

# **Отчёт по лабораторной работе №6**

**Дисциплина: Архитектура компьютеров**

Лазарев Даниил Михайлович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>5</b>	<b>Выполнение самостоятельной работы</b>	<b>16</b>
<b>6</b>	<b>Выводы</b>	<b>19</b>

## Список иллюстраций

4.1	Создание файла в каталоге . . . . .	7
4.2	Код программы в файле . . . . .	8
4.3	Преобразование в исполняемый файл . . . . .	9
4.4	Измененный код программы . . . . .	9
4.5	Преобразование измененного файла . . . . .	10
4.6	Проверка наличия текста . . . . .	10
4.7	Преобразование файла . . . . .	11
4.8	Преобразование исправленного файла . . . . .	11
4.9	Создание файла "lab6-3.asm" . . . . .	12
4.10	Проверка правильности работы файла . . . . .	12
4.11	Исправленный файл . . . . .	13
4.12	Преобразование файла в исполняемый . . . . .	13
4.13	Код программы в файле . . . . .	14
4.14	Преобразование файла и проверка работы . . . . .	14
4.15	Ответы на вопросы . . . . .	15
5.1	Код программы для вычисления функции . . . . .	17
5.2	Проверка работоспособности . . . . .	18

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Выполнение лабораторной работы №6
2. Заполнение отчета по выполнению лабораторной работы №6 с помощью языка разметки Markdown
3. Выполнение заданий для самостоятельной работы

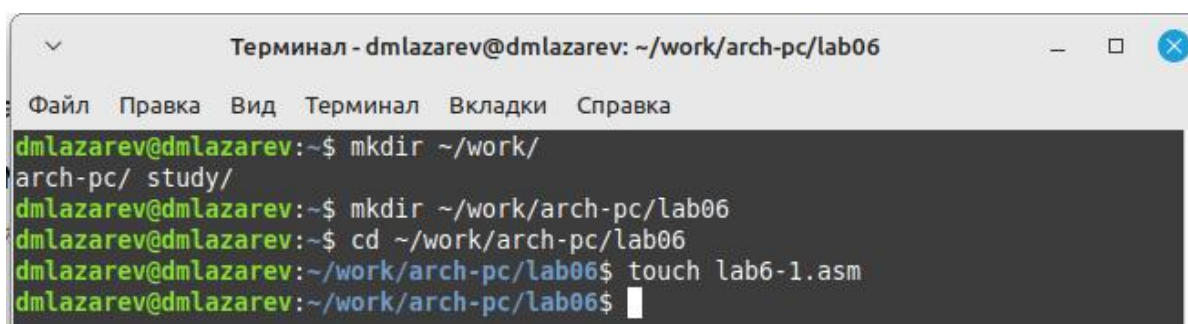
### 3 Теоретическое введение

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. 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.

## 4 Выполнение лабораторной работы

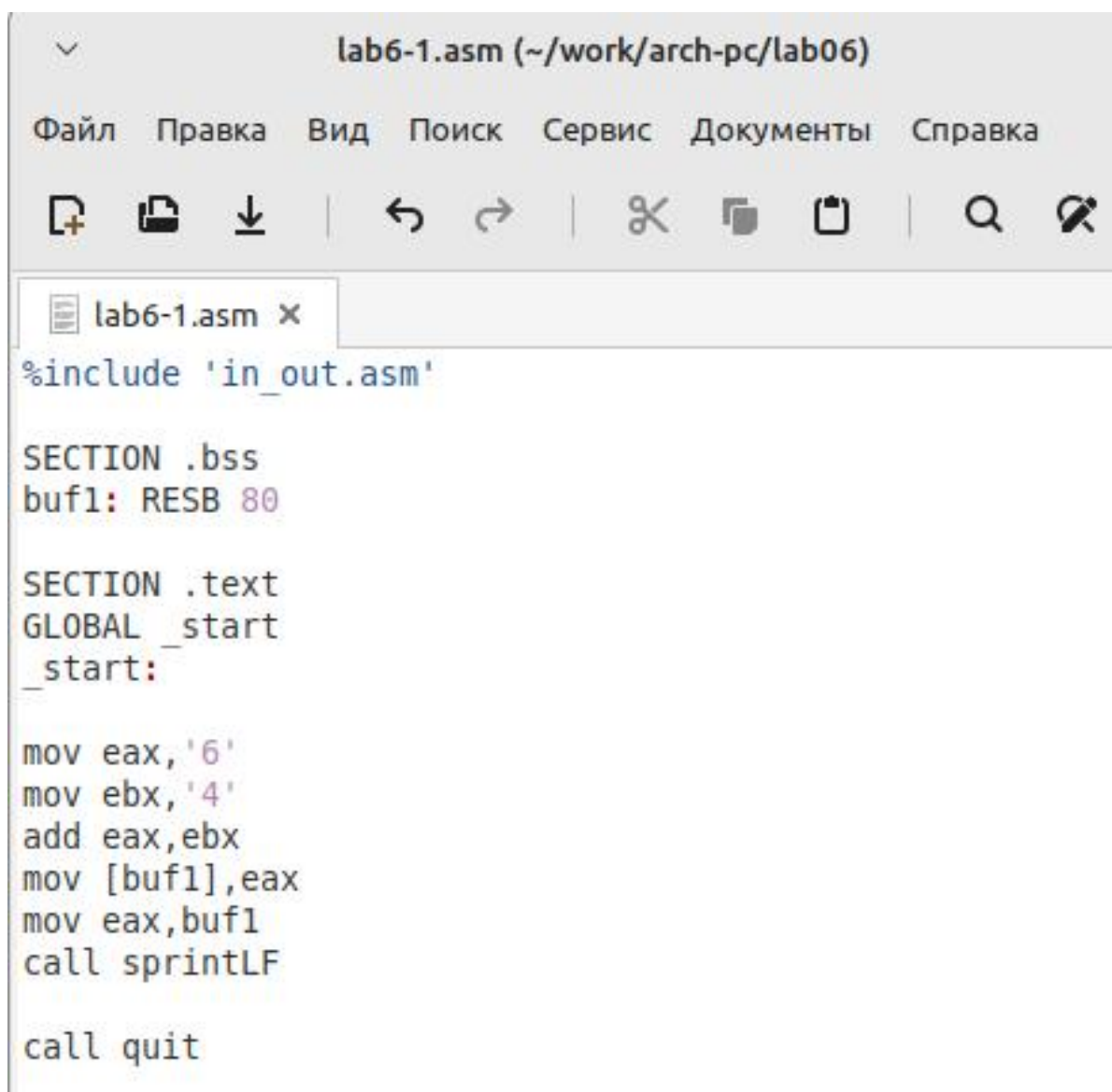
Создадим каталог для программ лаб. работы н.6, перейдем в него и создадим файл “lab6-1.asm” (рис. 4.1)



```
Терминал - dmlazarev@dmlazarev: ~/work/arch-pc/lab06
Файл  Правка  Вид  Терминал  Вкладки  Справка
dmlazarev@dmlazarev:~$ mkdir ~/work/
arch-pc/ study/
dmlazarev@dmlazarev:~$ mkdir ~/work/arch-pc/lab06
dmlazarev@dmlazarev:~$ cd ~/work/arch-pc/lab06
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ touch lab6-1.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание файла в каталоге

Введем в созданный файл текст программы из предложенного нам листинга 6.1(рис. 4.2)



The image shows a text editor window titled "lab6-1.asm (~/.work/arch-pc/lab06)". The menu bar includes "Файл", "Правка", "Вид", "Поиск", "Сервис", "Документы", and "Справка". The toolbar contains icons for file operations (new, open, save, print), editing (undo, redo, cut, copy, paste), and search. A single tab labeled "lab6-1.asm" is open. The code content is as follows:

```
%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. 4.2: Код программы в файле

Создадим исполняемый файл и запустим его, предварительно скопировав из предыдущей лаб. работы файл "in\_out.asm" для корректной работы (рис. 4.3)



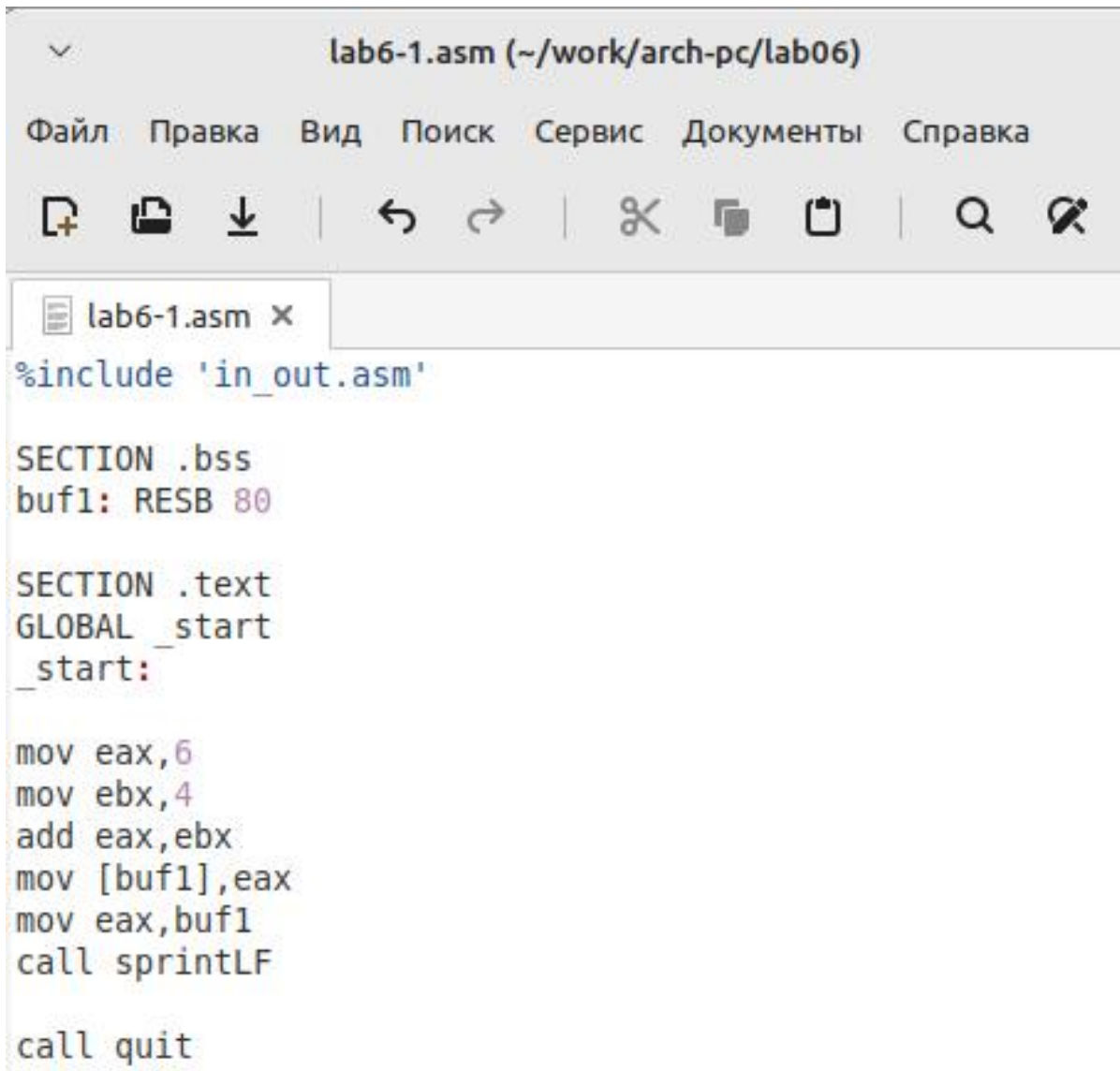
```

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

```

Рис. 4.3: Преобразование в исполняемый файл

Далее изменим часть строк в исходном файле. (рис. 4.5)



```

lab6-1.asm (~/.work/arch-pc/lab06)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Icons]
lab6-1.asm 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

```

Рис. 4.4: Измененный код программы

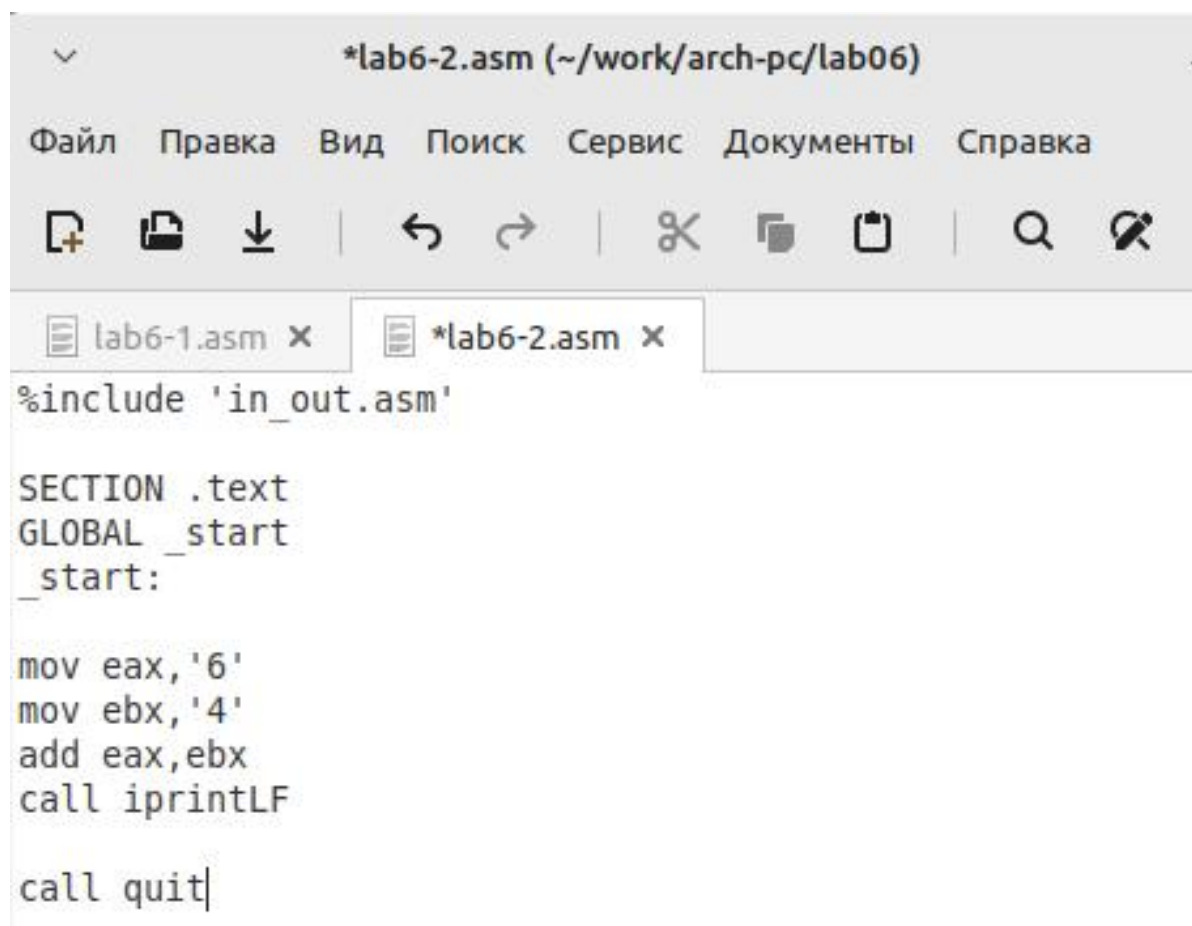
Преобразуем в исполняемый файл и проверим правильность выполнения. (рис. ??)

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

dmlazarev@dmlazarev:~/work/arch-pc/lab06$
```

Рис. 4.5: Преобразование измененного файла

Предварительно создав файл “lab6-2.asm” в нашем каталоге, вставим в него предложенный нам листинг 6.2 (рис. 4.6)



```
*lab6-2.asm (~/.work/arch-pc/lab06)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Icons]
lab6-1.asm x  *lab6-2.asm x
%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit|
```

Рис. 4.6: Проверка наличия текста

Преобразуем файл в исполняемый и проверим правильность выполнения. (рис. 4.7)

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

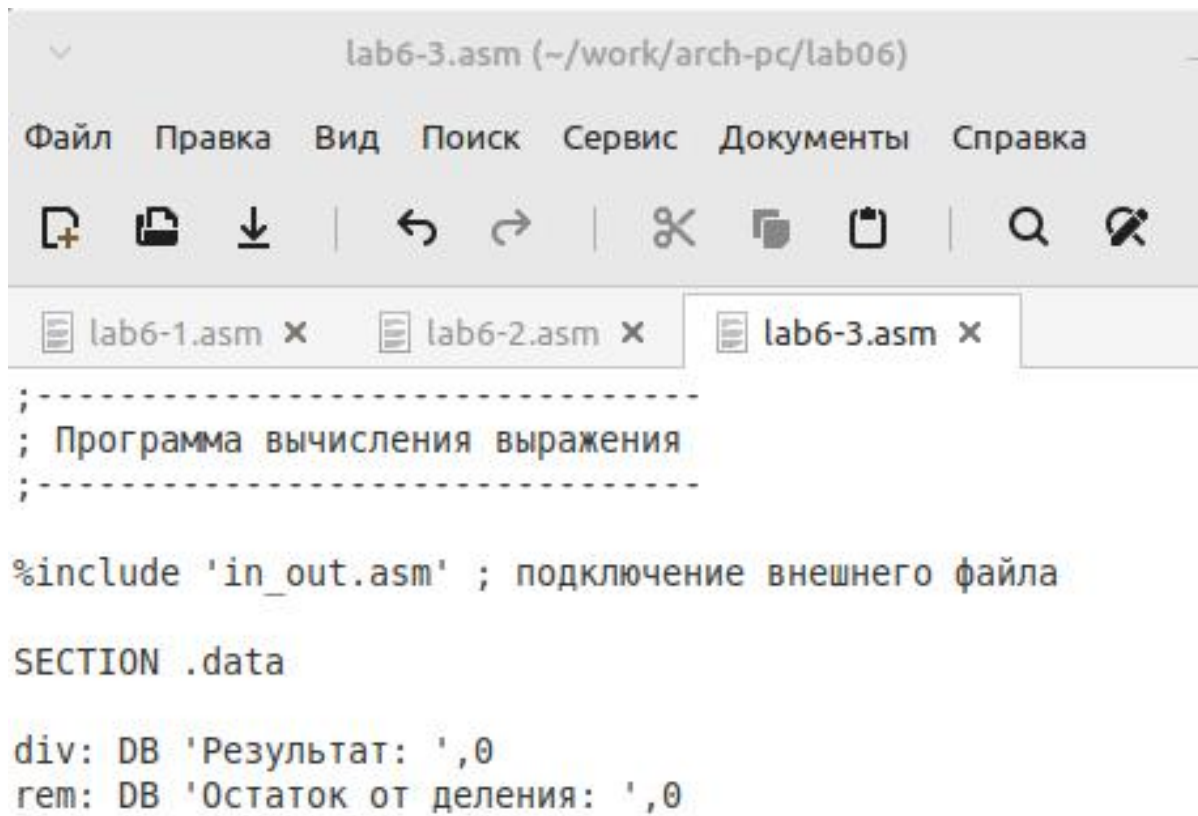
Рис. 4.7: Преобразование файла

Изменим часть строк в коде программы, преобразуем в исполняемый файл и проверим правильность выполнения. (рис. 4.8)

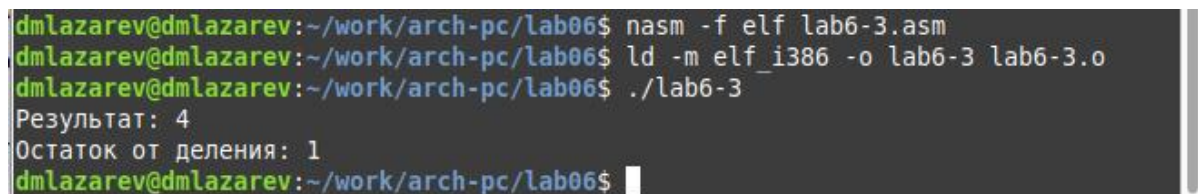
```
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ ./lab6-2
10
dmlazarev@dmlazarev:~/work/arch-pc/lab06$
```

Рис. 4.8: Преобразование исправленного файла

Создадим файл “lab6-3.asm” в каталоге и вставим в него предложенный нам листинг 6.3 (рис. 4.9)



Преобразуем его в исполняемый файл и проверим правильность выполнения (рис. 4.10)



Исправим листинг 6.3 так, чтобы выполнялась функция  $(4*6+2)/5$  (рис. 4.11)

```

;-----
; 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

```

Рис. 4.11: Исправленный файл

Преобразуем файл в исполняемый и проверим правильность работы (рис. 4.12)

```

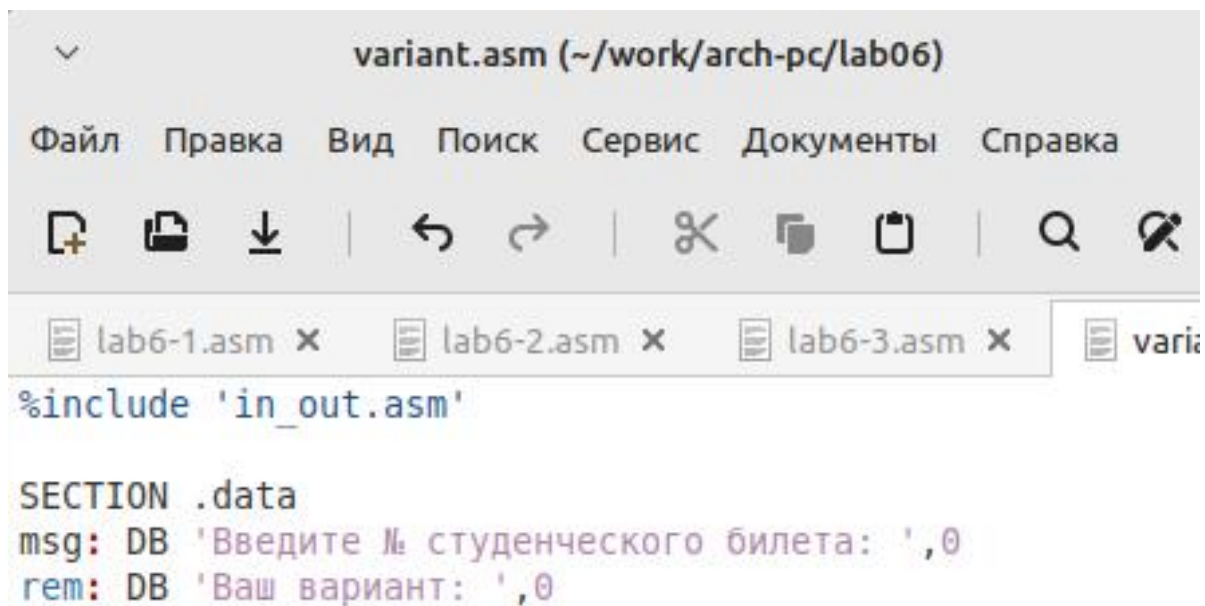
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
dmlazarev@dmlazarev:~/work/arch-pc/lab06$

```

Рис. 4.12: Преобразование файла в исполняемый



Создадим файл “variant.asm” и вставим в него предложенный нам листинг 6.4 (рис. 4.13)

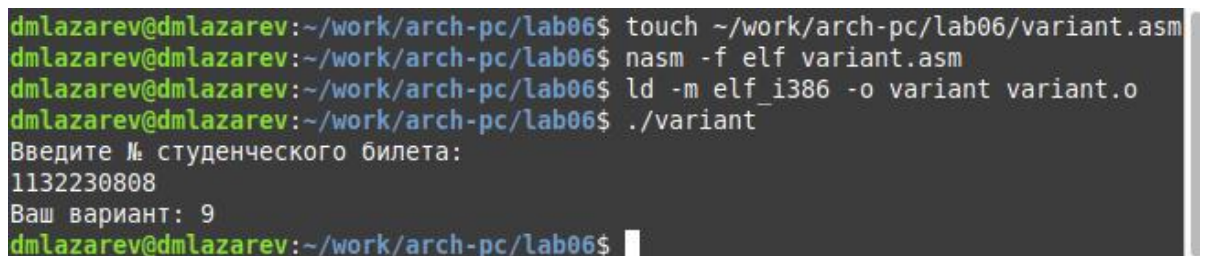


```
variant.asm (~/.work/arch-pc/lab06)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Icons]
lab6-1.asm x  lab6-2.asm x  lab6-3.asm x  variant.asm
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
```

Рис. 4.13: Код программы в файле

Преобразуем созданный нами файл в исполняемый и проверим работоспособность (рис. 4.14)



```
dmLazarev@dmLazarev:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
dmLazarev@dmLazarev:~/work/arch-pc/lab06$ nasm -f elf variant.asm
dmLazarev@dmLazarev:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
dmLazarev@dmLazarev:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132230808
Ваш вариант: 9
dmLazarev@dmLazarev:~/work/arch-pc/lab06$
```

Рис. 4.14: Преобразование файла и проверка работы

Ответим на вопросы, поставленные после листинга 6.4 (рис. 4.15)

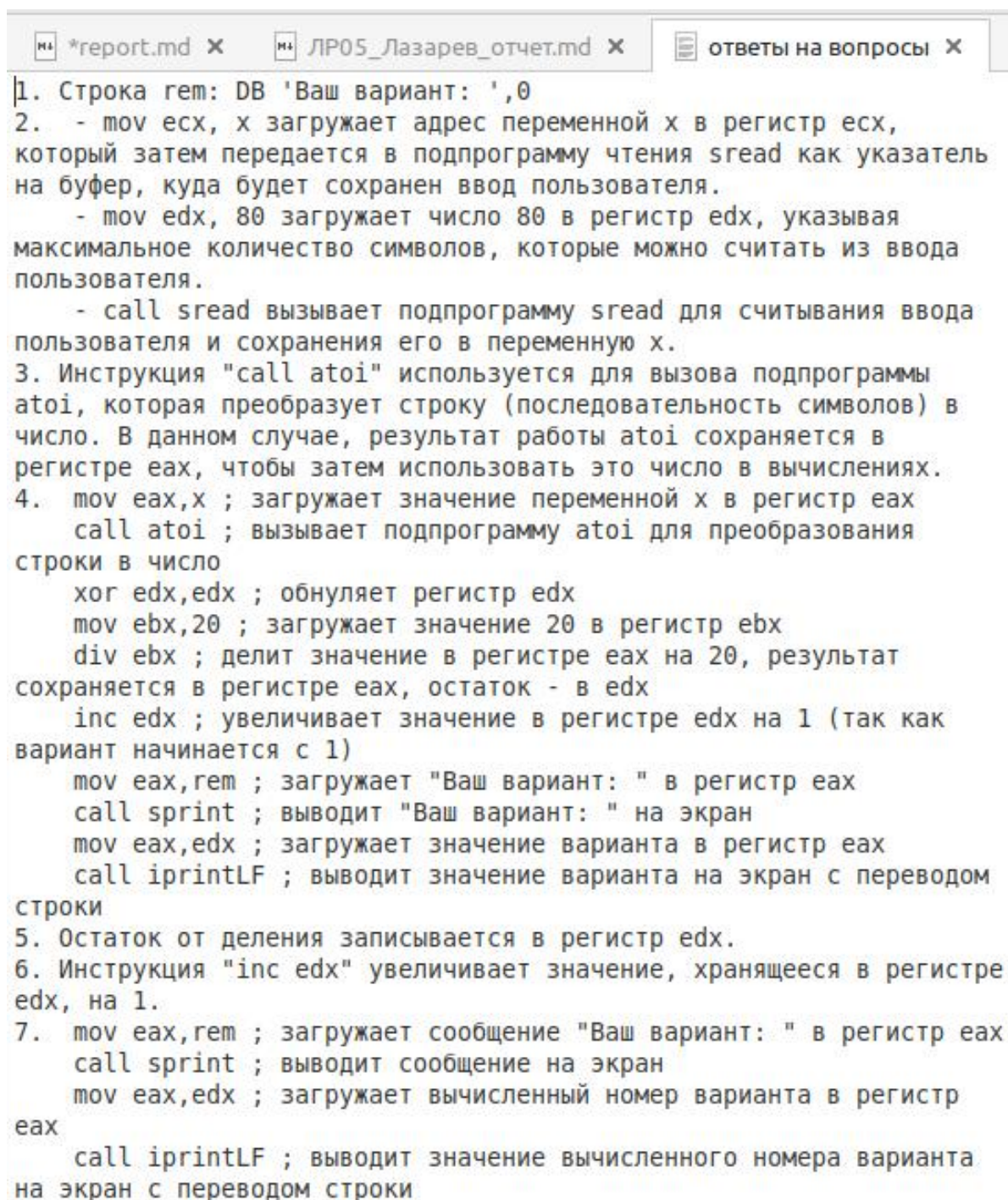


Рис. 4.15: Ответы на вопросы

## 5 Выполнение самостоятельной работы

Основываясь на результате файла “variant.asm” выберем функцию из таблицы, в нашем случае это -  $10+(31*x-5)$ . Самостоятельно напишем код программы, который будет проводить вычисления относительно введенного  $x$ . (рис. 5.1)



```

#include 'in_out.asm'

SECTION .data
    task: DB 'f(x)=10+(31*x-5)',0
    vvod: DB 'Введите x: ',0
    ans: DB 'ответ: ',0

SECTION .bss
    x: RESB 80

SECTION .text
GLOBAL _start
_start:

    mov eax,task
    call sprintfLF

    mov eax,vvod
    call sprintf
    mov ecx,x
    mov edx,80
    call sread

    mov eax,x
    call atoi

    mov ebx,31
    mul ebx
    sub eax,5
    add eax,10
    mov ebx,eax

    mov eax, ans
    call sprintf

    mov eax,ebx
    call iprintLF

    call quit

```

Рис. 5.1: Код программы для вычисления функции

Преобразуем файл в исполняемый и проверим правильность выполнения работы относительно заданных нам  $x$  (3 и 1) (рис. 5.2)

```
dmlazarev@dmlazarev:~$ cd ~/work/arch-pc/lab06
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ nasm -f elf sr.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ ld -m elf_i386 -o sr sr.o
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ ./sr
f(x)=10+(31*x-5)
Введите x: 3
ответ: 98
dmlazarev@dmlazarev:~/work/arch-pc/lab06$ ./sr
f(x)=10+(31*x-5)
Введите x: 1
ответ: 36
dmlazarev@dmlazarev:~/work/arch-pc/lab06$
```

Рис. 5.2: Проверка работоспособности

## **6 Выводы**

В ходе лабораторной работы мы освоили арифметические операции на языке NASM.