

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)**

Кафедра информационных технологий

ОТЧЕТ

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

Выполнил студент МО42.2 группы

Д.А. Вербицкий

Проверил доц. каф. ИТ

А.Н. Полетайкин

г. Краснодар
2025

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 и I2 (табл. 1) 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.

Таблица 1

№ вар.	I 1	I 2	R 1	Косвенная адресация
17	175	-1457	DX	[BP+DI]

Вариант 17

Ход работы

1. Для представления числа I_1 в формате DB, DW, DD, преобразуем его в двоичный формат:

$$175 = 10101111_2$$

Затем в шестнадцатиричный:

$$10101111_2 = AF_{16}$$

В формате DB число будет записываться как CB. В форматах DW и DD необходимо дополнить нулями до 16 и 32 бит, получаем соответственно 00AF и 0000 00AF

Для представления I_2 необходимо представить модуль числа в двоичном виде:

$$|-1457|_{10} = 1457_{10} = 10110110001_2$$

Дополним нулями до 16 и 32 бит:

$$0000\ 0101\ 1011\ 0001_2, 0000\ 0000\ 0000\ 0000\ 0000\ 0101\ 1011\ 0001_2$$

Инвертируем полученное число и прибавим к нему единицу:

$$1111\ 1010\ 0100\ 1110_2 + 1_2 = 1111\ 1010\ 0100\ 1111_2$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1010\ 0100\ 1110_2 + 1_2 =$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1010\ 0100\ 1111_2$$

Запишем полученный результат в 16-ричном виде:

$$1111\ 1010\ 0100\ 1111_2 = FA\ 4F_{16}$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1010\ 0100\ 1111_2 = FF\ FF\ FA\ 4F_{16}$$

Итоговое представление чисел I_1 и I_2 в памяти в форматах DB, DW, DD представлено в таблице 2

Таблица 2

Число	DB	DW	DD
175	AF	AF 00	AF 00 00 00
-1457	-	4F FA	4F FA FF FF

2. Запишем в секцию .data переменные в соответствующих форматах:

.data

I1_DB DB 175 ; AF

I1_DW DW 175 ; 00 AF

I1_DD DD 175 ; 00 00 00 AF

I2_DW DW -1457 ; FA 4F

I2_DD DD -1457 ; FF FF FA 4F

3. Была написана программа на языке ассемблера MASM, определяющая число I_1 в форматах DB, DW, DD, а число I_2 в форматах DW, DD.

Также была выполнена загрузка регистра R_1 числом I_2 из сегмента данных. Затем было записано в сегмент данных содержимое регистра R_1 со смещением на I_1 байт относительно метки I_2 , при этом число I_1 предварительно загружено в регистр DX.

.686

include /masm32/include/io.asm

.data

I1_DB DB 175

I1_DW DW 175

I1_DD DD 175

I2_DW DW -1457

I2_DD DD -1457

.code

start:

xor EDX, EDX

xor DX, DX

```
xor EBP, EBP
```

```
xor SI, SI
```

```
mov DX, I1_DW
```

```
mov EBP, offset I2_DW
```

```
mov EDI, I1_DD
```

```
mov [EBP + EDI], DX
```

```
exit
```

```
end start
```

4. Чтобы проверить результаты расчеты и пересылок в дампе памяти воспользуемся инструментами OllyDbg. Запустив программу в отладчике, находим адрес и значение переменных в памяти в различных форматах:

Address	Hex dump	ASCII
00403000	00 AF AF 00 AF 00 00 00	.II.I...
00403008	4F FA 4F FA FF FF 00 00	ОЪОъя..
00403010	00 00 00 00 00 00 00 00
00403018	00 00 00 00 00 00 00 00
00403020	00 00 00 00 00 00 00 00

Рисунок 1 – дамп памяти значений переменных

Registers (FPU)									
EAX	0019FFCC								
ECX	00401000	Task1.<ModuleEntryPoint>							
EDX	000000AF								
EBX	003B0000								
ESP	0019FF74								
EBP	00000000								
ESI	00400000	Task1.00400000							
EDI	00401000	Task1.<ModuleEntryPoint>							
EIP	00401011	Task1.00401011							
C 0	ES	002B	32bit	0	(FFFFFFFF)				
P 1	CS	0023	32bit	0	(FFFFFFFF)				
A 0	SS	002B	32bit	0	(FFFFFFFF)				
Z 1	DS	002B	32bit	0	(FFFFFFFF)				
S 0	FS	0053	32bit	3B3000	(FFF)				
T 0	GS	002B	32bit	0	(FFFFFFFF)				
D 0									
O 0	LastErr	ERROR	SUCCESS	(00000000)					

Рисунок 2 – регистр EDX после записи в него значения I_1_DW

Также находим результат пересылок, пролистав вниз. Получим, что смещение числа I_2 произошло ровно на $I_1=175$ байта

00403090	00	00	00	00	00	00	00	00	00
00403098	00	00	00	00	00	00	00	00	00
004030A0	00	00	00	00	00	00	00	00	00
004030A8	00	00	00	00	00	00	00	00	00
004030B0	00	00	00	00	00	00	00	AF	AFI

Рисунок 3 – дамп памяти пересылок

- Посчитаем количество тактов процессора на выполнение каждой команды:

start:

xor EDX, EDX ; 3

xor DX, DX ; 3

xor EBP, EBP ; 3

xor SI, SI ; 3

mov DX, I1_DW ; 12 + 5 = 17

```
mov EBP, offset I2_DW; 12 + 9 = 21
```

```
mov EDI, I1_DD ; 12 + 5 = 17
```

```
mov [EBP + EDI], DX ; 13 + 5 = 18
```

```
exit ; 0
```

```
end start
```

Выполнение программы требует 85 тактов процессора. При тактовой частоте процессора 2.5 ГГц, без учёта прочих факторов, время работы программы должно составлять

$$85 / 2.5 \cdot 10^9 = 3.4 \cdot 10^{-8} \text{ с} = 34 \text{ наносекунд.}$$

Вывод: в ходе выполнения лабораторной работы были рассмотрены операции загрузки и пересылки данных между регистрами и памятью, в том числе обращение к памяти с косвенной адресацией ([EBP + EDI]). Были изучены форматы представления целых чисел в памяти (DB, DW, DD) и реализованы соответствующие команды на языке ассемблера для выполнения необходимых операций с числами и адресами.