

# **Отчет по лабораторной работе №6**

**Арифметические операции в NASM**

Татьяна Александровна Буллер

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
2.1	Задание 1: . . . . .	5
2.2	Задание 2: . . . . .	5
2.3	Задание 3: . . . . .	7
2.4	Задание 4: . . . . .	8
2.5	Задание 5: . . . . .	9
2.5.1	Задание 5.1: . . . . .	11
2.6	Задание 6: . . . . .	11
2.6.1	Задание 6.1: . . . . .	13
2.7	Задание 7: . . . . .	15
2.8	Задание 8: . . . . .	19
2.8.1	1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? . . . . .	19
2.8.2	2. Для чего используется следующие инструкции? . . . . .	19
2.8.3	3. Для чего используется инструкция “call atoi”? . . . . .	19
2.8.4	4. Какие строки листинга 6.4 отвечают за вычисления варианта? . . . . .	19
2.8.5	5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”? . . . . .	19
2.8.6	6. Для чего используется инструкция “inc edx”? . . . . .	20
2.8.7	7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений? . . . . .	20
<b>3</b>	<b>Задание для самостоятельной работы</b>	<b>21</b>
3.1	Задание 8: . . . . .	21
<b>4</b>	<b>Вывод</b>	<b>24</b>

## Список иллюстраций

2.1	Переход в каталог курса и введение команды на создание файла .	5
2.2	Копирование текста программы из листинга . . . . .	6
2.3	Создание и запуск исполняемого файла . . . . .	6
2.4	Исправленный текст программы . . . . .	7
2.5	Создание и запуск исполняемого файла . . . . .	7
2.6	Преобразованный текст программы . . . . .	8
2.7	Создание и запуск исполняемого файла . . . . .	9
2.8	Отредактированный текст программы . . . . .	10
2.9	Создание и запуск исполняемого файла . . . . .	10
2.10	Вывод измененной программы . . . . .	11
2.11	Текст программы . . . . .	12
2.12	Создание и запуск исполняемого файла . . . . .	13
2.13	Измененный текст программы . . . . .	14
2.14	Создание и запуск исполняемого файла . . . . .	15
2.15	Текст программы для определения варианта . . . . .	16
2.16	Создание и запуск исполняемого файла . . . . .	17
2.17	Проверка калькулятором . . . . .	18
3.1	Код новой программы . . . . .	22
3.2	Компиляция программы . . . . .	23

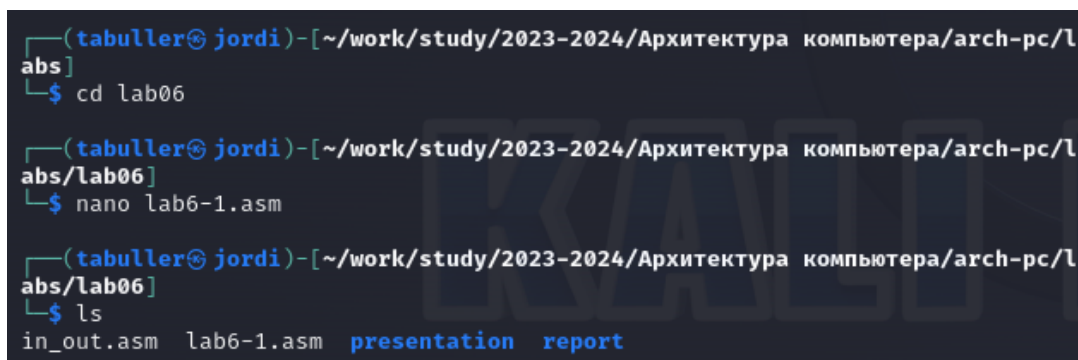
# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

### 2.1 Задание 1:

Создайте каталог для программ лабораторной работы № 6, перейдите в него и создайте файл `lab6-1.asm`.



```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
abs
$ cd lab06

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
abs/lab06
$ nano lab6-1.asm

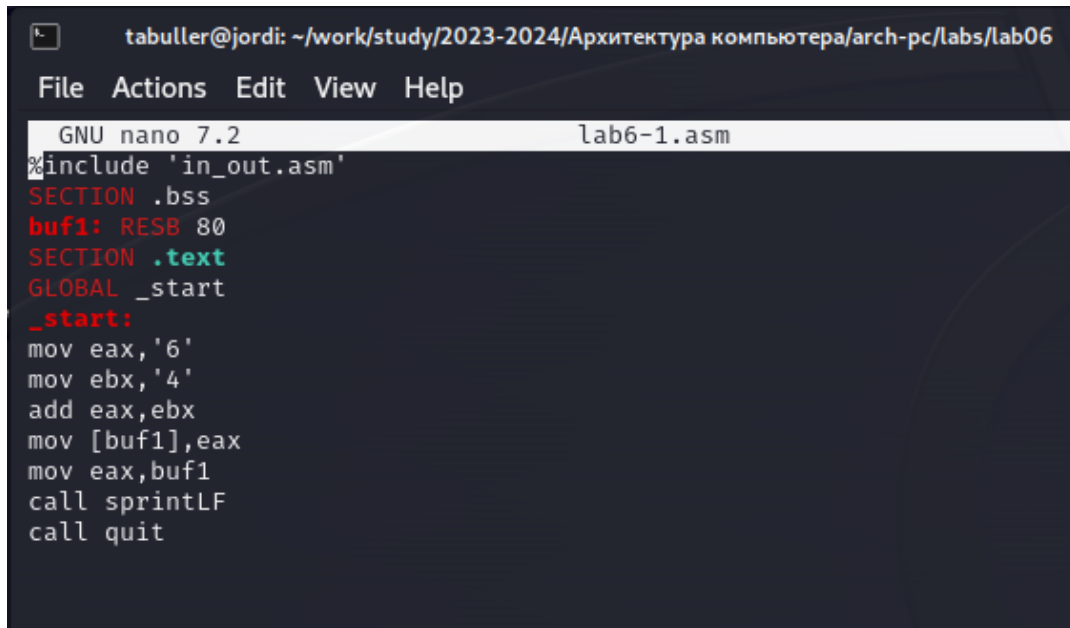
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
abs/lab06
$ ls
in_out.asm  lab6-1.asm  presentation  report
```

Рис. 2.1: Переход в каталог курса и введение команды на создание файла

Для дальнейшего успешного выполнения программы из листинга необходимо, чтобы файл `in_out.asm` находился в одном каталоге с рабочими файлами. Он был скопирован заранее, правильность копирования проверена с помощью команды `ls`.

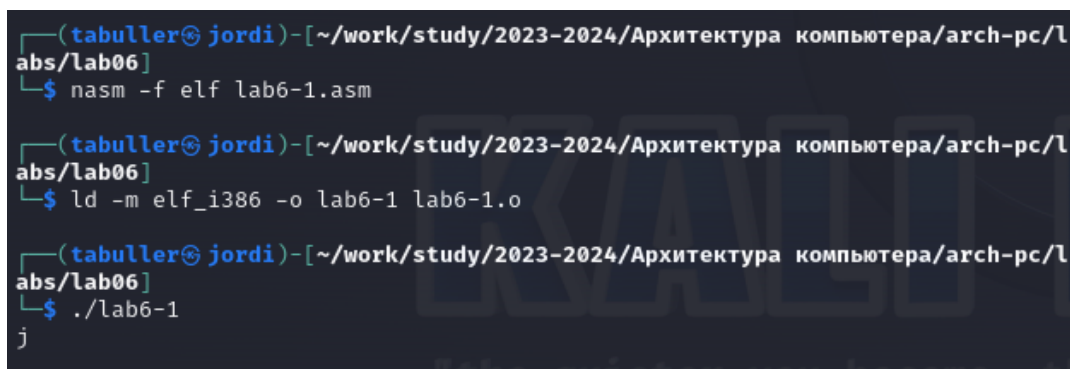
### 2.2 Задание 2:

Введите в файл `lab6-1.asm` текст программы из листинга 6.1, создайте исполняемый файл и запустите его.



```
tabuller@jordi: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06
File Actions Edit View Help
GNU nano 7.2 lab6-1.asm
%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 sprintLF
call quit
```

Рис. 2.2: Копирование текста программы из листинга



```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$ nasm -f elf lab6-1.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$ ld -m elf_i386 -o lab6-1 lab6-1.o

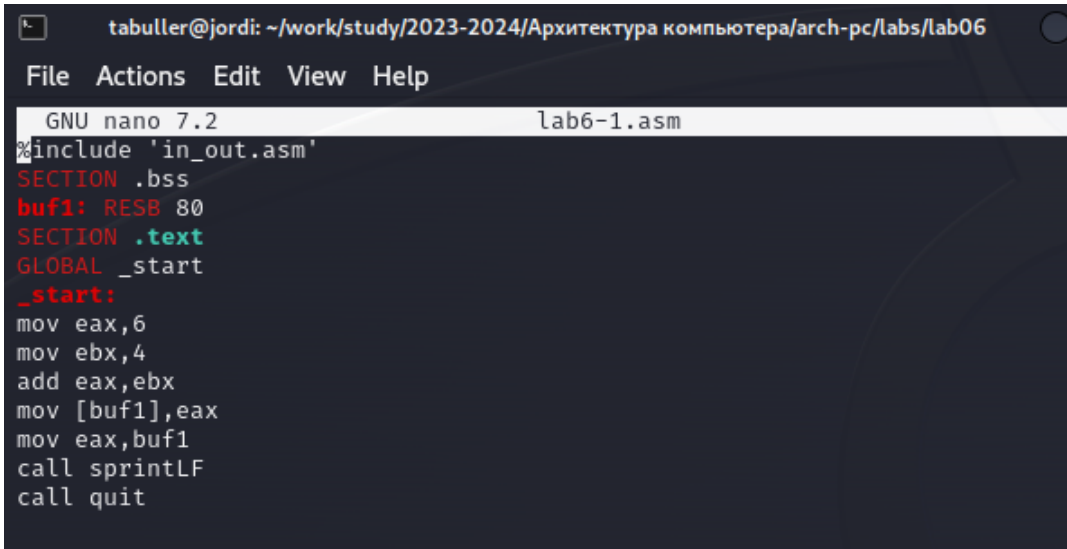
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$ ./lab6-1
j
```

Рис. 2.3: Создание и запуск исполняемого файла

Чего и следовало ожидать, вместо числа 10 в результат выводится символ 'j': в регистр `eax` была записана сумма кодов символов (106), что соответствует коду символа 'j'.

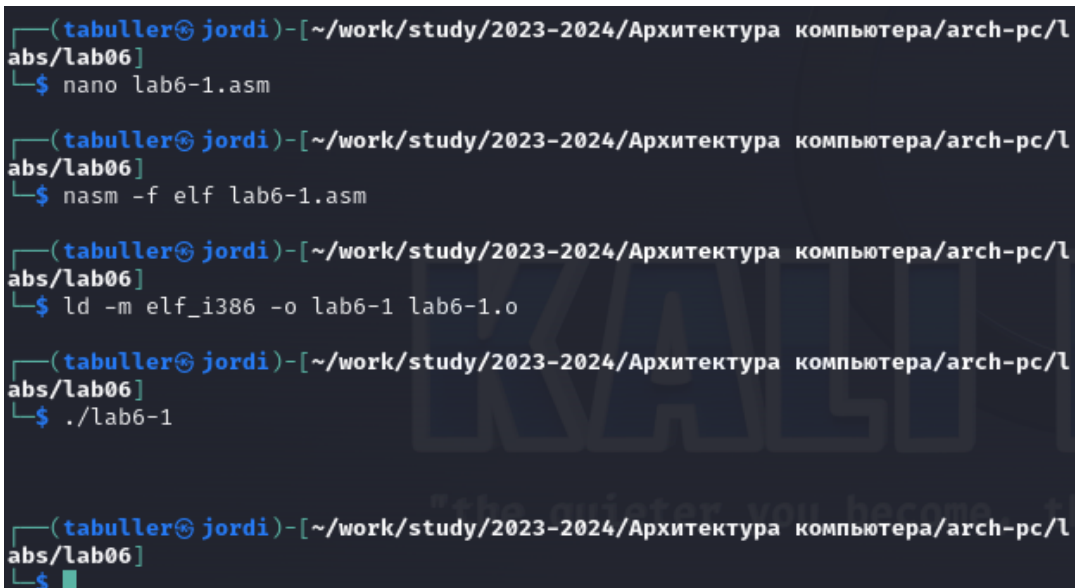
## 2.3 Задание 3:

Исправьте текст программы, записав в регистры числа. Создайте исполняемый файл и запустите его. Пользуясь таблицей ASCII определите какому символу соответствует код 10. Отображается ли этот символ при выводе на экран?



```
tabuller@jordi: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06
File Actions Edit View Help
GNU nano 7.2 lab6-1.asm
%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 sprintLF
call quit
```

Рис. 2.4: Исправленный текст программы



```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$ nano lab6-1.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$ nasm -f elf lab6-1.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$ ld -m elf_i386 -o lab6-1 lab6-1.o

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$ ./lab6-1

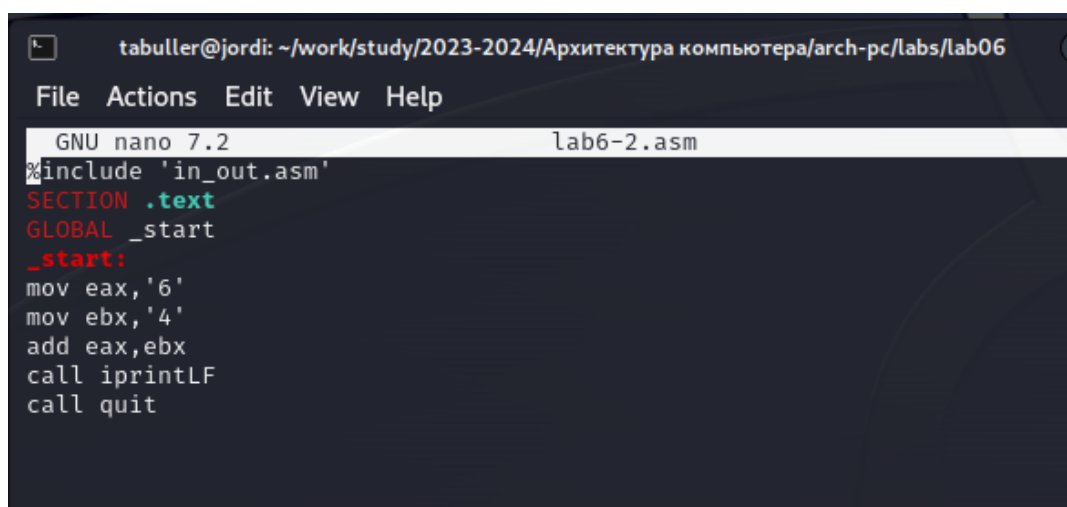
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$
```

Рис. 2.5: Создание и запуск исполняемого файла

Коду 10 соответствует символ VT (vertical tab). Это прозрачный символ - что-то такое, собственно, и вывелось на экран.

## 2.4 Задание 4:

Преобразуем текст программы из Листинга 6.1 с использованием функций для преобразования ASCII символов в числа и обратно из файла `in_out.asm`. Создайте файл `lab6-2.asm` и введите в него текст программы из листинга 6.2. Создайте исполняемый файл и запустите его.



```
tabuller@jordi: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06
File Actions Edit View Help
GNU nano 7.2 lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 2.6: Преобразованный текст программы



```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ nano lab6-2.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ nasm -f elf lab6-2.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ ld -m elf_i386 -o lab6-2 lab6-2.o

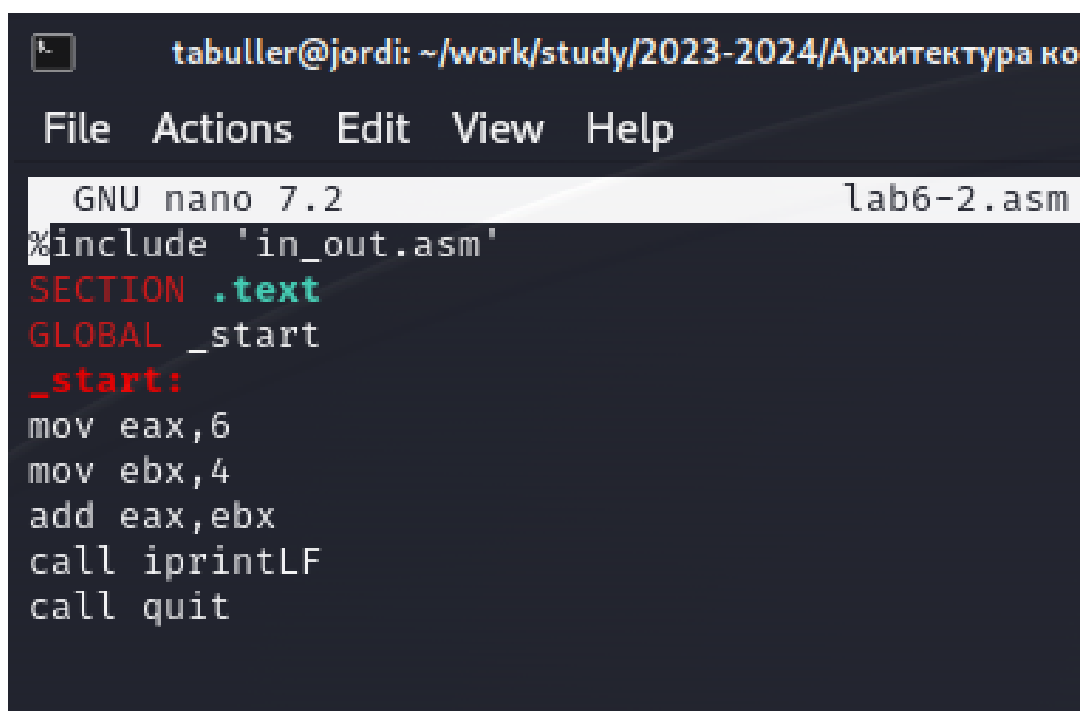
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ ./lab6-2
106
```

Рис. 2.7: Создание и запуск исполняемого файла

В результат вывелось число 106: складываются коды символов '6' и '4', но функция `iprintLF` выводит число, а не символ, кодом которого является это число.

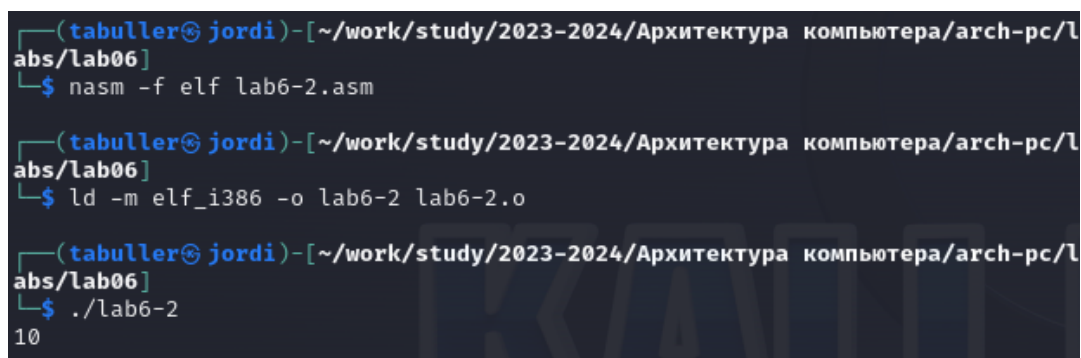
## 2.5 Задание 5:

Аналогично предыдущему примеру изменим символы на числа. Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы?



```
tabuller@jordi: ~/work/study/2023-2024/Архитектура ко
File Actions Edit View Help
GNU nano 7.2 lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 2.8: Отредактированный текст программы



```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab6]
abs/lab06
$ nasm -f elf lab6-2.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab6]
abs/lab06
$ ld -m elf_i386 -o lab6-2 lab6-2.o

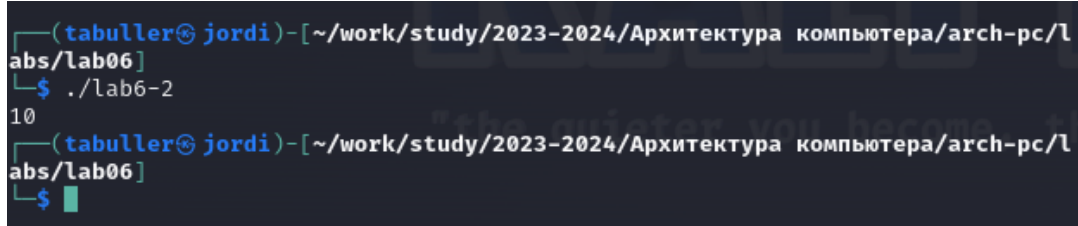
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab6]
abs/lab06
$ ./lab6-2
10
```

Рис. 2.9: Создание и запуск исполняемого файла

В результат вывелось число 10: сложились сами числа, а не их коды; программа отработала так, как было запланировано изначально.

### 2.5.1 Задание 5.1:

Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`?



```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
abs/lab06]
$ ./lab6-2
10
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
abs/lab06]
$
```

Рис. 2.10: Вывод измененной программы

Программа скомпилировалась без значительных изменений, но замена `sprintLF` на `sprint` привела к тому, что исчез символ переноса строки при выводе сообщения на экран.

## 2.6 Задание 6:

Создайте файл `lab6-3.asm` и введите в него текст программы из листинга 6.3. Создайте исполняемый файл и запустите его.

```
GNU nano 7.2 lab6-3.asm
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx
mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 2.11: Текст программы

```

10
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ nano lab6-3.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ nasm -f elf lab6-3.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ ld -m elf_i386 -o lab6-3 lab6-3.o

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ ./lab6-2
10
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 2.12: Создание и запуск исполняемого файла

Результат программы соответствует заданной функции:  $13/3$  при целочисленном делении даст результат 4 с остатком 1.

### 2.6.1 Задание 6.1:

Измените текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ . Создайте исполняемый файл и проверьте его работу.

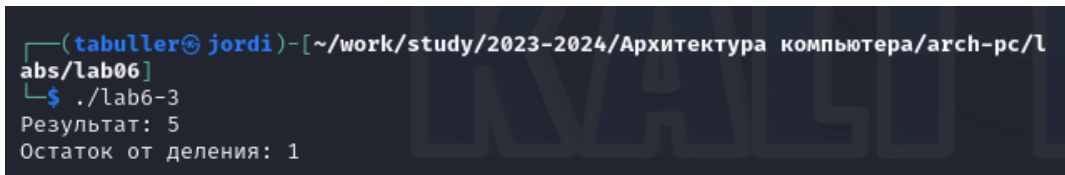
```
GNU nano 7.2                                     lal
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 2.13: Измененный экст программы

A terminal window with a dark background. The prompt is `(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]`. The user enters `./lab6-3`. The output shows `Результат: 5` and `Остаток от деления: 1`.

```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06]
$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 2.14: Создание и запуск исполняемого файла

Для изменения функции необходимо только заменить значения переменных. Результат остается верным:  $26/5$  при целочисленном делении дает 5 с остатком 1.

## 2.7 Задание 7:

Создайте файл `variant.asm` и введите в него текст программы из листинга 6.4. Проверьте результат работы программы вычислив номер варианта аналитически.



```
tabuller@jordi: ~/work/study/2023-2024/Архитектура компьютера/а
File Actions Edit View Help
GNU nano 7.2 lab6-4.asm *

%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 sprintLF

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

xor edx, edx
mov ebx, 20
div ebx
inc edx

mov eax, rem
call sprint
```

^G Help ^O Write Out ^W Where Is ^K Cut

Рис. 2.15: Текст программы для определения варианта



```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab6/abs/lab06]
$ nano lab6-4.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab6/abs/lab06]
$ nasm -f elf lab6-4.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab6/abs/lab06]
$ ld -m elf_i386 -o lab6-4 lab6-4.o

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab6/abs/lab06]
$ ./lab6-4
Введите № студенческого билета:
1132231835
Ваш вариант: 16
```

Рис. 2.16: Создание и запуск исполняемого файла

Программа отработывает без ошибок, выводя в результат номер варианта 16.  
Проверим результат аналитически:

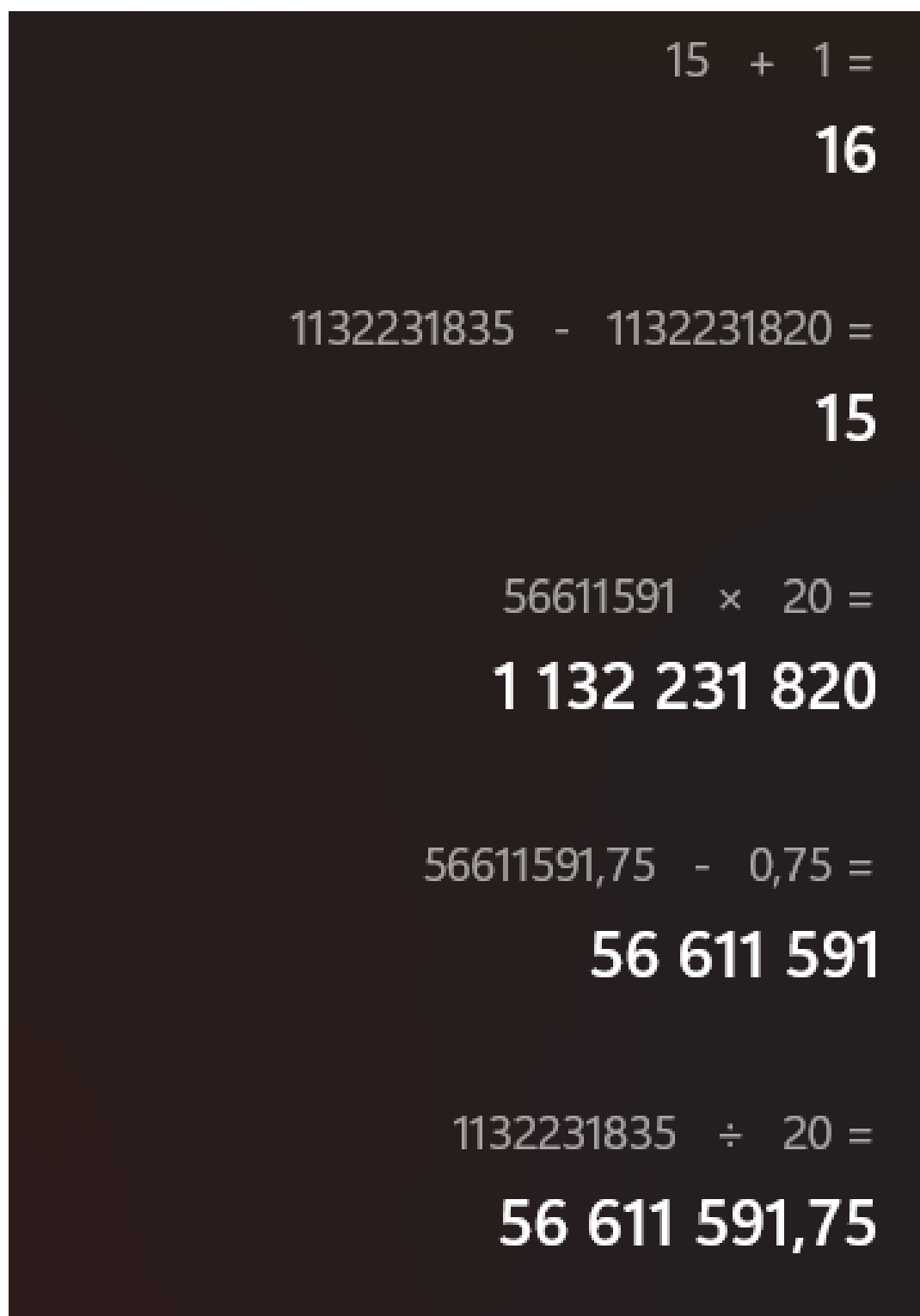


Рис. 2.17: Проверка калькулятором

Результаты совпадают: можно утверждать, что программа отработала верно при заданном номере студенческого билета.

## **2.8 Задание 8:**

Включите в отчет по выполнению лабораторной работы ответы на следующие вопросы:

### **2.8.1 1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?**

```
mov eax,rem call sprint
```

### **2.8.2 2. Для чего используются следующие инструкции?**

```
mov ecx, x mov edx, 80 call sread
```

Эти инструкции выполняют считывание ввода пользователя, в дальнейшем полученное значение передается в переменную `eax`.

### **2.8.3 3. Для чего используется инструкция “call atoi”?**

Для преобразования ASCII символов в числа.

### **2.8.4 4. Какие строки листинга 6.4 отвечают за вычисления варианта?**

```
xor edx,edx mov ebx,20 div ebx inc edx
```

### **2.8.5 5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?**

`edx`

### **2.8.6 6. Для чего используется инструкция “inc edx”?**

Увеличение на 1 значения регистра edx.

### **2.8.7 7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?**

```
mov eax,edx call iprintLF
```

## 3 Задание для самостоятельной работы

### 3.1 Задание 8:

Написать программу вычисления выражения  $y = f(x)$ . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3.

Функция в варианте 16:  $(10x - 5)^2$ . Разберем алгоритм: сперва программа должна получить ввод от пользователя ( $x$ ), затем умножить полученное число на 10, вычесть из результата 5 и умножить новый результат сам на себя. Функции умножения и считывания ввода пользователя уже известны, функция для вычитания - `sub`, ее синтаксис аналогичен синтаксису команды `add`.

```

#include 'in_out.asm'
SECTION .data
msg: DB 'Введите число: ',0
rem: DB 'f(x) = ',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
call atoi
mov ebx, 10
mul ebx
sub eax, 5
xor edx, edx
mov ebx, eax
mul ebx
mov edi, eax

```

Рис. 3.1: Код новой программы

Первый блок команд после команды “старт” вызывает сообщение ‘Введите число’. Следующий блок считывает введенное число. Далее переменной `eax` присваивается значение `x`, `ebx` - 10, после чего выполняется умножение `ebx` на `eax`. Из результата в переменной `eax` вычитается 5, после чего полученное число умножается само на себя.



```
(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ nano lab6-5.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ nasm -f elf lab6-5.asm

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ ld -m elf_i386 -o lab6-5 lab6-5.o

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ ./lab6-5
Введите число:
3
f(x) = 625

(tabuller@jordi)-[~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06]
$ ./lab6-5
Введите число:
1
f(x) = 25
```

Рис. 3.2: Компиляция программы

Для предложенных чисел 3 и 1 программа выдает результаты 625 и 25 соответственно. Это сходится с результатами, которые можно получить при самостоятельном решении: программа отработала верно.

## 4 Вывод

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