

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Гончарь Анастасия Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	13
4.2.1	Ответы на вопросы по программе	17
4.3	Выполнение заданий для самостоятельной работы	18
4.3.1	Листинг программы для вычисления значения выражения ($x+5$) ² -3	19
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Создание директории	8
4.2	Создание файла	8
4.3	Создание копии файла	8
4.4	Редактирование файла	9
4.5	Запуск исполняемого файла	9
4.6	Редактирование файла	10
4.7	Запуск исполняемого файла	10
4.8	Создание файла	10
4.9	Редактирование файла	11
4.10	Запуск исполняемого файла	11
4.11	Редактирование файла	12
4.12	Запуск исполняемого файла	12
4.13	Редактирование файла	13
4.14	Запуск исполняемого файла	13
4.15	Создание файла	13
4.16	Редактирование файла	14
4.17	Запуск исполняемого файла	14
4.18	Изменение программы	15
4.19	Запуск исполняемого файла	15
4.20	Создание файла	15
4.21	Редактирование файла	16
4.22	Запуск исполняемого файла	16
4.23	Создание файла	18
4.24	Написание программы	18
4.25	Запуск исполняемого файла	19

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . . .	7
-----	---	---

1 Цель работы

Целью данной лабораторной работы является 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

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Сначала с помощью `mkdir` создаю директорию, в которой буду создавать файлы с программами (рис. 4.1). Перехожу в созданный каталог с помощью утилиты `cd`.

```
aagoncharj@dk3n33 ~ $ mkdir ~/work/arch-pc/lab06  
aagoncharj@dk3n33 ~ $ cd ~/work/arch-pc/lab06
```

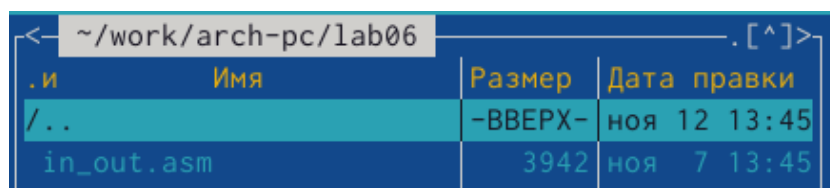
Рис. 4.1: Создание директории

С помощью `touch` создаю файл `lab6-1.asm` (рис. 4.2).

```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ touch lab6-1.asm  
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.2: Создание файла

Копирую в текущий каталог файл `in_out.asm`, т.к. он будет использоваться в других программах (рис. 4.3).



.и	Имя	Размер	Дата правки
/..		-ВВЕРХ-	ноя 12 13:45
	in_out.asm	3942	ноя 7 13:45

Рис. 4.3: Создание копии файла

Открываю созданный файл `lab6-1.asm` и вставляю в него программу вывода значения регистра `eax` (рис. 4.4).

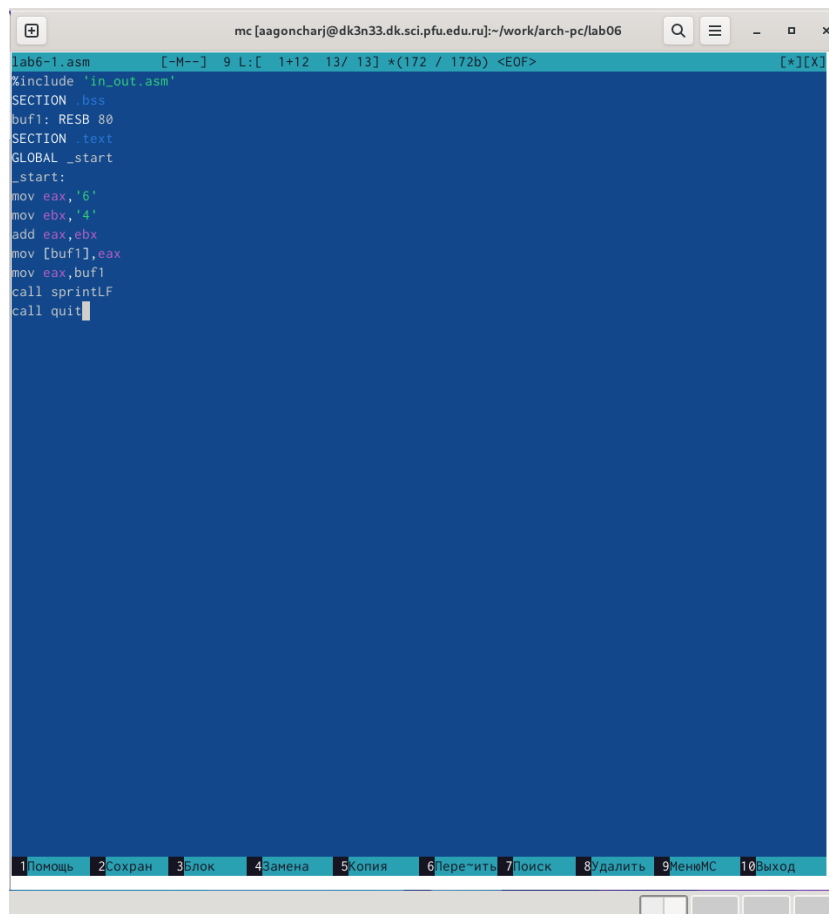


Рис. 4.4: Редактирование файла

Создаю исполняемый файл программы и запускаю его. Вывод программы - символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6. (рис. 4.5)

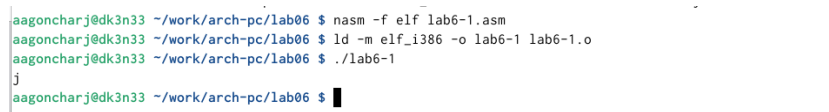


Рис. 4.5: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.6).

```
lab6-1.asm      [-M--]  9 L:[  1+ 7  8.
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1
call sprintf
call quit
```

Рис. 4.6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его. (рис. 4.7)

```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-1

aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.7: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью touch (рис. 4.8).

```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.8: Создание файла

Ввожу в файл другой текст программы для вывода значения регистра eax (рис. 4.9).

```
lab6-2.asm      [-M--]  9 L:[  1+ 8   9/  9] *(117 / 117b) <EOF>
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 4.9: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.10). Теперь выводится число 106.

```
aagoncharj@edk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aagoncharj@edk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aagoncharj@edk3n33 ~/work/arch-pc/lab06 $ ./lab6-2
106
aagoncharj@edk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.11).

```
lab6-2.asm      [-M--]  9 L:[ 1+ 4  5/ 9] *(
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
call iprintLF
call quit
```

Рис. 4.11: Редактирование файла

Создаю и запускаю новый исполняемый файл. Теперь выводится 10, так как программа складывает не соответствующие символам коды в системе ASCII, а сами числа. (рис. 4.12)

```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-2
10
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ █
```

Рис. 4.12: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.13).

```
lab6-2.asm [-M--] 11 L:[ 1+ 7 8/ 9] *(10
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 4.13: Редактирование файла

Создаю и запускаю новый исполняемый файл. 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. 4.14)

```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-2
10aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.14: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл `lab7-3.asm` с помощью утилиты `touch` (рис. 4.15).

```
10aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ 1
bash: 1: команда не найдена
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.asm.save lab6-2.o lab6-3.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.16).

```

lab6-3.asm  [-M--] 32 L: [ 1+ 6 7/ 29] *(299 /1365b) 0010 0x00A  [*][X]
; Программа вычисления выражения
;-----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintf ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintf ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.16: Редактирование файла

Создаю исполняемый файл и запускаю его. (рис. 4.17)

```

aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $

```

Рис. 4.17: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.18).

```
mc [aagoncharj@dk3n33.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab06
lab6-3.asm      [-M--] 19 L:[ 1+17 18/ 29] *(580 /1365b) 0044 0x02C
;-----
; Программа вычисления выражения
;-----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5 EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.18: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 4.19).

```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

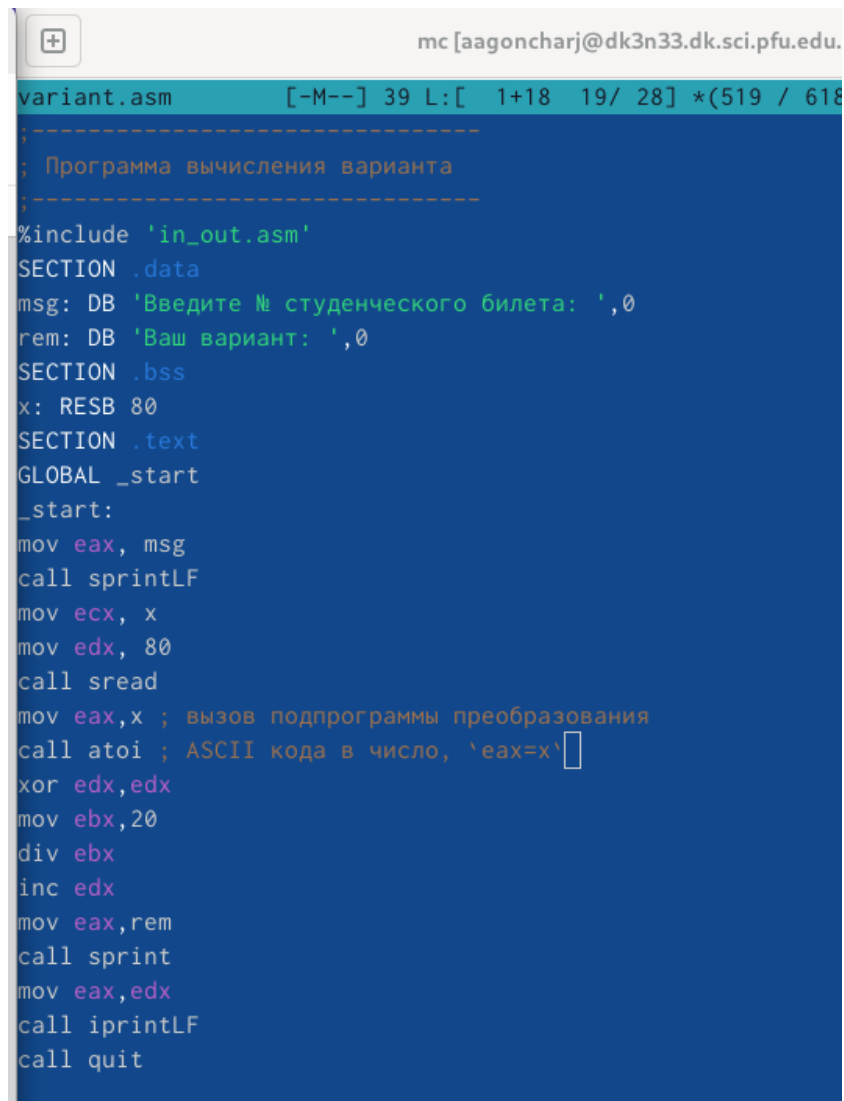
Рис. 4.19: Запуск исполняемого файла

Создаю файл variant.asm с помощью touch (рис. 4.20).

```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.asm.save  lab6-3      lab6-3.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-2.o         lab6-3.asm  variant.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.20: Создание файла

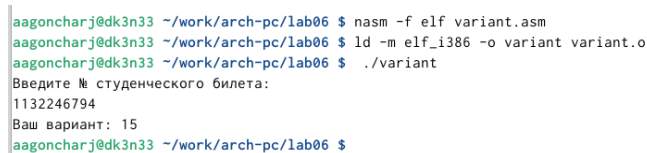
Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.21).



```
variant.asm [-M--] 39 L:[ 1+18 19/ 28] *(519 / 618
;-----
; Программа вычисления варианта
;-----
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintLF
call quit
```

Рис. 4.21: Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 15. (рис. 4.22)



```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132246794
Ваш вариант: 15
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.22: Запуск исполняемого 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

Создаю файл lab6-4.asm с помощью touch (рис. 4.23).

```
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-4.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1.asm lab6-2 lab6-2.asm.save lab6-3 lab6-3.o variant variant.o
lab6-1 lab6-1.o lab6-2.asm lab6-2.o lab6-3.asm lab6-4.asm variant.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 4.23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(x+5)^2-3$ из варианта 15 (рис. 4.24).

```
lab6-4.asm [----] 13 L: [ 1+25 26/ 28] *(1653/1765b) 0032 0x020 [*][X]
%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 иницированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт
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, 5; eax = eax + 5
mul eax; EAX=EAX*EAX = (x+5)*(x+5)
add eax, -3; eax = eax - 3 = (x+5)*(x+5) - 3
mov edi, eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprintf ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.24: Написание программы

Создаю и запускаю исполняемый файл. При вводе значения 1 выводится число 33, при вводе значения 5 - 97, что является верным. (рис. 4.25)

```

aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 1
Результат: 33
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 5
Результат: 97
aagoncharj@dk3n33 ~/work/arch-pc/lab06 $

```

Рис. 4.25: Запуск исполняемого файла

4.3.1 Листинг программы для вычисления значения выражения

$(x+5)^2-3$

```

#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 не инициализированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный >
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,5; eax = eax+5 = x + 5
mul eax; EAX=EAX*EAX = (x+5)*(x+5)
add eax,-3; eax = eax-3 = (x+5)*(x+5)-3

```

```
mov edi,eax ; запись результата вычисления в 'edi'  
; ---- Вывод результата на экран  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; 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.

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.