

# **Отчёт по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

Горобцова Арина Романовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Символьные и численные данные в NASM . . . . .	7
3.2	Выполнение арифметических операций в NASM . . . . .	12
3.3	Ответы на вопросы по программе . . . . .	15
3.4	Выполнение заданий для самостоятельной работы . . . . .	16
<b>4</b>	<b>Выводы</b>	<b>19</b>

# Список иллюстраций

3.1	Создание директории и файла . . . . .	7
3.2	Редактирование файла . . . . .	8
3.3	Запуск исполняемого файла . . . . .	8
3.4	Редактирование файла . . . . .	8
3.5	Запуск исполняемого файла . . . . .	9
3.6	Создание файла . . . . .	9
3.7	Открытие файла для просмотра . . . . .	9
3.8	Редактирование файла . . . . .	10
3.9	Запуск исполняемого файла . . . . .	10
3.10	Редактирование файла . . . . .	11
3.11	Запуск исполняемого файла . . . . .	11
3.12	Редактирование файла . . . . .	12
3.13	Запуск исполняемого файла . . . . .	12
3.14	Создание файла . . . . .	12
3.15	Редактирование файла . . . . .	13
3.16	Запуск исполняемого файла . . . . .	13
3.17	Изменение программы . . . . .	14
3.18	Запуск исполняемого файла . . . . .	14
3.19	Создание файла . . . . .	14
3.20	Редактирование файла . . . . .	15
3.21	Запуск исполняемого файла . . . . .	15
3.22	Создание файла . . . . .	16
3.23	Написание программы . . . . .	17
3.24	Запуск исполняемого файла . . . . .	17

## **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 Выполнение лабораторной работы

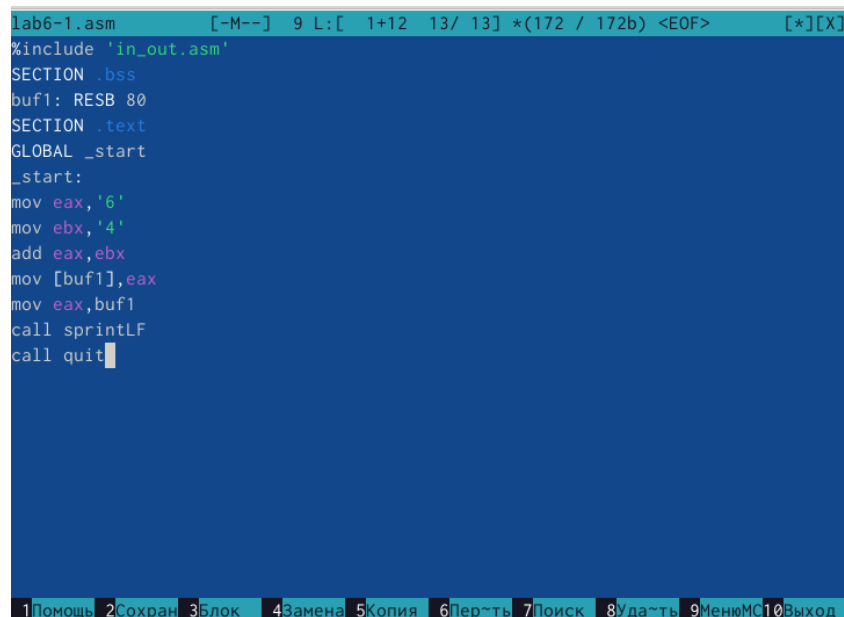
### 3.1 Символьные и численные данные в NASM

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

```
argorobcova@dk3n35 ~ $ mkdir ~/work/arch-pc/lab06
argorobcova@dk3n35 ~ $ cd ~/work/arch-pc/lab06
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ touch lab6-1.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 3.1: Создание директории и файла

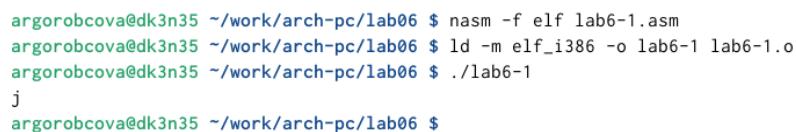
Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 3.2).



```
lab6-1.asm      [-M--]  9 L:[ 1+12 13/ 13] *(172 / 172b) <EOF>  [*][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 sprintf
call quit
```

Рис. 3.2: Редактирование файла

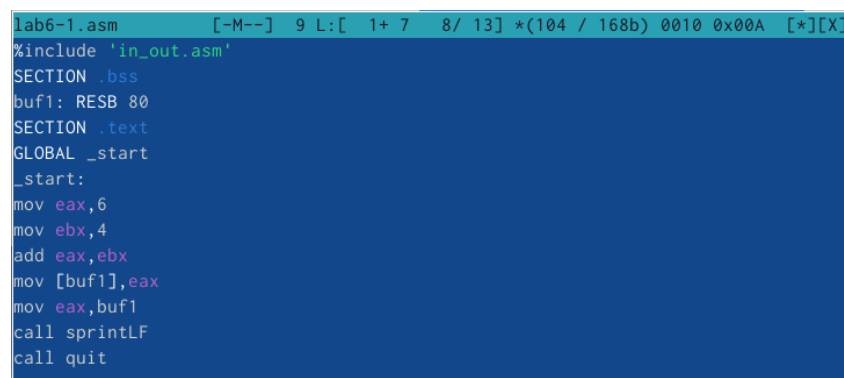
Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6 (рис. 3.3).



```
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-1
j
argorobcova@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 3.3: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 3.4).



```
lab6-1.asm      [-M--]  9 L:[ 1+ 7  8/ 13] *(104 / 168b) 0010 0x00A  [*][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 sprintf
call quit
```

Рис. 3.4: Редактирование файла



Создаю новый исполняемый файл программы и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран (рис. 3.5).

```
argorbcova@dk3n35 ~ $ cd work/arch-pc/lab06
argorbcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
argorbcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
argorbcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-1

argorbcova@dk3n35 ~/work/arch-pc/lab06 $ █
```

Рис. 3.5: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 3.6).

```
argorbcova@dk3n35 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
argorbcova@dk3n35 ~/work/arch-pc/lab06 $
```

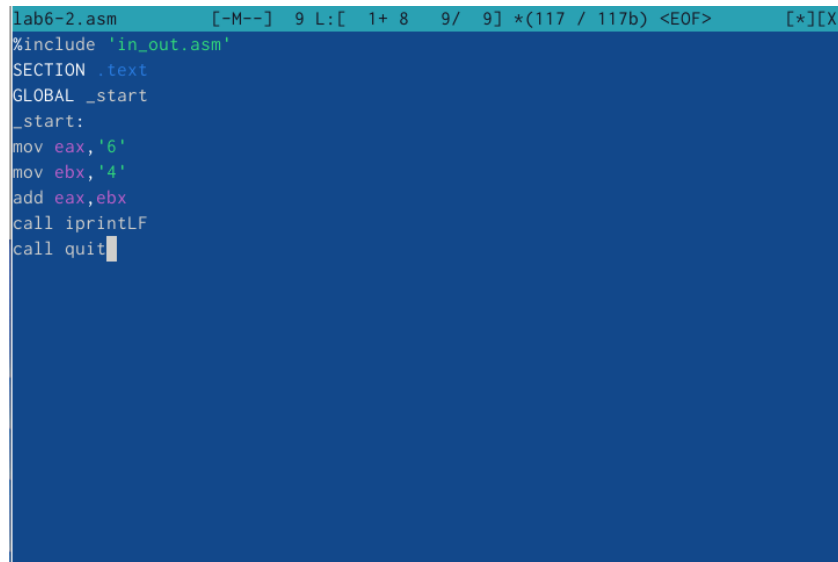
Рис. 3.6: Создание файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 3.7).

```
argorbcova@dk3n35 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
argorbcova@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 3.7: Открытие файла для просмотра

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 3.8).



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

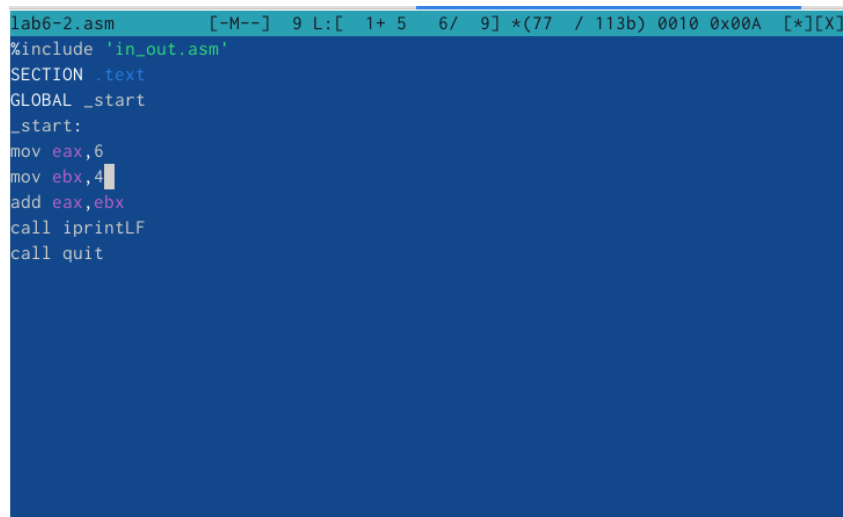
Рис. 3.8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2. 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”. (рис. 3.9).

```
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-2
106
argorobcova@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 3.9: Запуск исполняемого файла

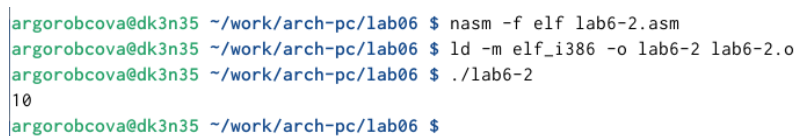
Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 3.10).



```
lab6-2.asm [-M--] 9 L: [ 1+ 5 6/ 9] *(77 / 113b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 3.10: Редактирование файла

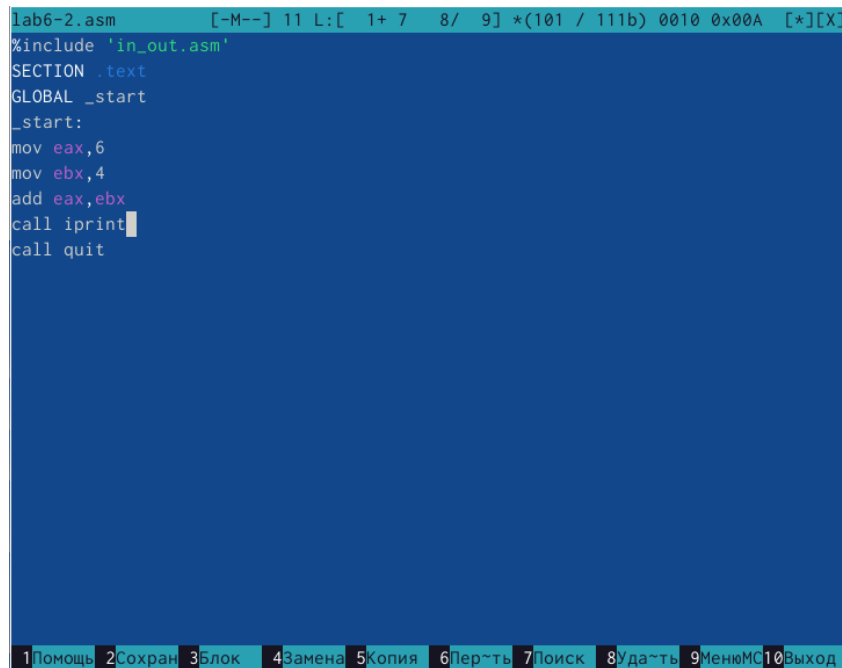
Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символы коды в системе ASCII, а сами числа, поэтому вывод 10. (рис. 3.11).



```
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-2
10
argorobcova@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 3.11: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 3.12).



```
lab6-2.asm      [-M--] 11 L: [ 1+ 7  8/  9] *(101 / 111b) 0010 0x00A  [*][X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
call iprint
call quit
```

Рис. 3.12: 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. 3.13).

```
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-2
10argorobcova@dk3n35 ~/work/arch-pc/lab06 $ █
```

Рис. 3.13: Запуск исполняемого файла

## 3.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` (рис. 3.14).

```
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ █
```

Рис. 3.14: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$  (рис. 3.15).

```
lab6-3.asm      [-M--] 17 L:[ 1+12 13/ 29] *(417 /1364b) 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 iprintLF ; из 'edi' в виде символов
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 3.15: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 3.16).

```
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
argorobcova@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 3.16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 3.17).

```

lab6-3.asm      [-M--] 19 L: [ 6+12 18/ 29] *(580 /1365b) 0044 0x02C [*][X]
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC 10Выход

```

Рис. 3.17: Изменение программы

Создаю и запускаю новый исполняемый файл. Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно (рис. 3.18).

```

argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ █

```

Рис. 3.18: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 3.19).

```

argorobcova@dk3n35 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ █

```

Рис. 3.19: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 3.20).

```

variant.asm      [-M--]  9 L: [ 1+17 18/ 28] *(402 / 618b) 0032 0x020  [*][X]
;-----
; 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 sprintLF
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

```

Рис. 3.20: Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 20. (рис. 3.21).

```

argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant.asm variant.
asm.o
ld: невозможно найти variant.asm.o: Нет такого файла или каталога
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
nasm: fatal: unable to open input file 'variant.asm' No such file or directory
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132246819
Ваш вариант: 20
argorobcova@dk3n35 ~/work/arch-pc/lab06 $

```

Рис. 3.21: Запуск исполняемого файла

### 3.3 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`.  
запись в регистр `edx` длины вводимой строки `call strlen` - вызов подпрограммы из внешнего файла

3. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует а  
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
```

## 3.4 Выполнение заданий для самостоятельной работы

Создаю файл `lab6-4.asm` с помощью утилиты `touch` (рис. 3.22).



```
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-4.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 3.22: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения  $x^3 \cdot 1/3 + 21$ . Это выражение было под вариантом 20. (рис. 3.23).



```

lab6-4.asm      [-M--] 41 L: [ 1+15 16/ 28] *(1143/1825b) 1072 0x430 [*][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 р
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, 11; eax = eax+11 = x + 11
mov ebx, 2 ; запись значения 2 в регистр ebx
mul ebx; EAX=EAX*EBX = (x+11)*2
add eax, -6; eax = eax-6 = (x+11)*2-6
mov edi, eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9Меню 10Выход

```

Рис. 3.23: Написание программы

Создаю и запускаю исполняемый файл. При вводе значения 21, вывод - 48.  
(рис. 3.24).

```

argorobcova@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 1
Результат: 21argorobcova@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 3
Результат: 48argorobcova@dk3n35 ~/work/arch-pc/lab06 $

```

Рис. 3.24: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения  $(11 + x) * 2 - 6$ .

```

%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, выде-
леный 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, 11;  $eax = eax + 11 = x + 11$ 
mov ebx, 2 ; запись значения 2 в регистр ebx
mul ebx;  $EAX = EAX * EBX = (x + 11) * 2$ 
add eax, -6;  $eax = eax - 6 = (x + 11) * 2 - 6$ 
mov edi, eax ; запись результата вычисления в 'edi'
; --- Вывод результата на экран
mov eax, ret ; вызов подпрограммы печати
call sprint ; сообщения 'Результат:'
mov eax, edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

```

## 4 Выводы

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