

Отчёт по лабораторной работе №6

Архитектура компьютеров и операционные системы.

Брыляков Никита Евгеньевич

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Начало работы	8
4.2	Выполнение арифметических операций в NASM	12
4.2.1	Ответы на вопросы	15
4.3	Выполнение заданий для самостоятельной работы	16
5	Вывод	18
6	Список литературы	19

Список иллюстраций

4.1	Создание каталога и файла внутри	8
4.2	Просмотр каталога	8
4.3	Открытие и ввод программы	9
4.4	Создание и запуск	9
4.5	Замена кода в программе	9
4.6	Создание и запуск	10
4.7	Создание файла	10
4.8	Ввод программы	10
4.9	Создание и запуск	11
4.10	Замена кода в программе	11
4.11	Создание и запуск	11
4.12	Изменение программы	12
4.13	Создание и проверка	12
4.14	Создание файла	12
4.15	Ввод программы	13
4.16	Создание и запуск	13
4.17	Изменение программы	14
4.18	Создание и запуск	14
4.19	Создание файла	14
4.20	Ввод программы	15
4.21	Создание и запуск	15
4.22	Создание файла	16
4.23	Ввод программы	17
4.24	Создание и запуск	17

1 Цель 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. 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. 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. Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной (см. Приложение.), а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться. 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.

4 Выполнение лабораторной работы

4.1 Начало работы

Создаю каталог для работы, перехожу в него и создаю файл lab6-1.asm (рис. [4.1]).

```
(nebrihlyakov@nebrihlyakov)-[~]  
$ mkdir ~/work/arch-pc/lab06  
  
(nebrihlyakov@nebrihlyakov)-[~]  
$ cd ~/work/arch-pc/lab06  
  
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]  
$ touch lab6-1.asm
```

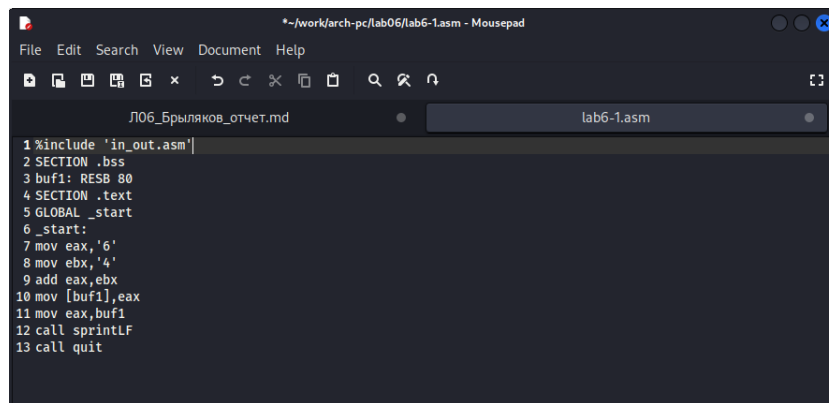
Рис. 4.1: Создание каталога и файла внутри

Вручную переношу файл in_out.asm и проверяю его наличие в каталоге (рис. [4.2]).

```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]  
$ ls  
in_out.asm  lab6-1.asm
```

Рис. 4.2: Просмотр каталога

Открываю файл lab6-1.asm и ввожу нужную программу (рис. [4.3]).

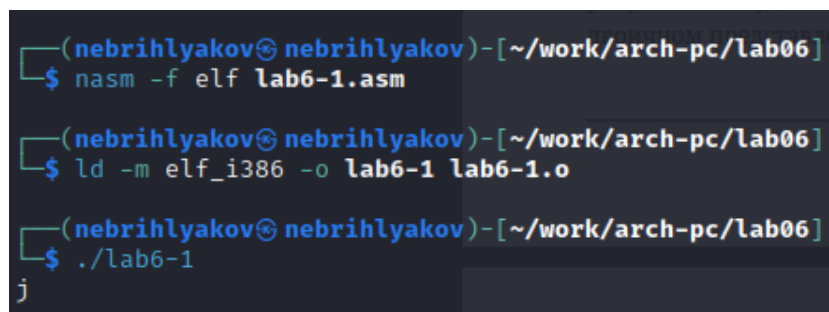


The screenshot shows a text editor window titled "*/work/arch-pc/lab06/lab6-1.asm - Mousepad". The menu bar includes File, Edit, Search, View, Document, and Help. The toolbar contains icons for file operations and editing. The file list shows "Л06_Брыляков_отчет.md" and "lab6-1.asm". The code in the editor is as follows:

```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintf
13 call quit
```

Рис. 4.3: Открытие и ввод программы

Создаю исполняемый файл и запускаю его (рис. [4.4]).



The screenshot shows a terminal window with the following commands and output:

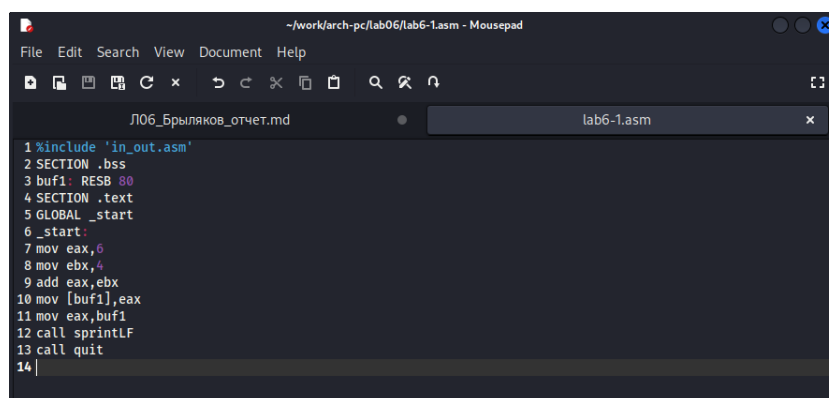
```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ nasm -f elf lab6-1.asm

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ld -m elf_i386 -o lab6-1 lab6-1.o

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ./lab6-1
j
```

Рис. 4.4: Создание и запуск

Заменяю некоторые строки в программе (рис. [4.5]).



The screenshot shows the same text editor window as in Figure 4.3, but with some modifications to the assembly code. The code is as follows:

```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, 6
8 mov ebx, 4
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintf
13 call quit
14 |
```

Рис. 4.5: Замена кода в программе

Создаю новый исполняемый файл и запускаю его. Код 10 соответствует символу переноса строки. Он не выводится на экран(рис. [4.6]).

```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ nasm -f elf lab6-1.asm

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ld -m elf_i386 -o lab6-1 lab6-1.o

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ./lab6-1
```

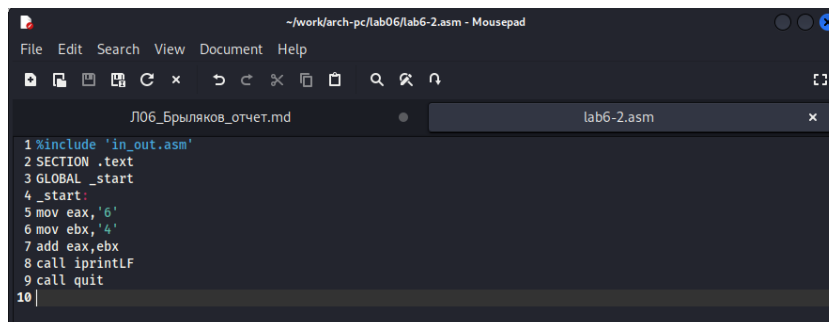
Рис. 4.6: Создание и запуск

Создаю новый файл в каталоге (рис. [4.7]).

```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ touch ~/work/arch-pc/lab06/lab6-2.asm
```

Рис. 4.7: Создание файла

Ввожу в него текст программы (рис. [4.8]).



```
~/work/arch-pc/lab06/lab6-2.asm - Mousepad
File Edit Search View Document Help
ЛЮБ_Брыляков_отчет.md lab6-2.asm x
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,'6'
6 mov ebx,'4'
7 add eax,ebx
8 call iprintLF
9 call quit
10|
```

Рис. 4.8: Ввод программы

Создаю исполняемый файл и запускаю его. В данном случае программа производит сложение кодов символов и выводит число, а не символ, кодом которого является это число (рис. [4.9]).

```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ nasm -f elf lab6-2.asm

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ld -m elf_i386 -o lab6-2 lab6-2.o

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ./lab6-2
106
```

Рис. 4.9: Создание и запуск

Заменяю некоторые строки в программе (рис. [4.10]).

```
~/work/arch-pc/lab06/lab6-2.asm - Mousepad
File Edit Search View Document Help
Л06_Брыляков_отчет.md lab6-2.asm

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintLF
9 call quit
10
```

Рис. 4.10: Замена кода в программе

Создаю новый исполняемый файл и запускаю его. Теперь программа складывает сами числа (рис. [4.11]).

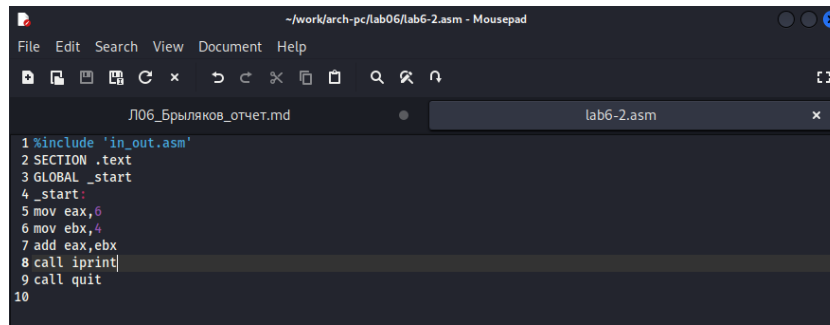
```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ nasm -f elf lab6-2.asm

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ld -m elf_i386 -o lab6-2 lab6-2.o

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ./lab6-2
10
```

Рис. 4.11: Создание и запуск

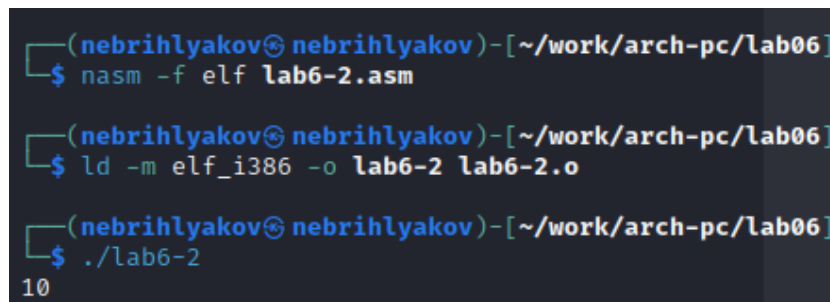
Заменяю функцию iprintLF на iprint (рис. [4.12]).



```
~/work/arch-pc/lab06/lab6-2.asm - Mousepad
File Edit Search View Document Help
Л06_Брыляков_отчет.md lab6-2.asm x
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintf
9 call quit
10
```

Рис. 4.12: Изменение программы

Создаю исполняемый файл и запускаю его. Вывод не изменился. С `iprintLF` символ перевода строки не отображался, а `iprint`, в отличие от `iprintLF`, не производит перевод строки (рис. [4.13]).



```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ nasm -f elf lab6-2.asm

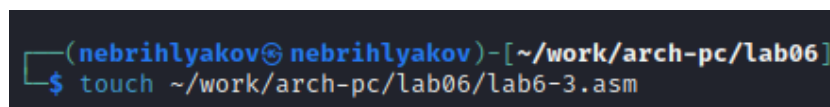
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ld -m elf_i386 -o lab6-2 lab6-2.o

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ./lab6-2
10
```

Рис. 4.13: Создание и проверка

4.2 Выполнение арифметических операций в NASM

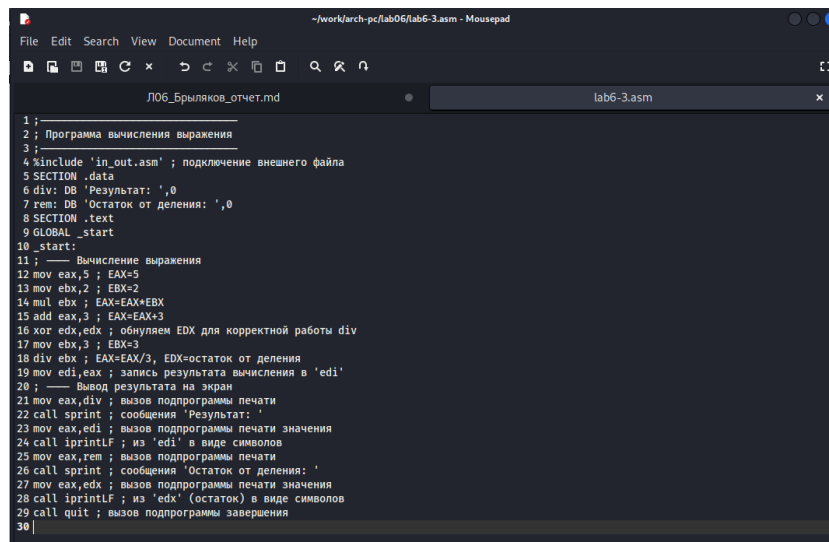
Создаю новый файл в каталоге (рис. [4.14]).



```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ touch ~/work/arch-pc/lab06/lab6-3.asm
```

Рис. 4.14: Создание файла

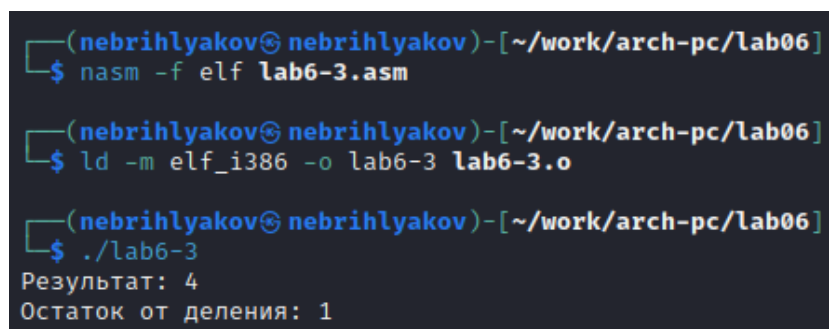
Ввожу в него текст программы. (рис. [4.15]).



```
1;  
2; Программа вычисления выражения  
3;  
4 %include 'in_out.asm' ; подключение внешнего файла  
5 SECTION .data  
6 div: DB 'Результат: ',0  
7 rem: DB 'Остаток от деления: ',0  
8 SECTION .text  
9 GLOBAL _start  
10 _start:  
11 ; ——— Вычисление выражения  
12 mov eax,5 ; EAX=5  
13 mov ebx,2 ; EBX=2  
14 mul ebx ; EAX=EAX*EBX  
15 add eax,3 ; EAX=EAX+3  
16 xor edx,edx ; обнуляем EDX для корректной работы div  
17 mov ebx,3 ; EBX=3  
18 div ebx ; EAX=EAX/3, EDX=остаток от деления  
19 mov edi,eax ; запись результата вычисления в 'edi'  
20 ; ——— Вывод результата на экран  
21 mov eax,div ; вызов подпрограммы печати  
22 call sprint ; сообщения 'Результат: '  
23 mov eax,edi ; вызов подпрограммы печати значения  
24 call iprintf ; из 'edi' в виде символов  
25 mov eax,rem ; вызов подпрограммы печати  
26 call sprint ; сообщения 'Остаток от деления: '  
27 mov eax,edx ; вызов подпрограммы печати значения  
28 call iprintf ; из 'edx' (остаток) в виде символов  
29 call quit ; вызов подпрограммы завершения  
30 |
```

Рис. 4.15: Ввод программы

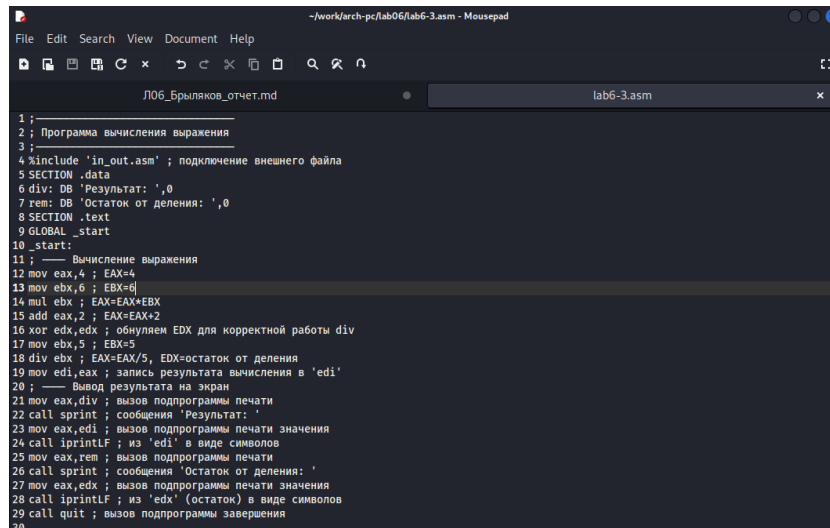
Создаю исполняемый файл и запускаю его (рис. [4.16]).



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

Рис. 4.16: Создание и запуск

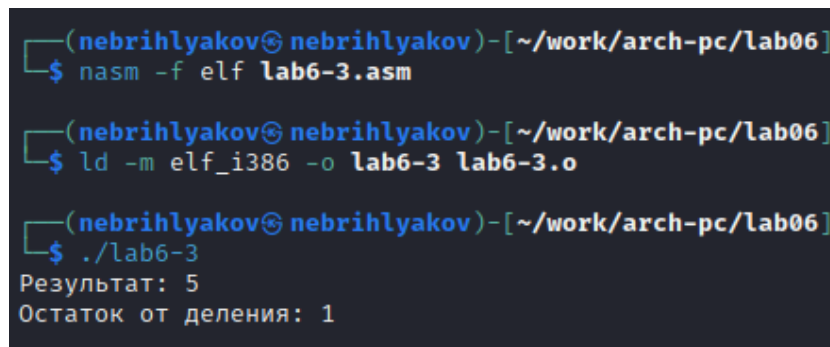
Изменяю текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$. (рис. [4.17]).



```
1;  
2; Программа вычисления выражения  
3;  
4 %include 'in_out.asm' ; подключение внешнего файла  
5 SECTION .data  
6 div: DB 'Результат: ',0  
7 rem: DB 'Остаток от деления: ',0  
8 SECTION .text  
9 GLOBAL _start  
10 _start:  
11 ; ——— Вычисление выражения  
12 mov eax,4 ; EAX=4  
13 mov ebx,6 ; EBX=6  
14 mul ebx ; EAX=EAX*EBX  
15 add eax,2 ; EAX=EAX+2  
16 xor edx,edx ; обнуляем EDX для корректной работы div  
17 mov ebx,5 ; EBX=5  
18 div ebx ; EAX=EAX/5, EDX=остаток от деления  
19 mov edi,eax ; запись результата вычисления в 'edi'  
20 ; ——— Вывод результата на экран  
21 mov eax,div ; вызов подпрограммы печати  
22 call sprint ; сообщения 'Результат: '  
23 mov eax,edi ; вызов подпрограммы печати значения  
24 call iprintLF ; из 'edi' в виде символов  
25 mov eax,rem ; вызов подпрограммы печати  
26 call sprint ; сообщения 'Остаток от деления: '  
27 mov eax,edx ; вызов подпрограммы печати значения  
28 call iprintLF ; из 'edx' (остаток) в виде символов  
29 call quit ; вызов подпрограммы завершения  
30
```

Рис. 4.17: Изменение программы

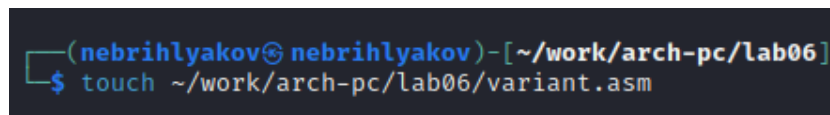
Создаю новый исполняемый файл и запускаю его. Всё верно (рис. [4.18]).



```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]  
$ nasm -f elf lab6-3.asm  
  
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]  
$ ld -m elf_i386 -o lab6-3 lab6-3.o  
  
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]  
$ ./lab6-3  
Результат: 5  
Остаток от деления: 1
```

Рис. 4.18: Создание и запуск

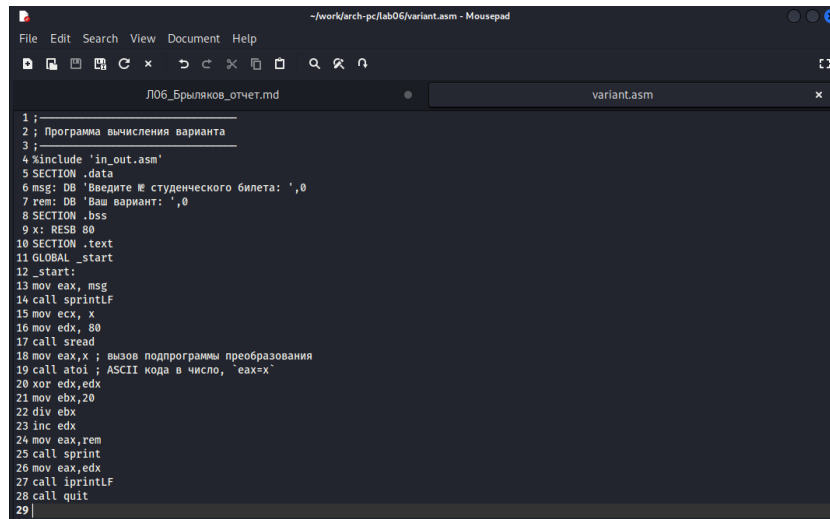
Создаю файл variant.asm в каталоге (рис. [4.19]).



```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]  
$ touch ~/work/arch-pc/lab06/variant.asm
```

Рис. 4.19: Создание файла

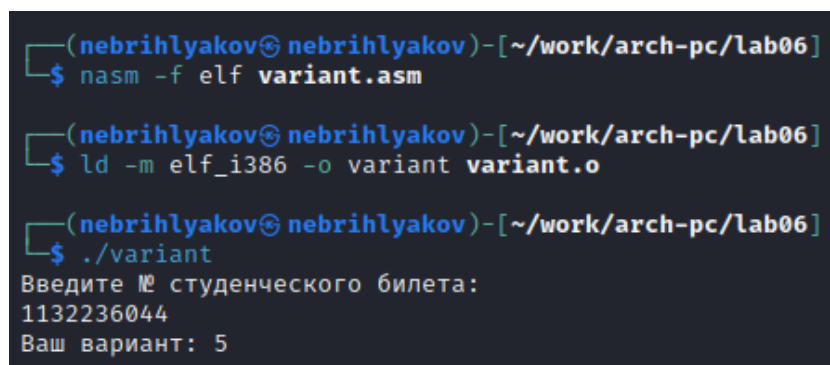
Ввожу в него текст программы для вычисления варианта. (рис. [4.20]).



```
1 ;
2 ; Программа вычисления варианта
3 ;
4 %include 'in_out.asm'
5 SECTION .data
6 msg: DB "Введите № студенческого билета: ",0
7 rem: DB "Ваш вариант: ",0
8 SECTION .bss
9 x: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprintf
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x ; вызов подпрограммы преобразования
19 call atoi ; ASCII кода в число, 'eax=x'
20 xor edx, edx
21 mov ebx, 20
22 div ebx
23 inc edx
24 mov eax, rem
25 call sprint
26 mov eax, edx
27 call iprintf
28 call quit
29 |
```

Рис. 4.20: Ввод программы

Создаю исполняемый файл и запускаю его. Мой вариант - 5. (рис. [??]).



```
(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ nasm -f elf variant.asm

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ld -m elf_i386 -o variant variant.o

(nebrihlyakov@nebrihlyakov)-[~/work/arch-pc/lab06]
$ ./variant
Введите № студенческого билета:
1132236044
Ваш вариант: 5
```

Рис. 4.21: Создание и запуск

4.2.1 Ответы на вопросы

1. За вывод на экран сообщения 'Ваш вариант:' отвечают строки "mov eax,rem" и "call sprint"
2. Инструкция "mov ecx, x" чтобы положить адрес строки, которая вводится, в регистр. "mov edx, 80" в регистр запись длины вводимой строки. "call sread" вывод подпрограммы, которая обеспечивает ввод с клавиатуры.

3. “call atoi” используется для вывода подпрограммы, которая производит преобразование ascii кода символа в целое число и записывает результат в регистр eax.
4. За вычисление варианта отвечают эти строки: “xor edx, edx”, “mov ebx, 20”, “div ebx”, “inc edx”
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 результата вычислений отвечают эти строки: “mov eax, edx”, “call iprintLF”.

4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm в каталоге (рис. [4.22]).

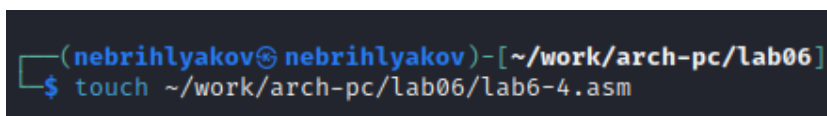
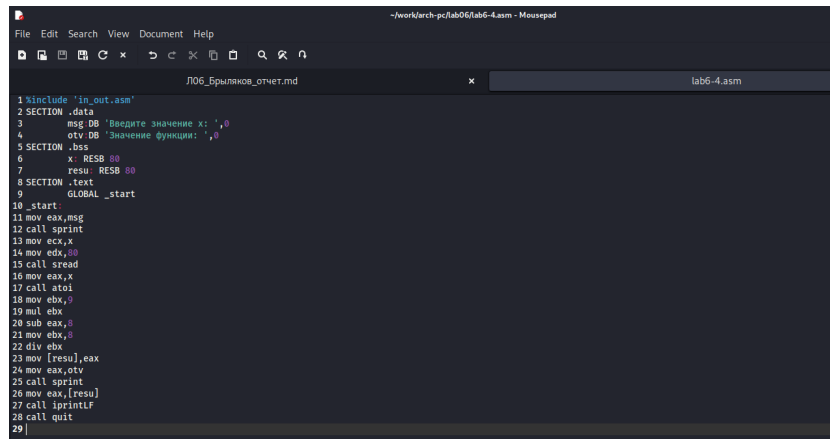


Рис. 4.22: Создание файла

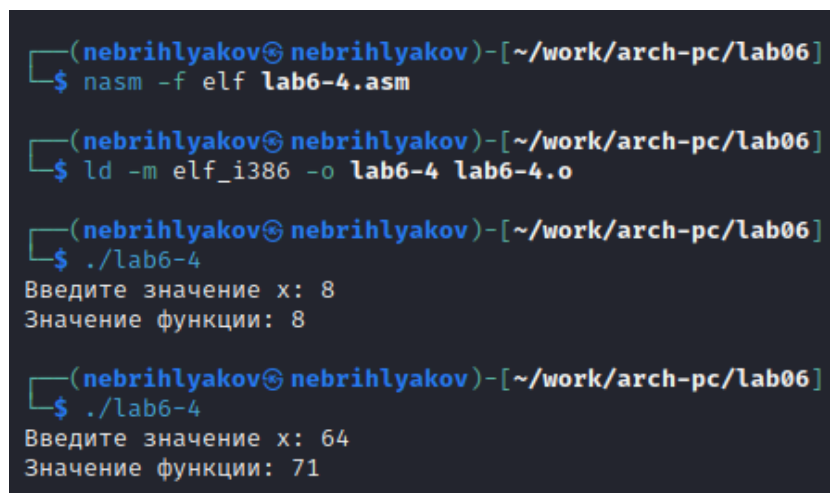
Ввожу в него текст программы для вычисления значения выражения моего варианта $(9x - 8)/8$. (рис. [??]).



```
1 include "in_out.asm"
2 section .data
3     msg db "Введите значение x: ",0
4     otv db "Значение функции: ",0
5 section .bss
6     x: resb 80
7     resu: resb 80
8 section .text
9     GLOBAL _start
10 _start:
11     mov eax,msg
12     call sprintf
13     mov ecx,x
14     mov edx,0
15     call _read
16     mov eax,x
17     call atoi
18     mov ebx,e
19     mul ebx
20     sub eax,0
21     mov ebx,0
22     div ebx
23     mov [resu],eax
24     mov eax,otv
25     call sprintf
26     mov eax,[resu]
27     call _printf
28     call _quit
29
```

Рис. 4.23: Ввод программы

Создаю исполняемый файл и запускаю его. Проверяю его для двух значений x : 8 и 64. Я также проверил вручную и результаты оказались верны. (рис. [??]).



```
(nebrihlyakov@nebrihlyakov)~/work/arch-pc/lab06
$ nasm -f elf lab6-4.asm

(nebrihlyakov@nebrihlyakov)~/work/arch-pc/lab06
$ ld -m elf_i386 -o lab6-4 lab6-4.o

(nebrihlyakov@nebrihlyakov)~/work/arch-pc/lab06
$ ./lab6-4
Введите значение x: 8
Значение функции: 8

(nebrihlyakov@nebrihlyakov)~/work/arch-pc/lab06
$ ./lab6-4
Введите значение x: 64
Значение функции: 71
```

Рис. 4.24: Создание и запуск

5 Вывод

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

6 Список литературы

- [illegible]