

Отчёт по лабораторной работе №6

Арифметические операция в NASM

Новикова Анастасия Андреевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Символьные и численные данные в NASM	7
3.2	Выполнение арифметических операций в NASM	10
3.2.1	Ответы на вопросы по программе	13
3.3	Выполнение заданий для самостоятельной работы	14
4	Выводы	17

Список иллюстраций

3.1	Создание директории	7
3.2	Создание файла	7
3.3	Создание копии файла	7
3.4	Редактирование файла	8
3.5	Запуск исполняемого файла	8
3.6	Редактирование файла	8
3.7	Запуск исполняемого файла	8
3.8	Создание файла	9
3.9	Редактирование файла	9
3.10	Запуск исполняемого файла	9
3.11	Редактирование файла	9
3.12	Запуск исполняемого файла	10
3.13	Редактирование файла	10
3.14	Запуск исполняемого файла	10
3.15	Создание файла	10
3.16	Редактирование файла	11
3.17	Запуск исполняемого файла	11
3.18	Изменение программы	11
3.19	Запуск исполняемого файла	12
3.20	Создание файла	12
3.21	Редактирование файла	12
3.22	Запуск исполняемого файла	12
3.23	Создание файла	14
3.24	Написание программы	14
3.25	Запуск исполняемого файла	14
3.26	Запуск исполняемого файла	15

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Символьные и численные данные в NASM

С помощью команды `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 3.1). Перехожу в созданный каталог с помощью команды `cd`.

```
aanovikova123@fedora:~/work/arch-pc$ mkdir ~/work/arch-pc/lab06
aanovikova123@fedora:~/work/arch-pc$ cd ~/work/arch-pc/lab06
aanovikova123@fedora:~/work/arch-pc/lab06$
```

Рис. 3.1: Создание директории

С помощью команды `touch` создаю файл `lab6-1.asm` (рис. 3.2).

```
aanovikova123@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ls
lab6-1.asm
aanovikova123@fedora:~/work/arch-pc/lab06$
```

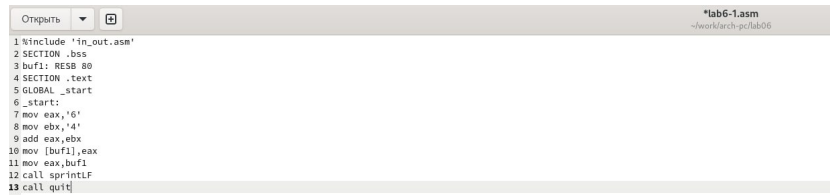
Рис. 3.2: Создание файла

Копирую в текущий каталог файл `in_out.asm` с помощью команды `cp`, т.к. он будет использоваться в других программах (рис. 3.3).

```
aanovikova123@fedora:~/work/arch-pc/lab06$ cp ~/work/arch-pc/lab05/in_out.asm in_out.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ █
```

Рис. 3.3: Создание копии файла

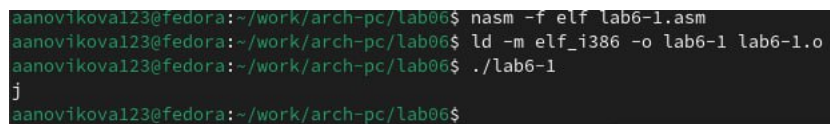
Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 3.4).



```
1 %include "in_out.asm"
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, 6
8 mov ebx, 4
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintf
13 call quit
```

Рис. 3.4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 3.5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



```
aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
aanovikova123@fedora:~/work/arch-pc/lab06$
```

Рис. 3.5: Запуск исполняемого файла

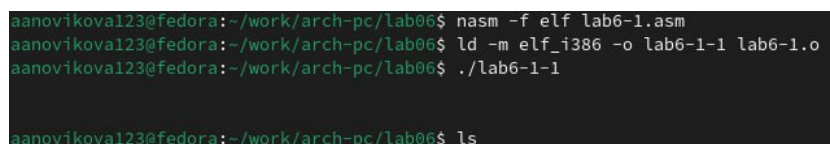
Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 3.6).



```
1 %include "in_out.asm"
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, 6
8 mov ebx, 4
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintf
13 call quit
```

Рис. 3.6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 3.7). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.



```
aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1-1 lab6-1.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-1-1
aanovikova123@fedora:~/work/arch-pc/lab06$ ls
```

Рис. 3.7: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью команды touch (рис. 3.8).

```
aanovikova123@fedora:~/work/arch-pc/lab06$ touch lab6-2.asm
aanovikova123@fedora:~/work/arch-pc/lab06$
```

Рис. 3.8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 3.9).



```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintlf
9 call quit
```

Рис. 3.9: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 3.10). Теперь выводится число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4” ($54 + 52 = 106$).

```
aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
aanovikova123@fedora:~/work/arch-pc/lab06$
```

Рис. 3.10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 3.11).



```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintlf
9 call quit
```

Рис. 3.11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 3.12). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```

aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2-1 lab6-2.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-2-1
10
aanovikova123@fedora:~/work/arch-pc/lab06$

```

Рис. 3.12: Запуск исполняемого файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 3.13).



```

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,0
6 mov ebx,4
7 add eax,ebx
8 call iprint
9 call quit

```

Рис. 3.13: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 3.14). Вывод значения не изменился, но курсор остался на той же строке. То есть функция `iprintLF` переводит курсор на следующую строку, а `iprint` не добавляет к выводу символ переноса строки.

```

aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2-2 lab6-2.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-2-2
10aanovikova123@fedora:~/work/arch-pc/lab06$

```

Рис. 3.14: Запуск исполняемого файла

3.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью команды `touch` (рис. 3.15).

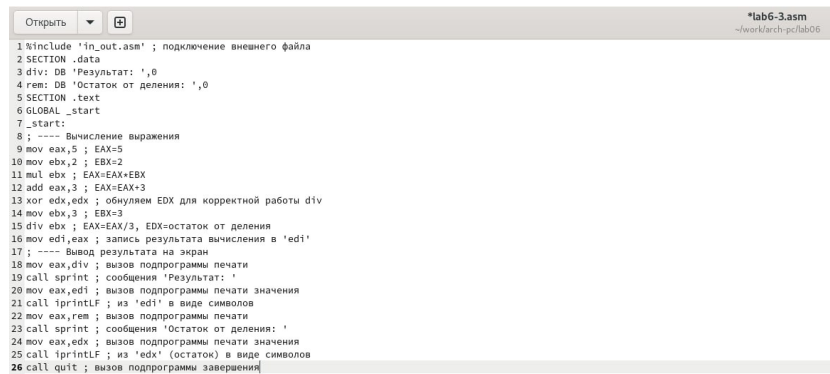
```

aanovikova123@fedora:~/work/arch-pc/lab06$ touch lab6-3.asm
aanovikova123@fedora:~/work/arch-pc/lab06$

```

Рис. 3.15: Создание файла

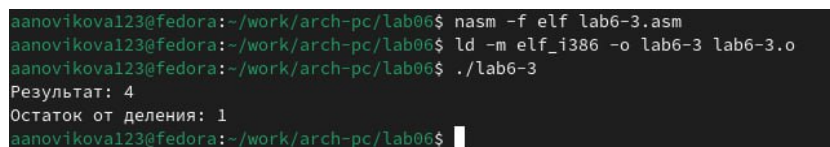
Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 3.16).



```
1 include "in_out.asm" ; подключение внешнего файла
2 section .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 section .text
6 global _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ; EAX=5
10 mov ebx,2 ; EBX=2
11 mul ebx ; EAX=EAX*EBX
12 add eax,3 ; EAX=EAX+3
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,3 ; EBX=3
15 div ebx ; EAX=EAX/3, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintf ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintf ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

Рис. 3.16: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 3.17).



```
aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
aanovikova123@fedora:~/work/arch-pc/lab06$
```

Рис. 3.17: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 3.18).



```
1 include "in_out.asm" ; подключение внешнего файла
2 section .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 section .text
6 global _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,4 ; EAX=4
10 mov ebx,6 ; EBX=6
11 mul ebx ; EAX=EAX*EBX
12 add eax,2 ; EAX=EAX+2
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,5 ; EBX=5
15 div ebx ; EAX=EAX/5, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintf ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintf ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

Рис. 3.18: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 3.19). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3-1 lab6-3.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-3-1
Результат: 5
Остаток от деления: 1
aanovikova123@fedora:~/work/arch-pc/lab06$

```

Рис. 3.19: Запуск исполняемого файла

Создаю файл variant.asm с помощью команды touch (рис. 3.20).

```

aanovikova123@fedora:~/work/arch-pc/lab06$ touch variant.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ gedit variant.asm

```

Рис. 3.20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 3.21).



```

1 %include "in_out.asm"
2 SECTION .data
3 msg: DB "Введите № студенческого билета: ",0
4 rem: DB "Ваш вариант: ",0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call read
15 mov eax, x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, 'eax=x'
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintf
25 call quit

```

Рис. 3.21: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 3.22). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 20.

```

aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246839
Ваш вариант: 20
aanovikova123@fedora:~/work/arch-pc/lab06$

```

Рис. 3.22: Запуск исполняемого файла

Я проверила аналитически, действительно, $1132246839/20 + 1 = 20$. Остаток от деления 1132246839 на 20 равен 19, $19 + 1 = 20$.

3.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax,rem
```

```
call sprint
```

2. Инструкция mov esx, x используется, чтобы положить адрес вводимой строки x в регистр esx. mov edx, 80 - запись в регистр edx длины вводимой строки. call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.
3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ASCII-код символа в целое число и записывает результат в регистр eax.
4. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div
```

```
mov ebx,20 ; ebx = 20
```

```
div ebx ; eax = eax/20, edx - остаток от деления
```

```
inc edx ; edx = edx + 1
```

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx увеличивает значение регистра edx на 1
7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx
```

```
call iprintLF
```

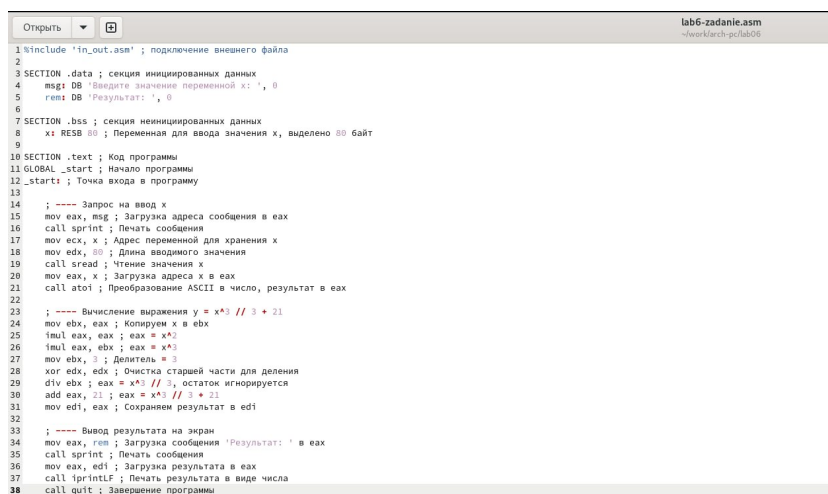
3.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-zadanie.asm с помощью команды touch (рис. 3.23).

```
aanovikova123@fedora:~/work/arch-pc/lab06$ touch lab6-zadanie.asm
```

Рис. 3.23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $x^3 \cdot \frac{1}{3} + 21$ (рис. 3.24). Это выражение было под вариантом 20.



```
1 %include "in_out.asm" ; подключение внешнего файла
2
3 SECTION .data ; секция инициализированных данных
4     msgt DB "Введите значение переменной x: ", 0
5     reslt DB "Результат: ", 0
6
7 SECTION .bss ; секция неинициализированных данных
8     x: RESB 80 ; Переменная для ввода значения x, выделено 80 байт
9
10 SECTION .text ; Код программы
11 GLOBAL _start ; Начало программы
12 _start: ; Точка входа в программу
13
14     ; ---- Запрос на ввод x
15     mov eax, msgt ; Загрузка адреса сообщения в eax
16     call sprint ; Печать сообщения
17     mov ecx, x ; Адрес переменной для хранения x
18     mov edx, 80 ; Длина вводимого значения
19     call sread ; Чтение значения x
20     mov eax, x ; Загрузка адреса x в eax
21     call atoi ; Преобразование ASCII в число, результат в eax
22
23     ; ---- Вычисление выражения y = x^3 // 3 + 21
24     mov ebx, eax ; Копируем x в ebx
25     imul ebx, ebx ; eax = x^2
26     imul ebx, ebx ; eax = x^3
27     mov ebx, 3 ; Делитель = 3
28     xor edx, edx ; Очистка старшей части для деления
29     div ebx ; eax = x^3 // 3, остаток игнорируется
30     add eax, 21 ; eax = x^3 // 3 + 21
31     mov edi, eax ; Сохраняем результат в edi
32
33     ; ---- Вывод результата на экран
34     mov eax, reslt ; Загрузка сообщения "Результат: " в eax
35     call sprint ; Печать сообщения
36     mov eax, edi ; Загрузка результата в eax
37     call fprintf ; Печать результата в виде числа
38     call quit ; Завершение программы
```

Рис. 3.24: Написание программы

Создаю и запускаю исполняемый файл (рис. 3.25). При вводе значения 1, вывод - 21. Так как $1^3 = 1$. Остаток от деления на 3 равен 0. Значит, $0 + 21 = 21$. Программа отработала верно.

```
aanovikova123@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-zadanie.asm
aanovikova123@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-zadanie lab6-zadanie.o
aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-zadanie
Введите значение переменной x: 1
Результат: 21
```

Рис. 3.25: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 3.26). Программа отработала верно. $3^3 = 27$. $27/3 = 9$. $9 + 21 = 30$.

```

aanovikova123@fedora:~/work/arch-pc/lab06$ ./lab6-zadanie
Введите значение переменной x: 3
Результат: 30

```

Рис. 3.26: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения $x^3 * 1/3 + 21$.

```

#include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; секция инициализированных данных
    msg: DB 'Введите значение переменной x: ', 0
    rem: DB 'Результат: ', 0

SECTION .bss ; секция неинициализированных данных
    x: RESB 80 ; Переменная для ввода значения x, выделено 80 байт

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

    ; ---- Запрос на ввод x
    mov eax, msg ; Загрузка адреса сообщения в eax
    call sprint ; Печать сообщения
    mov ecx, x ; Адрес переменной для хранения x
    mov edx, 80 ; Длина вводимого значения
    call sread ; Чтение значения x
    mov eax, x ; Загрузка адреса x в eax
    call atoi ; Преобразование ASCII в число, результат в eax

    ; ---- Вычисление выражения  $y = x^3 // 3 + 21$ 

```

```

mov ebx, eax ; Копируем x в ebx
imul eax, eax ;  $eax = x^2$ 
imul eax, ebx ;  $eax = x^3$ 
mov ebx, 3 ; Делитель = 3
xor edx, edx ; Очистка старшей части для деления
div ebx ;  $eax = x^3 // 3$ , остаток игнорируется
add eax, 21 ;  $eax = x^3 // 3 + 21$ 
mov edi, eax ; Сохраняем результат в edi

; ---- Вывод результата на экран
mov eax, rem ; Загрузка сообщения 'Результат: ' в eax
call sprint ; Печать сообщения
mov eax, edi ; Загрузка результата в eax
call iprintLF ; Печать результата в виде числа
call quit ; Завершение программы

```


4 Выводы

При выполнении данной лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.