

# **Отчёта по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Аджабханян Овик

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Символьные и численные данные в NASM . . . . .	9
4.2	Выполнение арифметических операций в NASM . . . . .	14
4.2.1	Ответы на вопросы по программе . . . . .	17
4.3	Выполнение заданий для самостоятельной работы . . . . .	18
<b>5</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

# Список иллюстраций

4.1	Создание директории . . . . .	9
4.2	Редактирование файла . . . . .	10
4.3	Запуск исполняемого файла . . . . .	10
4.4	Редактирование файла . . . . .	11
4.5	Запуск исполняемого файла . . . . .	11
4.6	Создание файла . . . . .	11
4.7	Редактирование файла . . . . .	12
4.8	Запуск исполняемого файла . . . . .	12
4.9	Редактирование файла . . . . .	13
4.10	Запуск исполняемого файла . . . . .	13
4.11	Редактирование файла . . . . .	14
4.12	Запуск исполняемого файла . . . . .	14
4.13	Создание файла . . . . .	14
4.14	Редактирование файла . . . . .	15
4.15	Запуск исполняемого файла . . . . .	15
4.16	Изменение программы . . . . .	16
4.17	Запуск исполняемого файла . . . . .	16
4.18	Создание файла . . . . .	16
4.19	Редактирование файла . . . . .	17
4.20	Запуск исполняемого файла . . . . .	17
4.21	Создание файла . . . . .	18
4.22	Написание программы . . . . .	19
4.23	Запуск исполняемого файла . . . . .	19

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной 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

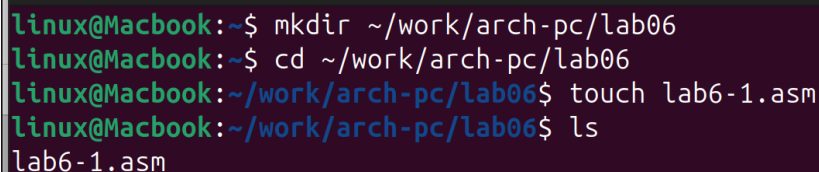
результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно



## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 4.1). Перехожу в созданный каталог с помощью утилиты `cd`, и с помощью утилиты `touch` создаю файл `lab6-1.asm`

A terminal window showing the following commands and their output:

```
linux@Macbook:~$ mkdir ~/work/arch-pc/lab06
linux@Macbook:~$ cd ~/work/arch-pc/lab06
linux@Macbook:~/work/arch-pc/lab06$ touch lab6-1.asm
linux@Macbook:~/work/arch-pc/lab06$ ls
lab6-1.asm
```

Рис. 4.1: Создание директории

Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.2).

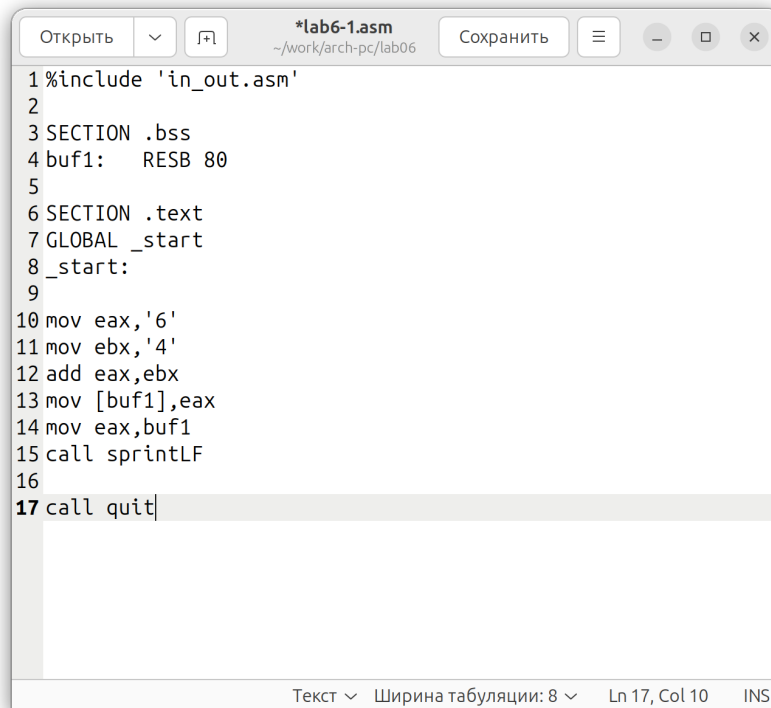


Рис. 4.2: Редактирование файла

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

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

Рис. 4.3: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.4).

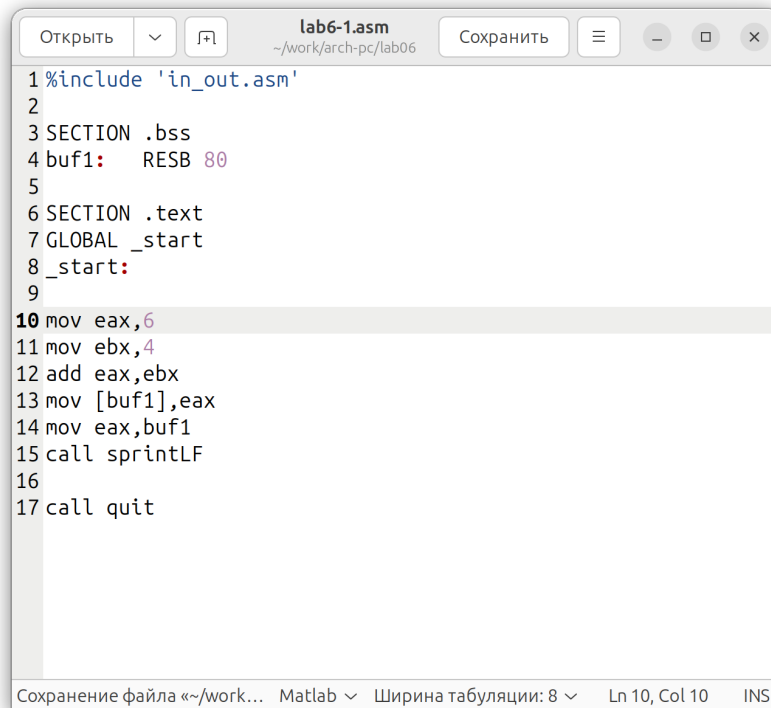


Рис. 4.4: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.5). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```

linux@Macbook:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
linux@Macbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
linux@Macbook:~/work/arch-pc/lab06$ ./lab6-1

```

Рис. 4.5: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 4.6).

```

linux@Macbook:~/work/arch-pc/lab06$ touch lab6-2.asm
linux@Macbook:~/work/arch-pc/lab06$

```

Рис. 4.6: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 4.7).

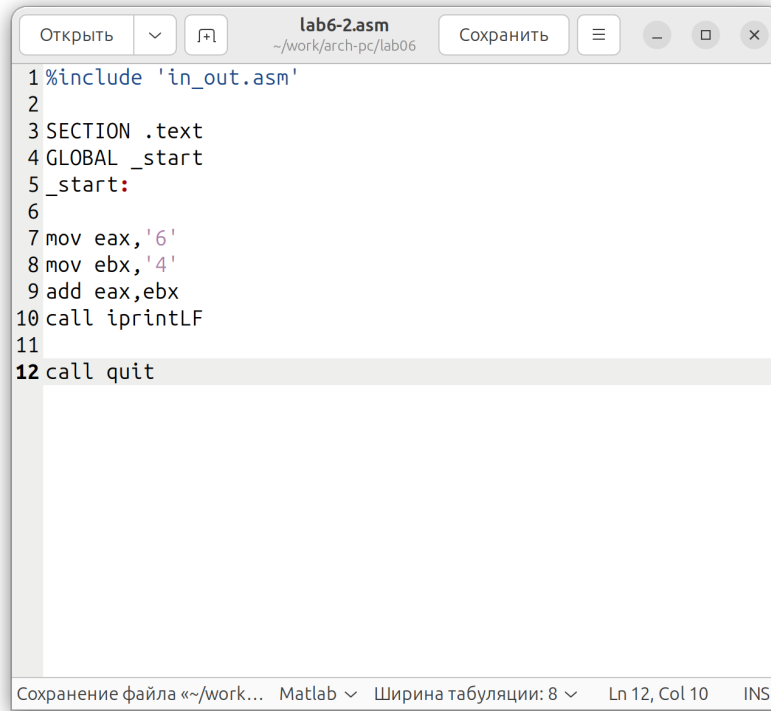


Рис. 4.7: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.8). 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”.

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

Рис. 4.8: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.9).

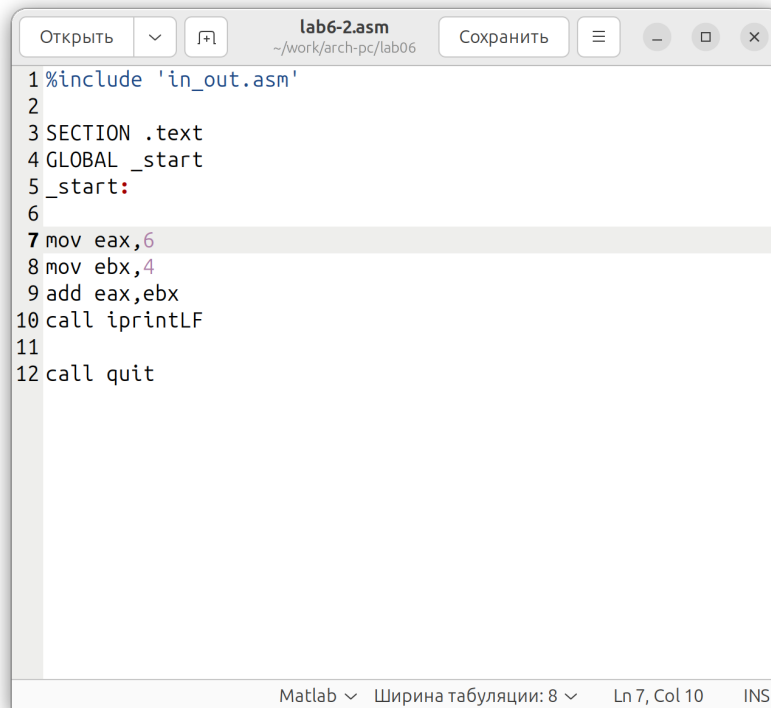


Рис. 4.9: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.10).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

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

Рис. 4.10: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.11).

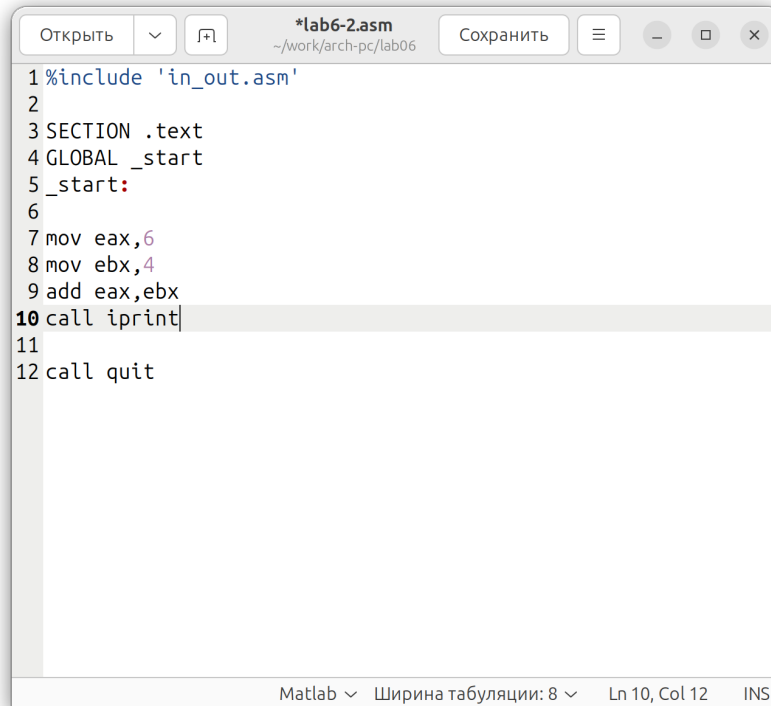


Рис. 4.11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.12). 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`.

```

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

```

Рис. 4.12: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` (рис. 4.13).

```

linux@Macbook:~/work/arch-pc/lab06$ touch lab6-3.asm

```

Рис. 4.13: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$  (рис. 4.14).

```
3 ; -----
4 ; -----
5 %include 'in_out.asm' ; подключение внешнего файла
6
7 SECTION .data
8
9 div: DB 'Результат: ',0
10 rem: DB 'Остаток от деления: ',0
11
12 SECTION .text
13 GLOBAL _start
14 _start:
15
16 ; ---- Вычисление выражения
17 mov eax,5 ; EAX=5
18 mov ebx,2 ; EBX=2
19 mul ebx ; EAX=EAX*EBX
20 add eax,3 ; EAX=EAX+3
21 xor edx,edx ; обнуляем EDX для корректной работы div
22 mov ebx,3 ; EBX=3
23 div ebx ; EAX=EAX/3, EDX=остаток от деления
24 mov edi,eax ; запись результата вычисления в 'edi'
25
26 ; ---- Вывод результата на экран
27
28 mov eax,div ; вызов подпрограммы печати
29 call sprint ; сообщения 'Результат: '
30 mov eax,edi ; вызов подпрограммы печати значения
31 call iprintf ; из 'edi' в виде символов
32
33 mov eax,rem ; вызов подпрограммы печати
34 call sprint ; сообщения 'Остаток от деления: '
35 mov eax,edx ; вызов подпрограммы печати значения
36 call iprintf ; из 'edx' (остаток) в виде символов
37
38 call quit ; вызов подпрограммы завершения
```

Рис. 4.14: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.15).

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

Рис. 4.15: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 4.16).

```

3
4
5 %include 'in_out.asm'          ; подключение внешнего файла
6
7 SECTION .data
8
9 div: DB 'Результат: ',0
10 rem: DB 'Остаток от деления: ',0
11
12 SECTION .text
13 GLOBAL _start
14 _start:
15
16 ; ---- Вычисление выражения
17 mov eax,4                      ; EAX=4
18 mov ebx,6                      ; EBX=6
19 mul ebx                       ; EAX=EAX*EBX
20 add eax,2                      ; EAX=EAX+2
21 xor edx,edx                   ; обнуляем EDX для корректной работы div
22 mov ebx,5                      ; EBX=5
23 div ebx                       ; EAX=EAX/5, EDX=остаток от деления
24 mov edi,eax                   ; запись результата вычисления в 'edi'
25
26 ; ---- Вывод результата на экран
27
28 mov eax,div                   ; вызов подпрограммы печати
29 call sprint                   ; сообщения 'Результат: '
30 mov eax,edi                   ; вызов подпрограммы печати значения
31 call tprintf                  ; из 'edi' в виде символов
32
33 mov eax,rem                   ; вызов подпрограммы печати
34 call sprint                   ; сообщения 'Остаток от деления: '
35 mov eax,edx                   ; вызов подпрограммы печати значения
36 call tprintf                  ; из 'edx' (остаток) в виде символов
37
38 call quit                     ; вызов подпрограммы завершения

```

Рис. 4.16: Изменение программы

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

```

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

```

Рис. 4.17: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 4.18).

```

linux@Macbook:~/work/arch-pc/lab06$ touch variant.asm

```

Рис. 4.18: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.19).





Рис. 4.19: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.20). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 7.

```

linux@Macbook:~/work/arch-pc/lab06$ nasm -f elf variant.asm
linux@Macbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
linux@Macbook:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032245486
Ваш вариант: 7

```

Рис. 4.20: 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

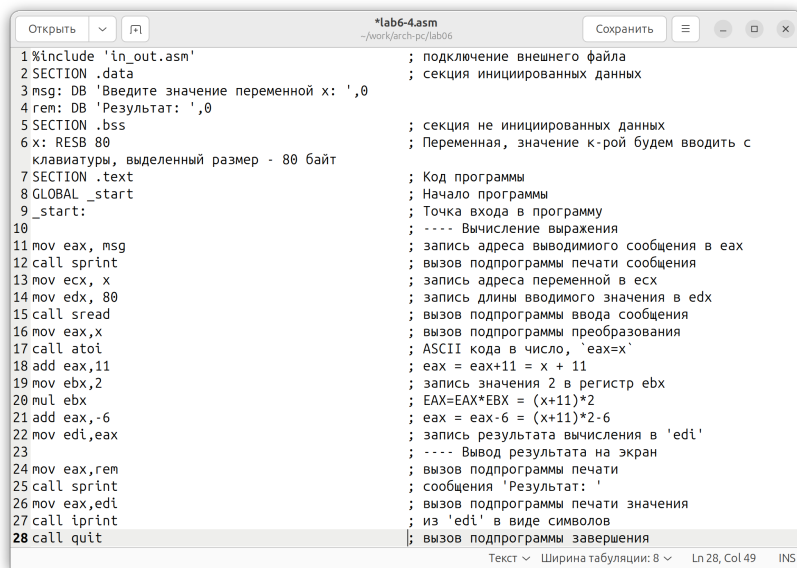
Создаю файл `lab6-4.asm` с помощью утилиты `touch` (рис. 4.21).



```
linux@Macbook:~/work/arch-pc/lab06$ touch lab6-4.asm
```

Рис. 4.21: Создание файла

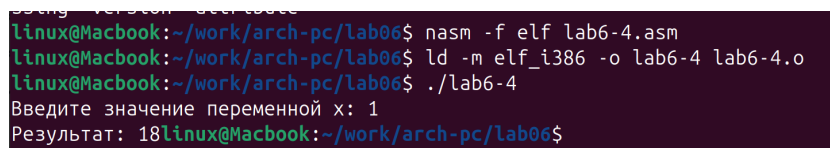
Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения  $(11 + x) * 2 - 6$  (рис. 4.22). Это выражение было под вариантом 8.



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

Рис. 4.22: Написание программы

Создаю и запускаю исполняемый файл (рис. 4.23). При вводе значения 1, вывод  
- 18.



```
linux@Macbook:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
linux@Macbook:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
linux@Macbook:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 18linux@Macbook:~/work/arch-pc/lab06$
```

Рис. 4.23: Запуск исполняемого файла

## **5 Выводы**

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

# **Список литературы**

1. Лабораторная работа №6