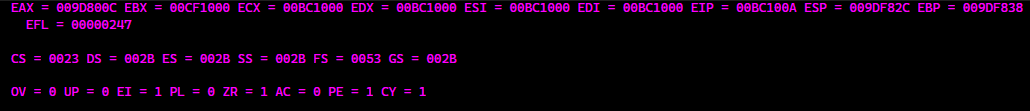
19 ; Циклический сдвиг вправо на 3 бита для X

mov ax, X1 ; Загружаем X в AX

ror ax, 3 ; Циклический сдвиг вправо на 3 бита

mov bx, ax ; Сохраняем X' в BX



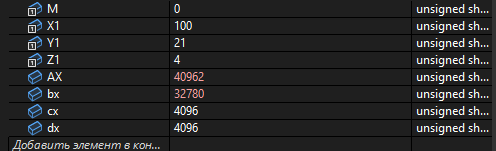


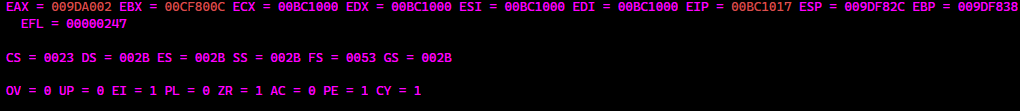
24 ; Циклический сдвиг вправо на 3 бита для Y

mov ax, Y1 ; Загружаем Y в AX

ror ax, 3 ; Циклический сдвиг вправо на 3 бита

mov cx, ax ; Сохраняем Y' в CX



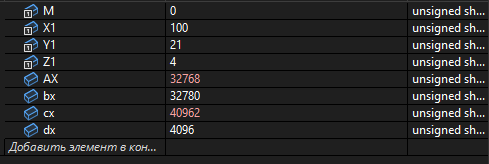


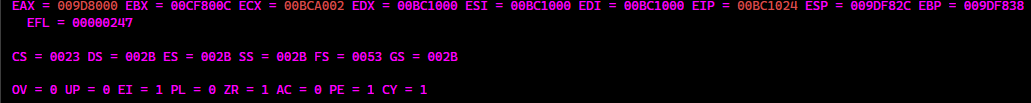
29 ; Циклический сдвиг вправо на 3 бита для Z

mov ax, Z1 ; Загружаем Z в AX

ror ax, 3 ; Циклический сдвиг вправо на 3 бита

mov dx, ax ; Сохраняем Z' в DX



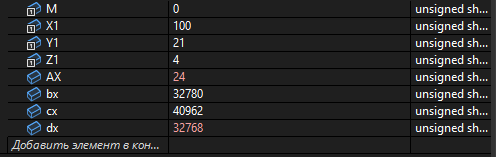


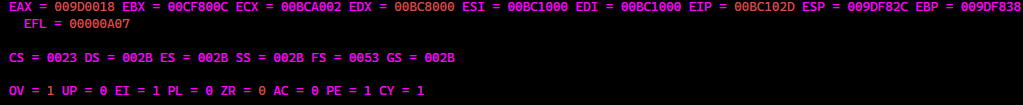
34 ; Вычисляем M = (X' \* 3 + Y') OR Z'

mov ax, bx ; Загружаем X' в AX

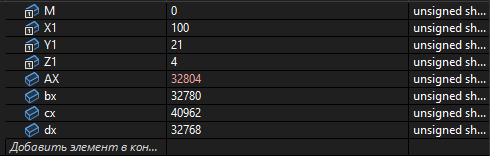
shl ax, 1 ; Умножаем на 2 (AX = X' \* 2)

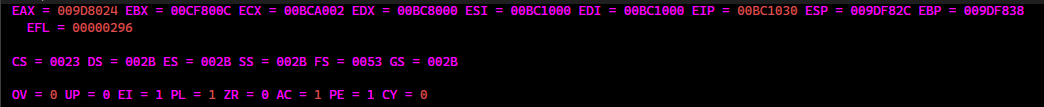
add ax, bx ; AX = X' \* 2 + X' (AX = X' \* 3)





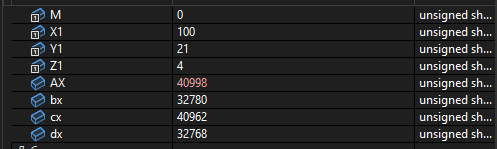
39 add ax, cx ; AX = (X' \* 3) + Y'

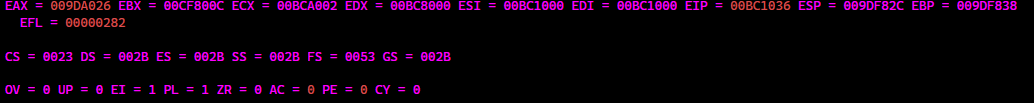




44 or ax, dx ; AX = ((X' \* 3) + Y') OR Z'

mov M, ax ; Сохраняем результат в M





Чтобы вычислить значение M. X=100, Y=21 и Z=4 найдем значения X′, Y′ и Z′.

**Шаг 1: Циклический сдвиг вправо на 3 бита**

1. **Для X=100**:
   * В двоичном представлении: 100=01100100
   * Циклический сдвиг вправо на 3 бита:
     + 01100100 становится 10001100 (последние 3 бита перемещаются в начало).
   * В десятичном представлении: 10001100=140
   * Таким образом, X′=140.
2. **Для Y=21**:
   * В двоичном представлении: 21=00010101
   * Циклический сдвиг вправо на 3 бита:
     + 00010101 становится 10100001.
   * В десятичном представлении: 10100001=129
   * Таким образом, Y′=129.
3. **Для Z=4**:
   * В двоичном представлении: 4=00000100
   * Циклический сдвиг вправо на 3 бита:
     + 00000100 становится 00010000.
   * В десятичном представлении: 00010000=16
   * Таким образом, Z′=16.

**Шаг 2: Вычисление M**

Теперь подставим найденные значения в выражение:

M=(X′⋅3+Y′) or Z′

1. **Вычисляем X′⋅3**: 140⋅3=420
2. **Вычисляем X′⋅3+Y′**: 420+129=549
3. **Теперь вычисляем M**: M=549 or 16

В двоичном представлении:

* + 549=1000101101
  + 16=00000010000

Выполним операцию OR:

1. В десятичном представлении:

100011 0101=565

**Итог**

Таким образом, значение M должно быть равно 565.

Вывод: В ходе лабораторной работы были изучены основные принципы работы с ассемблером, включая выполнение арифметических операций, использование циклического сдвига и логических операций, а также взаимодействие с Windows API для вывода информации. Работа с ассемблером позволила глубже понять, как низкоуровневое программирование взаимодействует с аппаратным обеспечением и как реализуются базовые операции на уровне машинного кода.