

Лабораторная работа №6

Дисциплина: Архитектура компьютера

Малюга Валерия Васильевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	12
4.2.1	Ответы на вопросы по программе	15
4.3	Выполнение заданий для самостоятельной работы	16
5	Выводы	19
6	Список литературы	20

Список иллюстраций

4.1	Создание файла lab6-1.asm и копирование in_out.asm	8
4.2	Редактирование файла	9
4.3	Создание исполняемого файла и его запуск	9
4.4	Редактирование файла	10
4.5	Создание исполняемого файла и его запуск	10
4.6	Создание и редактирование файла lab6-2.asm	10
4.7	Создание исполняемого файла и его запуск	11
4.8	Редактирование программы	11
4.9	Создание исполняемого файла и его запуск	11
4.10	Редактирование файла и его запуск	12
4.11	Создание файла	13
4.12	Запуск исполняемого файла	13
4.13	Изменение программы и запуск исполняемого файла	14
4.14	Создание, редактирование и запуск файла	15
4.15	Запуск программы для выполнения задания для самостоятельной работы	17

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 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом.

Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти

коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы.

Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций.

Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

4 Выполнение лабораторной работы

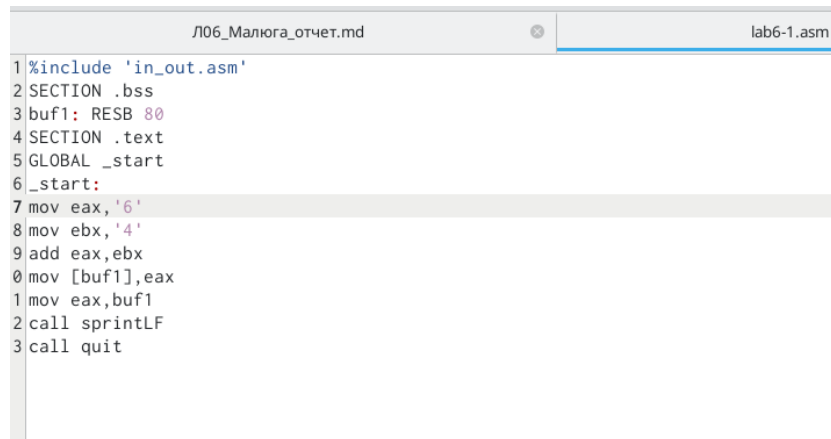
4.1 Символьные и численные данные в NASM

С помощью утилиты `mkdir` создала каталог для программ лабораторной работы № 6, перешла в него и создала файл `lab6-1.asm`. Скопировала в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах (рис. 4.1).

```
vvmalyuga@edk6n57 ~ $ mkdir ~/work/arch-pc/lab06
vvmalyuga@edk6n57 ~ $ cd ~/work/arch-pc/lab06
vvmalyuga@edk6n57 ~/work/arch-pc/lab06 $ touch lab6-1.asm
vvmalyuga@edk6n57 ~/work/arch-pc/lab06 $ ls
lab6-1.asm
vvmalyuga@edk6n57 ~/work/arch-pc/lab06 $ cp ~/"Загрузки"/in_out.asm in_out.asm
vvmalyuga@edk6n57 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm
vvmalyuga@edk6n57 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание файла `lab6-1.asm` и копирование `in_out.asm`

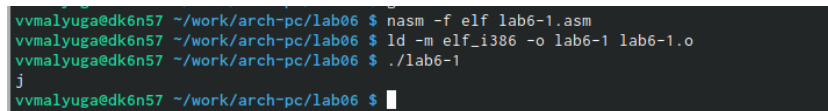
Ввела в файл `lab6-1.asm` текст программы из листинга 6.1 (рис. 4.2).



```
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
0 mov [buf1], eax
1 mov eax, buf1
2 call sprintLF
3 call quit
```

Рис. 4.2: Редактирование файла

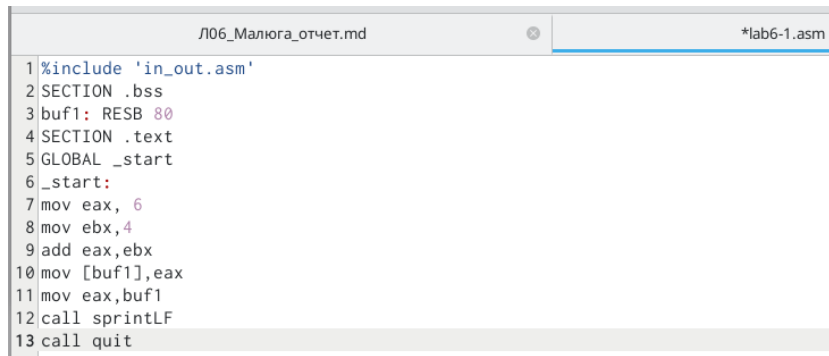
Создала исполняемый файл и запустила его (рис. 4.3). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



```
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ./lab6-1
j
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $
```

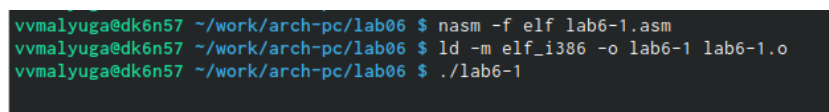
Рис. 4.3: Создание исполняемого файла и его запуск

Исправила текст программы: поменяла символы “6” и “4” на цифры 6 и 4 (рис. 4.4). Создала исполняемый файл и запустила его (рис. 4.5). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.



```
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 sprintLF
13 call quit
```

Рис. 4.4: Редактирование файла



```
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 4.5: Создание исполняемого файла и его запуск

Создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы из листинга 6.2 (рис. 4.6).



```
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. 4.6: Создание и редактирование файла lab6-2.asm

Создала исполняемый файл и запустила его (рис. 4.7). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

```

vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ./lab6-2
106
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $

```

Рис. 4.7: Создание исполняемого файла и его запуск

Заменяла в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.8).



```

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. 4.8: Редактирование программы

Создала и запустила новый исполняемый файл (рис. 4.9). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```

vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ./lab6-2
10
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $

```

Рис. 4.9: Создание исполняемого файла и его запуск

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.10). Создала и запустила новый исполняемый файл (рис. 4.10). Вывод не изменился потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

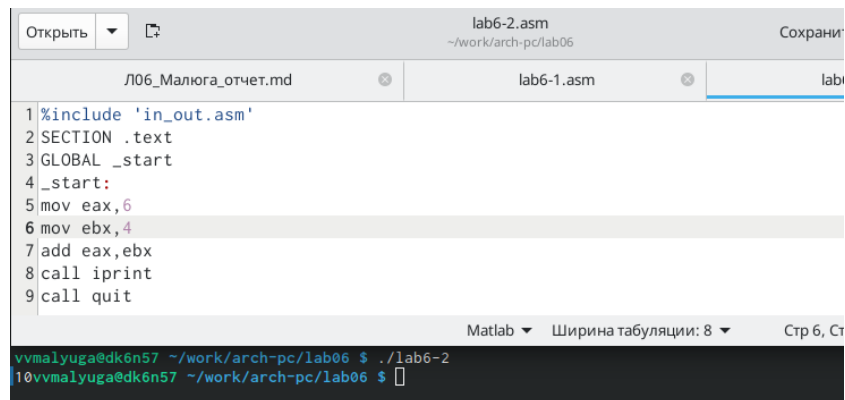


Рис. 4.10: Редактирование файла и его запуск

4.2 Выполнение арифметических операций в NASM

Создала файл lab6-3.asm с помощью утилиты touch (рис. 4.11). Ввела в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.11).

```
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ touch lab6-3.asm
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
```

```
1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4
5 %include 'in_out.asm' ; подключение внешнего файла
6 SECTION .data
7 div: DB 'Результат: ',0
8 rem: DB 'Остаток от деления: ',0
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13
14 ; ---- Вычисление выражения
15 mov eax,5 ; EAX=5
16 mov ebx,2 ; EBX=2
17 mul ebx ; EAX=EAX*EBX
18 add eax,3 ; EAX=EAX+3
19 xor edx,edx ; обнуляем EDX для корректной работы div
20 mov ebx,3 ; EBX=3
21 div ebx ; EAX=EAX/3, EDX=остаток от деления
22 mov edi,eax ; запись результата вычисления в 'edi'
23
24 ; ---- Вывод результата на экран
25 mov eax,div ; вызов подпрограммы печати
26 call sprint ; сообщения 'Результат: '
27 mov eax,edi ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edi' в виде символов
29
30 mov eax,rem ; вызов подпрограммы печати
31 call sprint ; сообщения 'Остаток от деления: '
32 mov eax,edx ; вызов подпрограммы печати значения
33 call iprintLF ; из 'edx' (остаток) в виде символов
34
35 call quit ; вызов подпрограммы завершения
```

Рис. 4.11: Создание файла

Создала исполняемый файл и запустила его (рис. [4.12]).

```
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $
```

Рис. 4.12: Запуск исполняемого файла

Изменила программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.13). Создала и запустила новый исполняемый файл (рис. 4.13). Программа отработала верно.

The image shows a screenshot of a code editor window titled 'lab6-3.asm' with the path '~/.work/arch-pc/lab06'. Below the editor is a terminal window showing the execution of the program. The assembly code in the editor calculates the product of 4 and 6, adds 2, and then divides by 5, printing the result and remainder.

```
1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4
5 %include 'in_out.asm' ; подключение внешнего файла
6 SECTION .data
7 div: DB 'Результат: ',0
8 rem: DB 'Остаток от деления: ',0
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13
14 ; ---- Вычисление выражения
15 mov eax,4 ; EAX=4
16 mov ebx,6 ; EBX=6
17 mul ebx ; EAX=EAX*EBX
18 add eax,2 ; EAX=EAX+2
19 xor edx,edx ; обнуляем EDX для корректной работы div
20 mov ebx,5 ; EBX=5
21 div ebx ; EAX=EAX/5, EDX=остаток от деления
22 mov edi,eax ; запись результата вычисления в 'edi'
23
24 ; ---- Вывод результата на экран
25 mov eax,div ; вызов подпрограммы печати
26 call sprint ; сообщения 'Результат: '
27 mov eax,edi ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edi' в виде символов
29
30 mov eax,rem ; вызов подпрограммы печати
31 call sprint ; сообщения 'Остаток от деления: '
32 mov eax,edx ; вызов подпрограммы печати значения
33 call iprintLF ; из 'edx' (остаток) в виде символов
34
35 call quit ; вызов подпрограммы завершения
```

Текст ▾ Ширина

```
vvmalyuga@dk6n57 ~/.work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
vvmalyuga@dk6n57 ~/.work/arch-pc/lab06 $
```

Рис. 4.13: Изменение программы и запуск исполняемого файла

Создала файл `variant.asm` с помощью утилиты `touch`. Ввела в файл текст из листинга 6.4. Создала исполняемый файл и запустила его. Ввела номер своего сту-

денческого билета и получила номер своего варианта – 11 (рис. 4.14). Проверила правильность выполнения, вычислив номер варианта аналитически.

```

vmalyuga@dk6n57 ~/work/arch-pc/lab06 $ touch variant.asm
vmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o lab6-3 lab6-3.asm lab6-3.o variant.asm
vmalyuga@dk6n57 ~/work/arch-pc/lab06 $ gedit variant.asm
vmalyuga@dk6n57 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
vmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
vmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236050
Ваш вариант: 11
vmalyuga@dk6n57 ~/work/arch-pc/lab06 $

```

```

1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg: DB 'Введите No студенческого билета: ',0
7 rem: DB 'Ваш вариант: ',0
8 SECTION .bss
9 x: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprintf
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x ; вызов подпрограммы преобразования
19 call atoi ; ASCII кода в число, 'eax=x'
20
21 xor edx,edx
22 mov ebx,20
23 div ebx
24 inc edx
25
26 mov eax,rem
27 call sprint
28 mov eax,edx
29 call iprintf
30
31 call quit

```

Рис. 4.14: Создание, редактирование и запуск файла

4.2.1 Ответы на вопросы по программе

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

```
mov eax, rem
```

```
call sprint
```

2. Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx`

`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
```

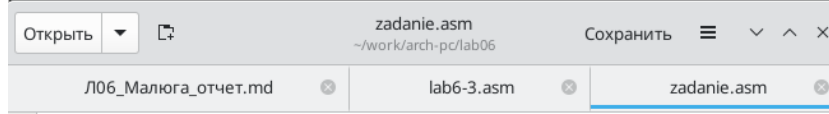
4.3 Выполнение заданий для самостоятельной работы

Создала файл `zadanie.asm`. Открыла его для редактирования, ввела в него текст программы для вычисления значения выражения $10 \cdot (x + 1) - 10$ (вариант 11). Создала и запустила исполняемый файл. При вводе значения 1 на входе вывод программы = 10. При вводе значения 7 на входе вывод программы = 70. Значит, программа работает верно. (рис. 4.15).


```

vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ./zadanie
Введите значение переменной x: 1
Результат:
10
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ ./zadanie
Введите значение переменной x: 7
Результат:
70
vvmalyuga@dk6n57 ~/work/arch-pc/lab06 $ 

```



```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data ; секция иницированных данных
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5
6 SECTION .bss ; секция не иницированных данных
7 x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры,
   выделенный размер - 80 байт
8
9 SECTION .text ; Код программы
10 GLOBAL _start ; Начало программы
11 _start: ; Точка входа в программу
12
13 ; ---- Вычисление выражения
14 mov eax, msg ; запись адреса выводимого сообщения в eax
15 call sprint ; вызов подпрограммы печати сообщения
16 mov ecx, x ; запись адреса переменной в ecx
17 mov edx, 80 ; запись длины вводимого значения в edx
18 call sread ; вызов подпрограммы ввода сообщения
19
20 mov eax,x ; вызов подпрограммы преобразования
21 call atoi ; ASCII кода в число, 'eax=x'
22 add eax,1; eax = eax+1 = x + 1
23 mov ebx,10 ; запись значения 10 в регистр ebx
24 mul ebx; EAX=EAX*EBX = (x+1)*10
25 add eax,-10; eax = eax-10 = (x+1)*10-10
26 mov edi,eax ; запись результата вычисления в 'edi'
27
28 ; ---- Вывод результата на экран
29 mov eax,rem ; вызов подпрограммы печати
30 call sprintf ; сообщения 'Результат: '
31 mov eax,edi ; вызов подпрограммы печати значения
32 call iprintLF ; из 'edi' в виде символов
33 call quit ; вызов подпрограммы завершения

```

Рис. 4.15: Запуск программы для выполнения задания для самостоятельной работы

Программа для вычисления значения выражения $10 * (x + 1) - 10$.

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция иницированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0

```

SECTION .bss ; секция не инициированных данных

x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный ра

SECTION .text ; Код программы

GLOBAL _start ; Начало программы

_start: ; Точка входа в программу

; ---- Вычисление выражения

mov eax, msg ; запись адреса выводимого сообщения в eax

call sprint ; вызов подпрограммы печати сообщения

mov ecx, x ; запись адреса переменной в ecx

mov edx, 80 ; запись длины вводимого значения в edx

call sread ; вызов подпрограммы ввода сообщения

mov eax,x ; вызов подпрограммы преобразования

call atoi ; ASCII кода в число, `eax=x`

add eax,1; $eax = eax + 1 = x + 1$

mov ebx,10 ; запись значения 10 в регистр ebx

mul ebx; $EAX = EAX * EBX = (x+1) * 10$

add eax,-10; $eax = eax - 10 = (x+1) * 10 - 10$

mov edi,eax ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран

mov eax,rem ; вызов подпрограммы печати

call sprintLF ; сообщения 'Результат: '

mov eax,edi ; вызов подпрограммы печати значения

call iprintLF ; из 'edi' в виде символов

call quit ; вызов подпрограммы завершения

5 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

6 Список литературы

1. Лабораторная работа №7