Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**Отчёт о выполнении лабораторной работы №1 по дисциплине «Системы реального времени»**

Работу выполнил студент 44 группы А. А. Иванов

Работу проверил доц. каф. ИТ А. Н. Полетайкин

Краснодар

2024

**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 и I2 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**).

4. Проверить результаты расчетов и пересылок в дампе памяти.

5. Сделать расчет времени выполнения программы.

Индивидуальные условия лабораторной работы представлены в таблице (Таблица 1):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | I1 | I2 | R1 | Косвенная адресация |
| 10 | 75 | -457 | DX | [BP+DI] |

Таблица 1 – индивидуальные условия лабораторной работы

**Ход работы**

*1. Представить целые числа I1 и I2 в формате DB, DW, DD.*

Для выполнения данного пункта воспользовался таблицей 1.1 из пособия и теоретическим материалом после таблицы. Результаты представления чисел представлены на рисунках (Рисунок 1, Рисунок 2):

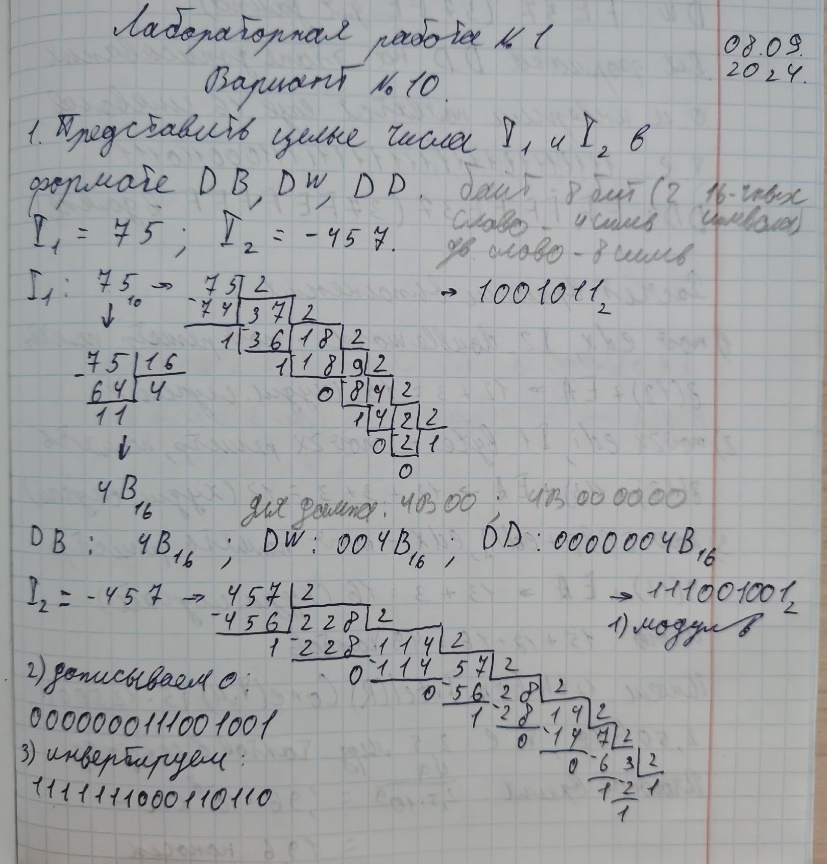


Рисунок 1 – перевод чисел I1 и I2 в форматы DB, DW и DD

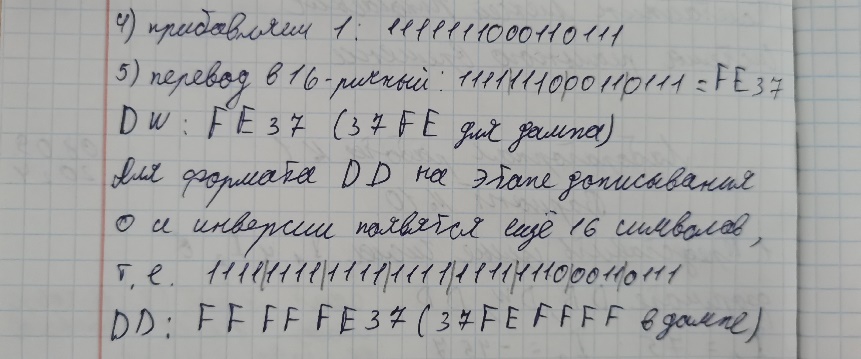


Рисунок 2 – перевод чисел I1 и I2 в форматы DB, DW, DD

По итогу получаем такие значения:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | DB | DW | DD | DW (дамп) | DD (дамп) |
| I1 | 4B | 004B | 0000004B | 4B00 | 4B000000 |
| I2 | - | FE37 | FFFFFE37 | 37FE | 37FEFFFF |

Таблица 2 – I1 и I2 в разных форматах

*2. Составить и откомпилировать программу, определив число I1 в форматах DB, DW, DD, а число I2 в форматах DW, DD.*

Для составления программы воспользовался теоретическими положениями из пособия для лабораторных работ. Готовая программа представлена на рисунке (Рисунок 3), а отдельно код программы представлен в приложении (Приложение А).

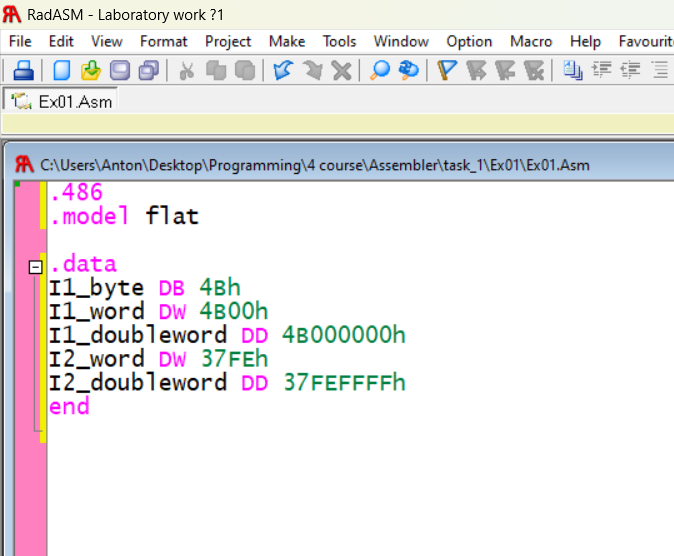


Рисунок 3 – код программы

Пояснения к коду:

– .486 – стандартная архитектура процессора для 32-битного кода;

– .model flat – модель памяти;

– .data – блок, в котором перечисляются данные, с которыми будет работать программа;

– далее следует несколько строк с определением данных в виде НАЗВАНИЕ ФОРМАТ ЗНАЧЕНИЕ;

– end – конец программы.

*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****).*

Для составления программы с операциями пересылки воспользовался теоретическими положениями из пособия для лабораторных работ и пособия по программированию в RADAsm. Код программы представлен на рисунке (Рисунок 4), а отдельно код программы в приложении (Приложение А).

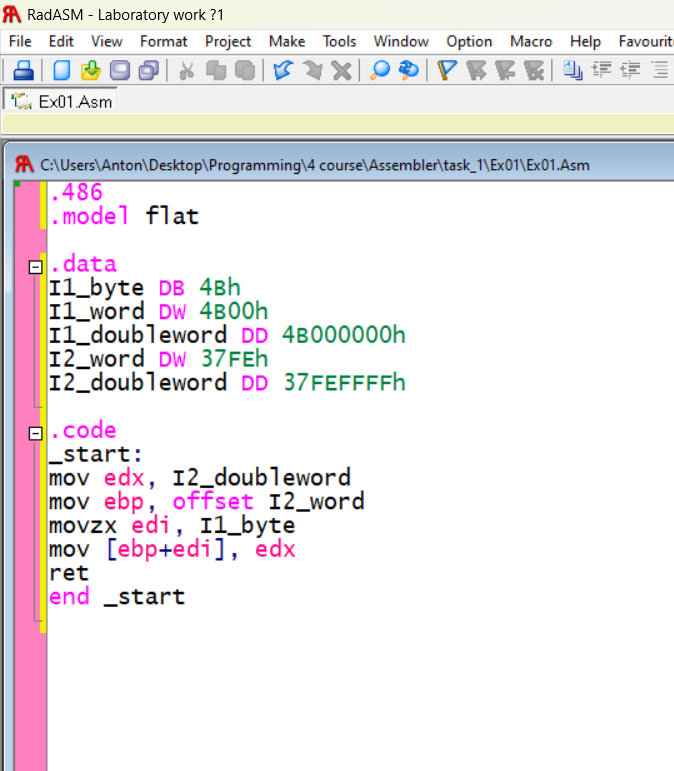


Рисунок 4 – код программы с операциями пересылки

Пояснения к коду:

– .code – секция кода;

– \_start: –начало программы;

– mov edx, I2\_doubleword – загружаем значение I2\_doubleword в 32-битный регистр EDX;

– mov ebp, offset I2\_word – загружаем в регистр EBP адрес памяти, по которому находится переменная I2\_word;

– movzx edi, I1\_byte – загружаем значение I1\_byte в 32-битный регистр EDI с расширением нулями (регистр-то 32-битный);

– mov [ebp+edi], edx – загружаем значение из регистра EDX в адрес памяти, который равен адресу записи переменной I2\_word + отступу I1\_byte байт (в данном случае 00402008 + 75 байт = 00402053);

– ret – завершение работы программы;

– end \_start – конец программы.

*4. Проверить результаты расчетов и пересылок в дампе памяти.*

Для выполнения проверки работоспособности программы переходим в отладчик. Далее представлен на рисунках (Рисунок 5, Рисунок 6) дамп памяти до выполнения программы и после.

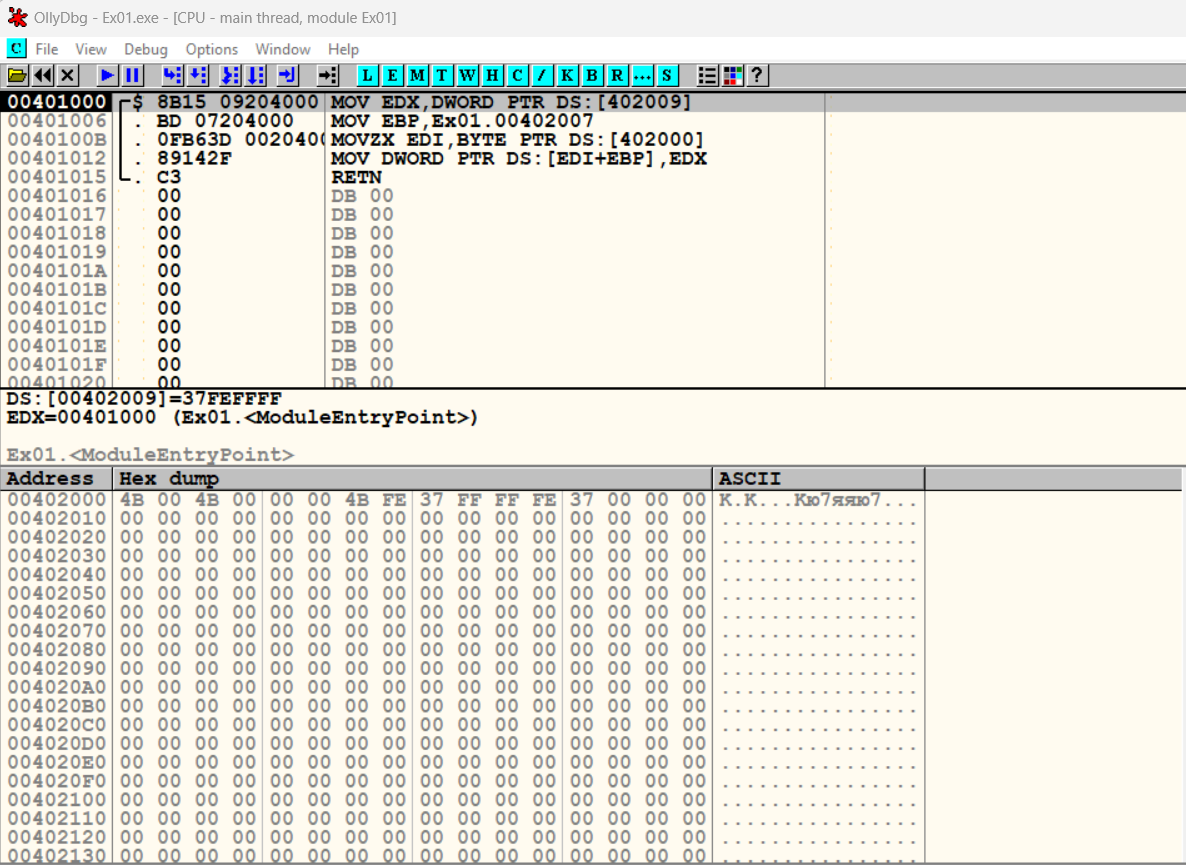


Рисунок 5 – дамп памяти до выполнения программы

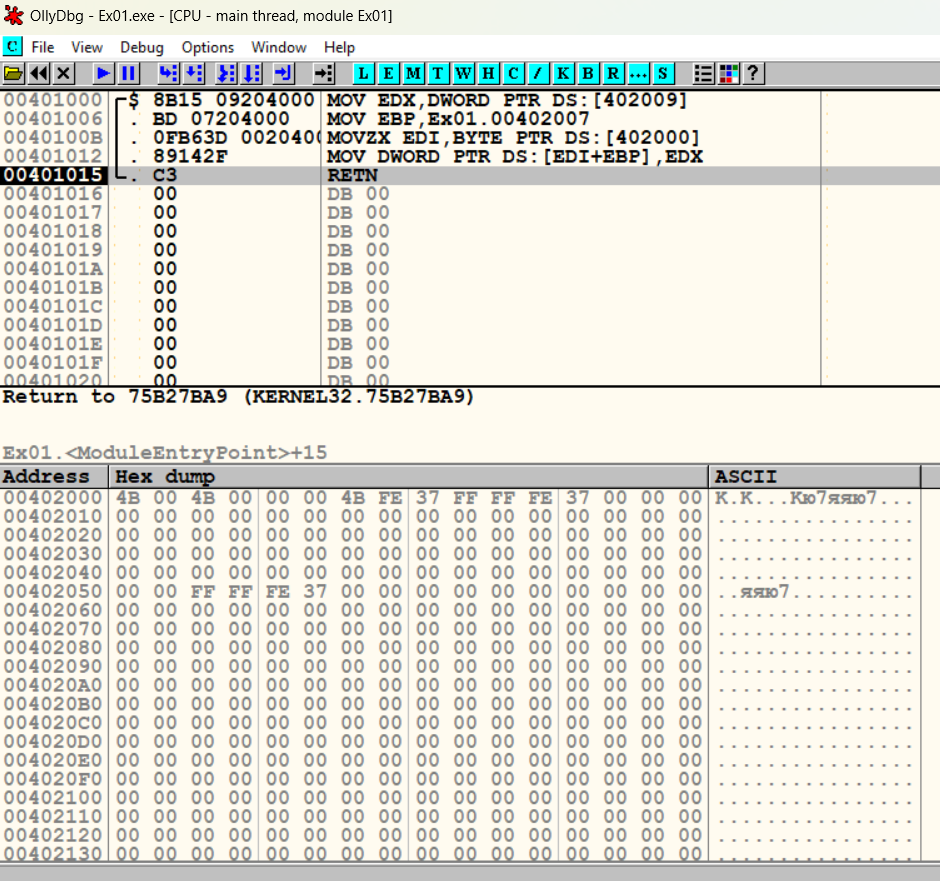


Рисунок 6 – дамп памяти после выполнения программы

Видно, что после выполнения программы по адресу 00402053 записалось значение I2.

*5. Сделать расчет времени выполнения программы.*

Для выполнения расчёта скорости выполнения программы в тактах процессоры воспользовался приложением А из пособия для лабораторных работ, однако там не была представлена характеристика в тактах команды MOVZX, её нашёл в интернете. Расчёт времени выполнения программы представлен на рисунке (Рисунок 7).

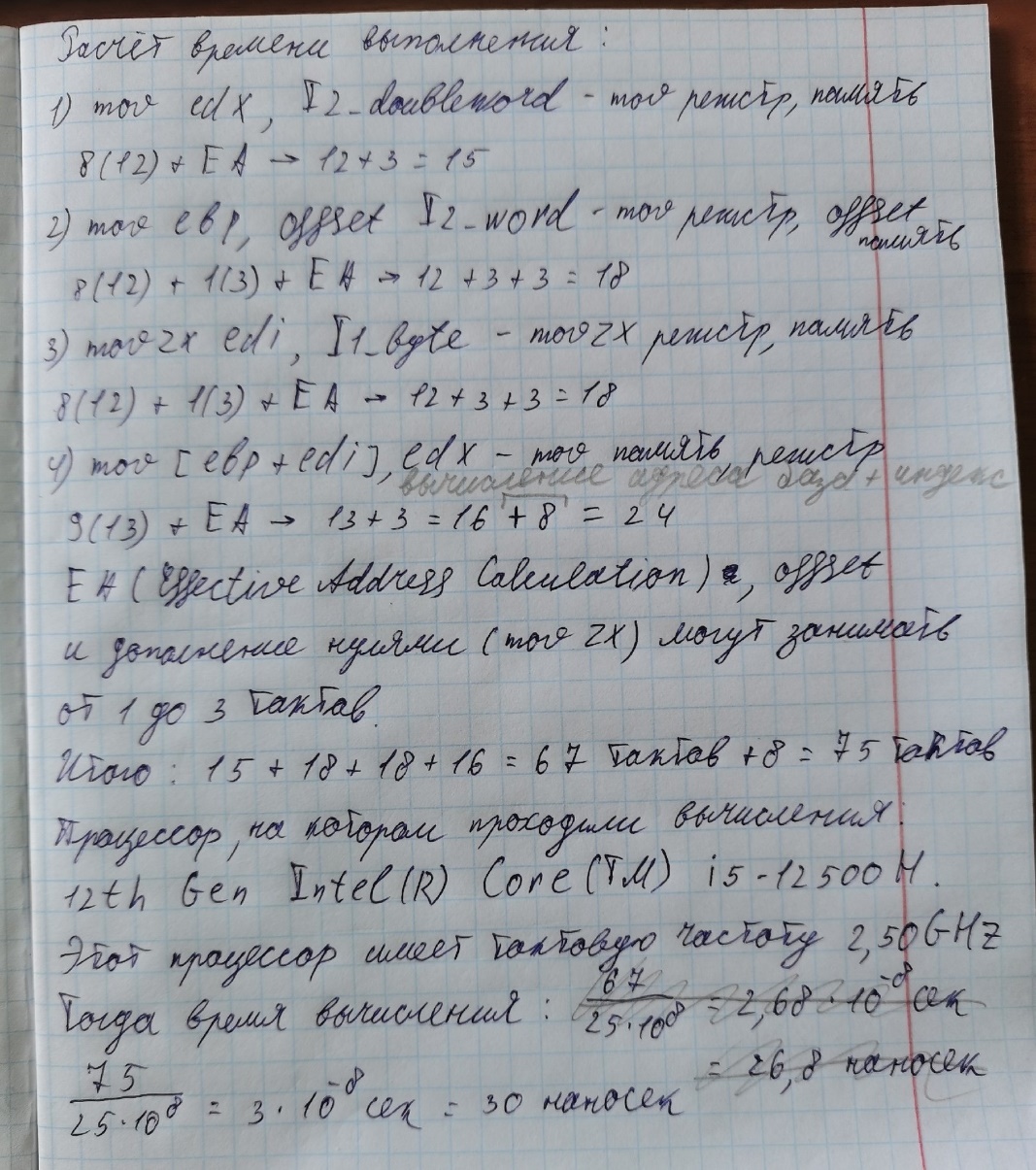


Рисунок 7 – расчёт времени выполнения программы

Кроме примерного времени дополнения нулями (MOVZX) также в интернете пришлось искать примерную скорость выполнения OFFSET и EA, так что итоговые значения могут колебаться (на рисунке представлены расчёты для самого худшего по длительности времени выполнения).

**ПРИЛОЖЕНИЕ А**

.486

.model flat

.data

I1\_byte DB 4Bh

I1\_word DW 4B00h

I1\_doubleword DD 4B000000h

I2\_word DW 37FEh

I2\_doubleword DD 37FEFFFFh

.code

\_start:

mov edx, I2\_doubleword

mov ebp, offset I2\_word

movzx edi, I1\_byte

mov [ebp+edi], edx

ret

end \_start