

# **Лабораторная работа №6**

**Дисциплина: Архитектура компьютера**

Савостин Олег

# 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 . . . . .	13
4.3	Ответы на вопросы . . . . .	17
4.4	Выполнение заданий для самостоятельной работы . . . . .	19
<b>5</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

4.1	Создание нужного каталога и lab6-1.asm . . . . .	9
4.2	Введение листинга 6.1 в lab6-1.asm . . . . .	10
4.3	Создание и запуск исполняемого файла lab6-1 . . . . .	10
4.4	Изменения в файле . . . . .	11
4.5	Создание и запуск нового файла . . . . .	11
4.6	Новый файл lab6-2.asm . . . . .	11
4.7	Текст из листинга 6.2 . . . . .	11
4.8	Создание и запуск исполняемого файла . . . . .	12
4.9	Изменения файла lab6-2.asm . . . . .	12
4.10	Создание и запуск исполняемого файла lab6-2 . . . . .	12
4.11	Изменение ipringLF на iprint . . . . .	13
4.12	Исполняемый файл при iprint . . . . .	13
4.13	Создание файла lab6-3.asm . . . . .	13
4.14	Текст кода в созданном файле lab6-3.asm . . . . .	14
4.15	Результат кода исполняемого файла . . . . .	14
4.16	Текст измененного кода . . . . .	15
4.17	Проверка на правильность измененного кода . . . . .	15
4.18	Создание файла . . . . .	17
4.19	Текст кода листинга 6.4 . . . . .	17
4.20	Запуск программы . . . . .	17
4.21	Создание файла function.asm . . . . .	19
4.22	Код файла . . . . .	19
4.23	Проверка на правильность. . . . .	20

## **Список таблиц**

# 1 Цель работы

Целью данной работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Символьные и численные данные в NASM.
2. Выполнение арифметических операций в NASM.
3. Ответы на вопросы.
4. Задание 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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации.

Существует три основных способа адресации: • 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

Схема команды целочисленного сложения `add` (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Допустимые сочетания операндов для команды `add` аналогичны сочетаниям операндов для команды `mov`.

Так, например, команда `add eax,ebx` прибавит значение из регистра `eax` к значению из регистра `ebx` и запишет результат в регистр `eax`

Команда целочисленного вычитания `sub` (от англ. subtraction – вычитание) работает аналогично команде `add`.

Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание – декрементом. Для этих операций существуют специальные ко-

манды: inc (от англ. increment) и dec (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд. Команда neg рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды

Для беззнакового умножения используется команда mul

Для знакового умножения используется команда imul

Для деления, как и для умножения, существует 2 команды div (от англ. divide - деление) и idiv



## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

Сперва, я создаю каталог, где буду делать лабораторную работу, в папке arch-pc и lab6-1.asm (рис. 4.1).

```
savostinoleg@vbox:~$ cd work/study/2024-2025/"Архитектура компьютера"/arch-pc/  
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdir lab06  
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab06  
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-  
1.asm  
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ls  
lab6-1.asm  
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.1: Создание нужного каталога и lab6-1.asm

В созданный мною файл я ввожу текст программы из листинга 6.1 (Архитектура ЭВМ, ТУИС РУДН, Лабораторная работа №6 )(рис. 4.2).

```
lab6-1.asm [---] 0 L: [ 1+ 0 1/ 18] *(0 / 177b) 0037 0x025 [*][X]
#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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ить 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 4.2: Введение листинга 6.1 в lab6-1.asm

Теперь я создаю исполняемый файл и запускаю его. При исполнении программы, выводится символ “j”(рис. 4.3).

```
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-1.asm
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-1
j
```

Рис. 4.3: Создание и запуск исполняемого файла lab6-1

Далее изменяю текст программы и вместо символов записываю в регистры числа (рис. 4.4).

```

lab6-1.asm      [----]  0 L:[ 1+ 0  1/ 18] *(0  / 173b) 0037 0x025
#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. 4.4: Изменения в файле

Создаю исполняемый файл и запускаю его. Выводится символ, не отображаемый на экране(рис. 4.5).

```

savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-1.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-1

```

Рис. 4.5: Создание и запуск нового файла

Теперь я создаю новый файл (рис. 4.6) и в него ввожу текст программы из листинга 6.2 ( ТУИС РУДН ) (рис. 4.7).

```

savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-2.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$

```

Рис. 4.6: Новый файл lab6-2.asm

```

lab6-2.asm      [----]  0 L:[ 1+ 0  1/ 13] *(0  / 121b) 0037 0x025
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF

call quit

```

Рис. 4.7: Текст из листинга 6.2

Теперь, я создаю исполняемый файл и запускаю его. После запуска на экран выводится 106, так как в программе складываются коды символов '6' и '4' = 54+52=106 (рис. 4.8).

```
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ nasm -f elf lab6-2.asm
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ ld -m elf_i386 -o lab6-2 lab6-2.o
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ ./lab6-2
106
```

Рис. 4.8: Создание и запуск исполняемого файла

Теперь повторяю действия, проделанные с предыдущим файлом и записываю 6 и 4 как числа в текст.(рис. 4.9). Создаю исполняющий файл и запускаю его (рис. 4.10). На экран выводится число 10. Значит были сложены 6 и 4.

```
lab6-2.asm [BM--] 8 L:[ 1+ 7 8/ 13] *(7
%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: Изменения файла lab6-2.asm

```
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ nasm -f elf lab6-2.asm
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ ld -m elf_i386 -o lab6-2 lab6-2.o
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ ./lab6-2
10
```

Рис. 4.10: Создание и запуск исполняемого файла lab6-2

Теперь заменяю функцию “iprintLF” на “iprint”(рис. 4.11). При запуске исполняемого файла видим, что результат пишется на одной строчке с строчкой, где

указан путь к каталогу, в котором нахожусь.(рис. 4.12).

```
lab6-2.asm [-M--] 11 L:[ 1+ 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. 4.11: Изменение ipringLF на iprint

```
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
10savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
10savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.12: Исполняемый файл при iprint

## 4.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических действий в NASM приведу программу для вычисления  $f(x)=(5*2+3)/3$

Для начала, создаю файл lab6-3.asm (рис. 4.13) и записываю в него текст кода из листинга 6.3 (ТУИС РУДН) (рис. 4.14).

```
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-3.asm
```

Рис. 4.13: Создание файла lab6-3.asm

```

lab6-3.asm      [----]  0 L: [ 1+ 0  1/ 39] *(0  /1375b) 0059 0x03B      [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 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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пере-тить 7Поиск 8Удалить 9МенюМС 10Выход

```

Рис. 4.14: Текст кода в созданном файле lab6-3.asm

Создаю исполняемый файл и запускаю его. Результат соответствует результатам на ТУИС(рис. 4.15).

```

savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-3.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 4.15: Результат кода исполняемого файла

Теперь, я изменяю код чтобы было  $f(x) = (4*6+2)/5$  (рис. 4.16) и создаю исполняемый файл что проверить на правильность проведенных действий. Всё верно.(рис. 4.17).

```

lab6-3.asm      [----]  0 L: [ 2+37  39/ 39] *(1339/1339b) <EOF>      [*][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,4
mov ebx,6
mul ebx ; EAX=EAX*EBX
add eax,2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5
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 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. 4.16: Текст измененного кода

```

savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ nasm -f elf lab6-3.asm
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ ld -m elf_i386 -o lab6-3 lab6-3.o
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ ./lab6-3
Результат: 5
Остаток от деления: 1
savostinoleg@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$

```

Рис. 4.17: Проверка на правильность измененного кода

Код:

%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

```

mov ebx,6
mul ebx ; EAX=EAX*EBX
add eax,2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5
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 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

```

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. Сперва, я создаю файл variant.asm(рис. 4.18), в который я вставляю текст из листинга 6.4 ( ТУИС РУДН ) (рис. 4.19). и узнаю свой вариант(рис. 4.20). Мне был выбран 13 вариант, следовательно я буду делать задачу номер 13. Мой студенческий номер является 1032245472. Данная команда делит мой номер на 20 и учитывает только остаток. Остатком является 12. Затем добавляется 1, откуда и берется 13.



```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch variant.asm
```

Рис. 4.18: Создание файла

```
variant.asm [----] 0 L:[ 1+ 0 1/ 35] *(0 / 500b) 0037 0x025
#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 sprintfLF

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 sprint
```

Рис. 4.19: Текст кода листинга 6.4

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch variant.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf variant.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032245472
Ваш вариант: 13
```

Рис. 4.20: Запуск программы

## 4.3 Ответы на вопросы

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

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

2. Для чего используются следующие инструкции?

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

mov ecx, x для того, чтобы положить адрес вводимой строки x в регистр  
ecx mov edx,80 - запись в регистр длины вводимой строки  
call sread - вызов подпрограммы из внешнего файла который обеспечивает  
ввод сообщения с клавиатуры

3. Для чего используется инструкция “call atoi”?

Данная инструкция используется для вызова подпрограммы из внешнего файла. Он преобразует код символа ascii в целое число, затем записывает в регистр eax

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

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

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

Остаток от деления записывается в edx

6. Для чего используется инструкция “inc edx”?

Данная инструкция увеличивает значение регистра edx на 1

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычисления?

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

## 4.4 Выполнение заданий для самостоятельной работы

Мне следует написать код, который будет вычислять  $(8x+6)*10$  при переменных 1 и 4, так как у меня 13 вариант.

Сперва, создаю файл в котором буду писать код(рис. 4.21).

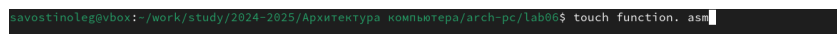
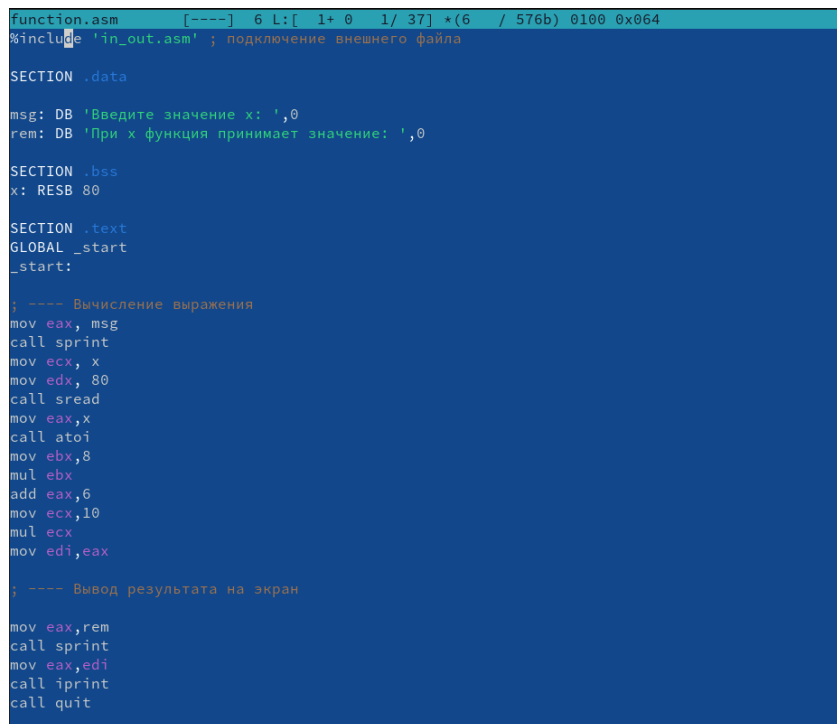


Рис. 4.21: Создание файла function.asm

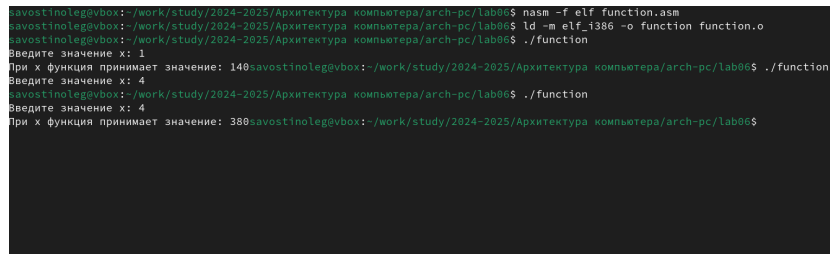
Записываю код который будет вычислять уравнение. (рис. 4.22).



```
function.asm  [----]  6 L:[ 1+ 0  1/ 37] *(6  / 576b) 0100 0x064  
%include 'in_out.asm' ; подключение внешнего файла  
  
SECTION .data  
msg: DB 'Введите значение x: ',0  
rem: DB 'При x функция принимает значение: ',0  
  
SECTION .bss  
x: RESB 80  
  
SECTION .text  
GLOBAL _start  
_start:  
  
; ---- Вычисление выражения  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
mov ebx, 8  
mul ebx  
add eax, 6  
mov ecx, 10  
mul ecx  
mov edi, eax  
  
; ---- Вывод результата на экран  
mov eax, rem  
call sprint  
mov eax, edi  
call iprint  
call quit
```

Рис. 4.22: Код файла

Проверяю на правильность написанного кода. Всё верно. (рис. 4.23).



```
savostinolege@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf function.asm
savostinolege@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o function function.o
savostinolege@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./function
Введите значение x: 1
При x функция принимает значение: 140savostinolege@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./function
Введите значение x: 4
savostinolege@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./function
Введите значение x: 4
При x функция принимает значение: 388savostinolege@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.23: Проверка на правильность.

Код:

%include 'in\_out.asm' ; подключение внешнего файла

SECTION .data

msg: DB 'Введите значение x:',0

rem: DB 'При x функция принимает значение:',0

SECTION .bss

x: RESB 80

SECTION .text

GLOBAL \_start

\_start:

; -- Вычисление выражения

mov eax, msg

call sprint

mov ecx, x

mov edx, 80

call sread

mov eax,x

call atoi

mov ebx,8

mul ebx

add eax,6

mov ecx,10

```
mul ecx
mov edi,eax
; --- Вывод результата на экран
mov eax,rem
call sprint
mov eax,edi
call iprint
call quit
```

## **5 Выводы**

В ходе данной лабораторной работы, я освоил арифметических инструкций языка ассемблера NASM.

# **Список литературы**

1. Лабораторная работа №6