

# **Отчёт по лабораторной работе №6**

**Дисциплина: компьютерные науки и технологии программирования**

Мирзоян Ян Игоревич

# 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>8</b>
4.1	Символьные и численные данные в NASM . . . . .	8
4.2	Выполнение арифметических операций в NASM . . . . .	12
4.2.1	Ответы на вопросы . . . . .	15
4.3	Выполнение заданий для самостоятельной работы . . . . .	16
<b>5</b>	<b>Выводы</b>	<b>17</b>
	<b>Список литературы</b>	<b>18</b>

## Список иллюстраций

4.1	Создаю директорию, перехожу в нее и создаю необходимый файл .	8
4.2	Открываю файл и заполняю его в соответствии с указаниями . . .	8
4.3	Создаю и запускаю исполняемый файл . . . . .	9
4.4	Корректирую вышеописанный файл, создаю и запускаю исполняе- мый файл. Теперь вывелся символ с кодом 10, это символ перевода строки, он символ не отображается при выводе на экран. . . . .	9
4.5	Создаю новый файл и заполняю его в соответствии с указаниями. .	9
4.6	Создаю и запускаю исполняемый файл . . . . .	10
4.7	Корректирую вышеописанный файл, создаю и запускаю исполняе- мый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому выводится 10 10	10
4.8	Заменяю в тексте программы функцию <code>iprintLF</code> на <code>iprint</code> . Создаю и запускаю новый исполняемый файл. Вывод не изменился, пото- му что символ переноса строки не отображался, когда программа исполнялась с функцией <code>iprintLF</code> , а <code>iprint</code> не добавляет к выводу символ переноса строки, в отличие от <code>iprintLF</code> . . . . .	11
4.9	Создаю файл <code>lab6-3.asm</code> и заполняю его согласно инструкции . .	12
4.10	Создаю и запускаю исполняемый файл. Выведенный результат сов- пал с ожидаемым . . . . .	12
4.11	Корректирую вышеописанный файл, изменяя функцию, создаю и запускаю исполняемый файл. Подсчеты проведены корректно . .	13
4.12	Создаю файл <code>variant.asm</code> и заполняю его соответственно требова- ниям. Создаю и запускаю исполняемый файл. Ввожу номер студен- ческого билета и получаю свой вариант . . . . .	14
4.13	Название рисунка . . . . .	15
4.14	Создаю файл <code>lab6-4.asm</code> , начинаю заполнять для формулы варианта 3	16
4.15	Проверяю работоспособность программы для $x_1=4$ и $x_2=5$ из пункта 6.3, значения получены верные . . . . .	16

## Список таблиц

# 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.

## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

```
[yimirzoyan@fedora ~]$ mkdir ~/work/arch-pc/lab06  
[yimirzoyan@fedora ~]$ cd ~/work/arch-pc/lab06  
[yimirzoyan@fedora lab06]$ touch lab6-1.asm  
[yimirzoyan@fedora lab06]$
```

Рис. 4.1: Создаю директорию, перехожу в нее и создаю необходимый файл

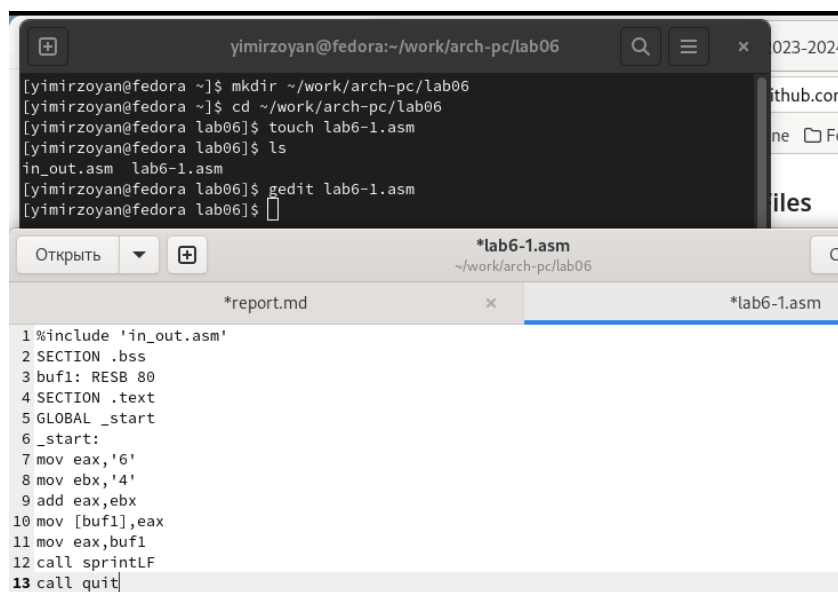


Рис. 4.2: Открываю файл и заполняю его в соответствии с указаниями

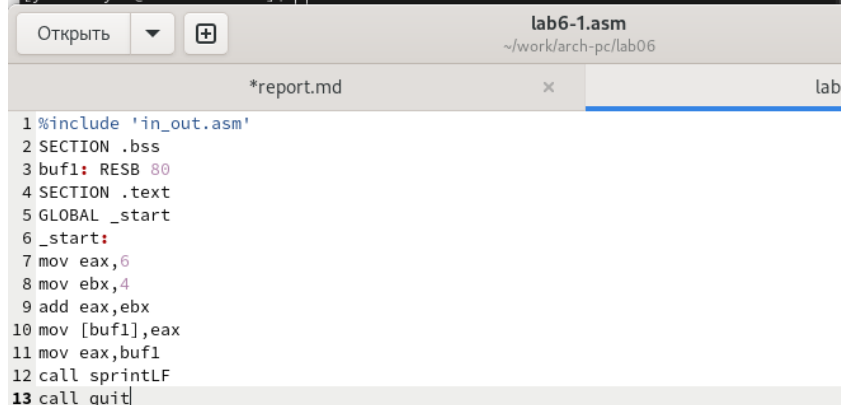


```
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-1.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[yimirzoyan@fedora lab06]$ ./lab6-1
j
[yimirzoyan@fedora lab06]$
```

Рис. 4.3: Создаю и запускаю исполняемый файл

```
[yimirzoyan@fedora lab06]$ gedit lab6-1.asm
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-1.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[yimirzoyan@fedora lab06]$ ./lab6-1

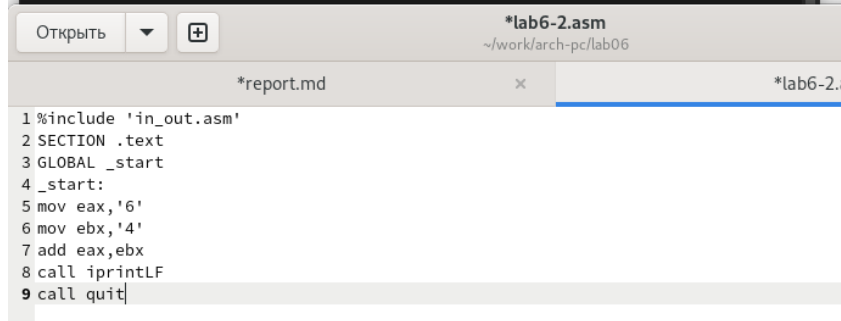
[yimirzoyan@fedora lab06]$ gedit lab6-1.asm
[yimirzoyan@fedora lab06]$
```



```
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 sprintLF
13 call quit
```

Рис. 4.4: Корректирую вышеописанный файл, создаю и запускаю исполняемый файл. Теперь вывелся символ с кодом 10, это символ перевода строки, он символ не отображается при выводе на экран.

```
[yimirzoyan@fedora lab06]$ touch ~/work/arch-pc/lab06/lab6-2.asm
[yimirzoyan@fedora lab06]$ gedit lab6-2.asm
[yimirzoyan@fedora lab06]$
```



```
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
```

Рис. 4.5: Создаю новый файл и заполняю его в соответствии с указаниями.

```
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-2.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[yimirzoyan@fedora lab06]$ ./lab6-2
106
```

Рис. 4.6: Создаю и запускаю исполняемый файл

```
[yimirzoyan@fedora lab06]$ gedit lab6-2.asm
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-2.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[yimirzoyan@fedora lab06]$ ./lab6-2
10
[yimirzoyan@fedora lab06]$ gedit lab6-2.asm
[yimirzoyan@fedora lab06]$
```

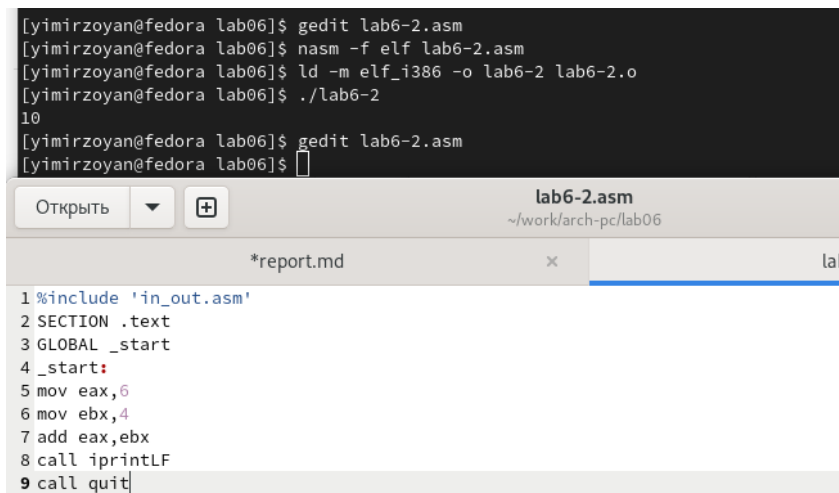
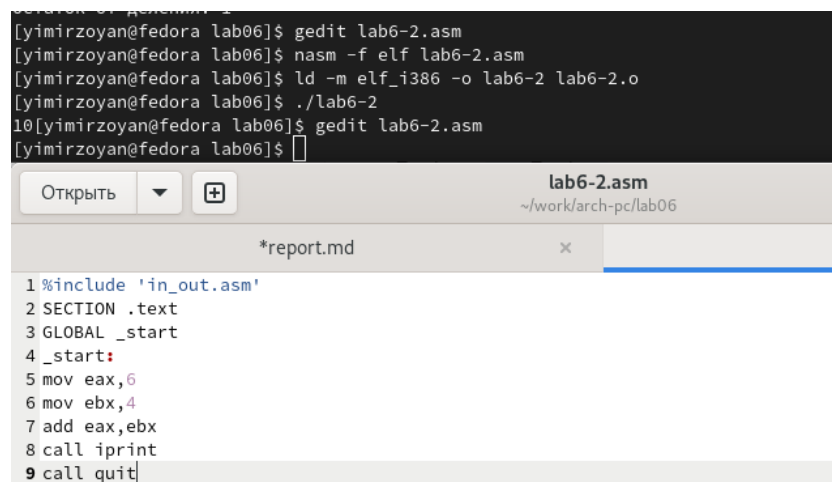


Рис. 4.7: Корректирую вышеописанный файл, создаю и запускаю исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому выводится 10

```
[yimirzoyan@fedora lab06]$ gedit lab6-2.asm
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-2.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[yimirzoyan@fedora lab06]$ ./lab6-2
10[yimirzoyan@fedora lab06]$ gedit lab6-2.asm
[yimirzoyan@fedora lab06]$
```



```
lab6-2.asm
~/work/arch-pc/lab06

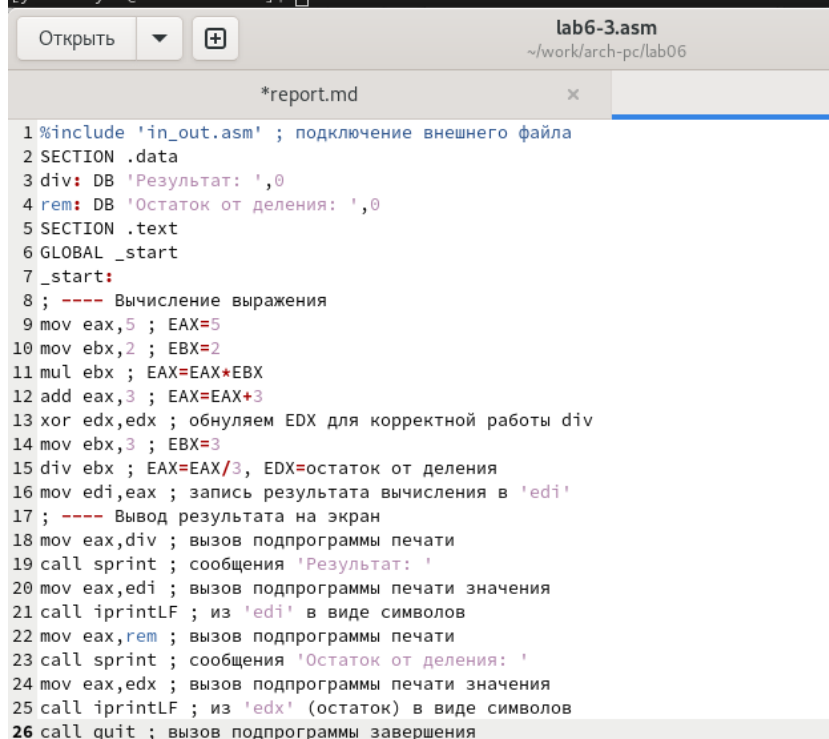
*report.md x

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 iprint
9 call quit
```

Рис. 4.8: Заменяю в тексте программы функцию `iprintLF` на `iprint`. 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`.

## 4.2 Выполнение арифметических операций в NASM

```
[yimirzoyan@fedora lab06]$ touch ~/work/arch-pc/lab06/lab6-3.asm
[yimirzoyan@fedora lab06]$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
[yimirzoyan@fedora lab06]$ gedit lab6-3.asm
[yimirzoyan@fedora lab06]$
```



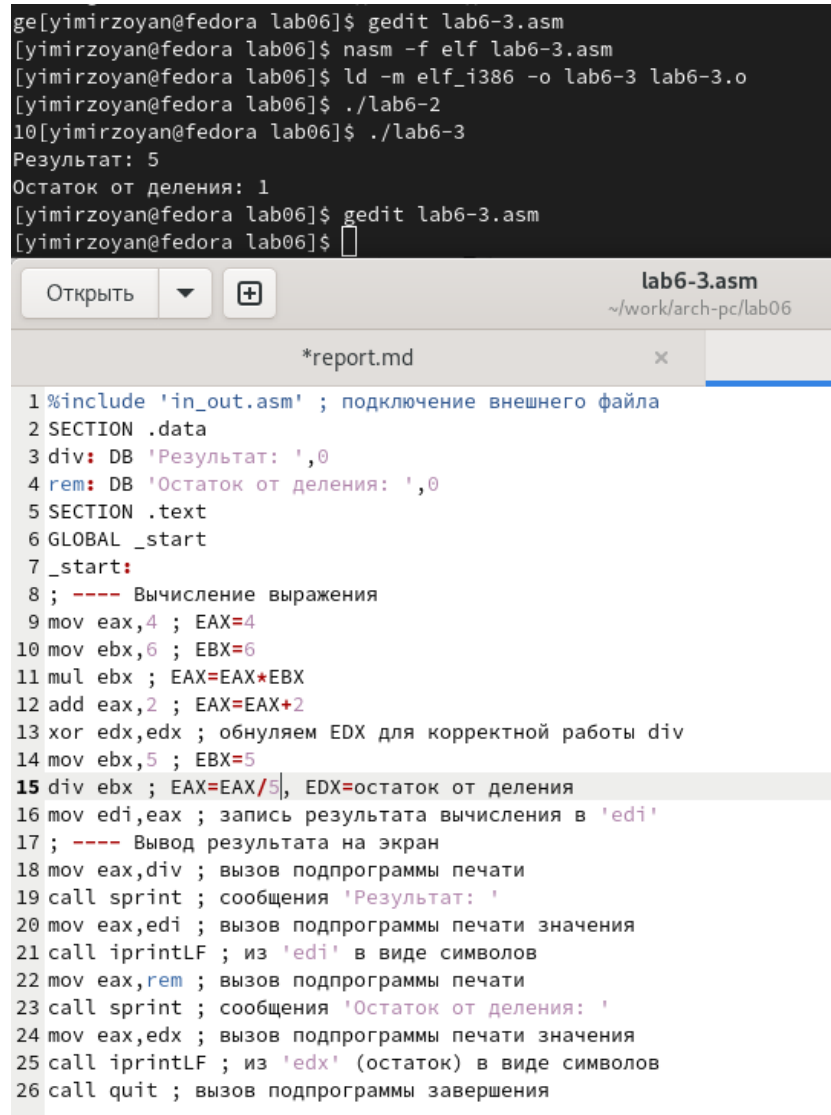
```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ; EAX=5
10 mov ebx,2 ; EBX=2
11 mul ebx ; EAX=EAX*EBX
12 add eax,3 ; EAX=EAX+3
13 xor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

Рис. 4.9: Создаю файл lab6-3.asm и заполняю его согласно инструкции

```
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-3.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[yimirzoyan@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.10: Создаю и запускаю исполняемый файл. Выведенный результат совпал с ожидаемым

```
ge[yimirzoyan@fedora lab06]$ gedit lab6-3.asm
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-3.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[yimirzoyan@fedora lab06]$ ./lab6-2
10[yimirzoyan@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[yimirzoyan@fedora lab06]$ gedit lab6-3.asm
[yimirzoyan@fedora lab06]$
```



The image shows a terminal window at the top with the following commands and output:

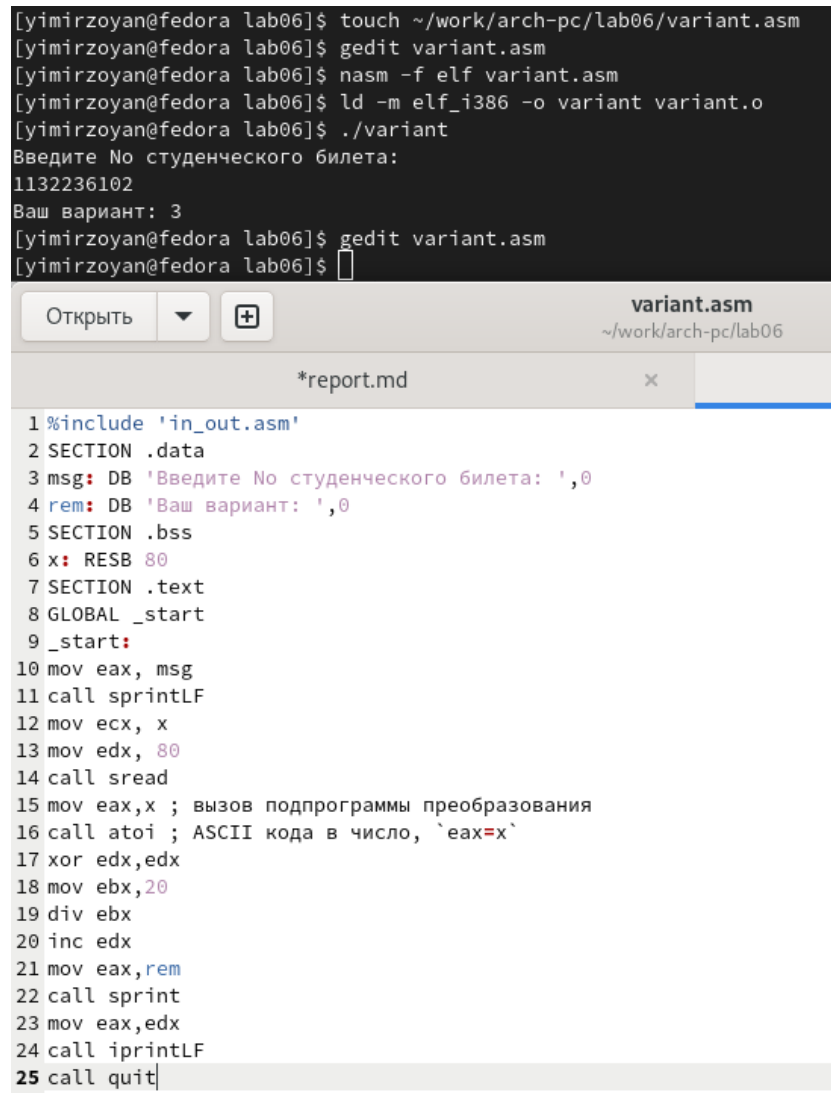
```
ge[yimirzoyan@fedora lab06]$ gedit lab6-3.asm
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-3.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[yimirzoyan@fedora lab06]$ ./lab6-2
10[yimirzoyan@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[yimirzoyan@fedora lab06]$ gedit lab6-3.asm
[yimirzoyan@fedora lab06]$
```

Below the terminal is a window titled "lab6-3.asm" with a file path of "~/.work/arch-pc/lab06". It contains the following assembly code:

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

Рис. 4.11: Корректирую вышеописанный файл, изменяя функцию, создаю и запускаю исполняемый файл. Подсчеты проведены корректно

```
[yimirzoyan@fedora lab06]$ touch ~/work/arch-pc/lab06/variant.asm
[yimirzoyan@fedora lab06]$ gedit variant.asm
[yimirzoyan@fedora lab06]$ nasm -f elf variant.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[yimirzoyan@fedora lab06]$ ./variant
Введите No студенческого билета:
1132236102
Ваш вариант: 3
[yimirzoyan@fedora lab06]$ gedit variant.asm
[yimirzoyan@fedora lab06]$
```



The image shows a terminal window and a code editor. The terminal window displays the commands used to create and run the assembly program. The code editor shows the assembly code for variant.asm, which includes instructions for reading input, converting it to a number, and calculating the variant.

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

Рис. 4.12: Создаю файл variant.asm и заполняю его соответственно требованиям.  
Создаю и запускаю исполняемый файл. Ввожу номер студенческого билета и получаю свой вариант

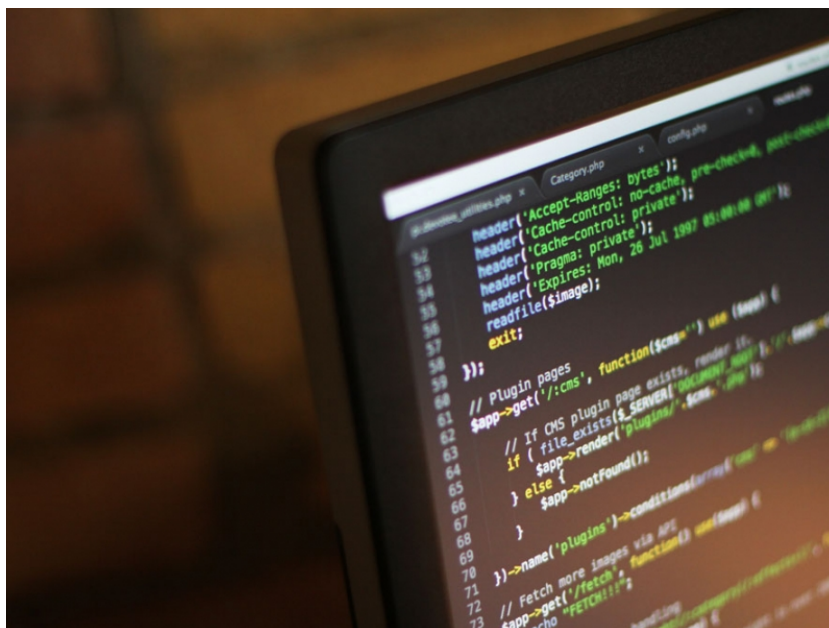


Рис. 4.13: Название рисунка

#### 4.2.1 Ответы на вопросы

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` `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
3. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-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

```

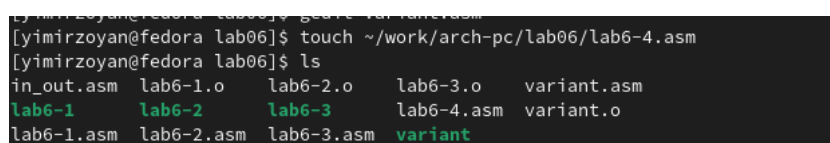
5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx увеличивает значение регистра edx на 1
7. За вывод на экран результатов вычислений отвечают строки:

```

mov eax,edx
call iprintLF

```

## 4.3 Выполнение заданий для самостоятельной работы

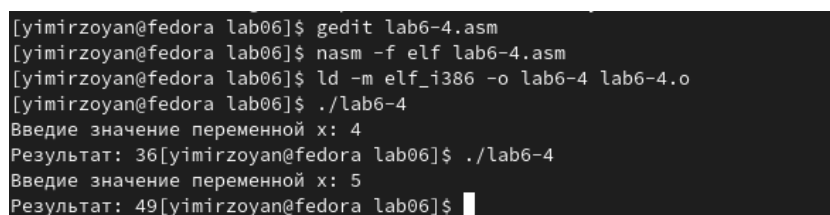


```

[yimirzoyan@fedora lab06]$ touch ~/work/arch-pc/lab06/lab6-4.asm
[yimirzoyan@fedora lab06]$ ls
in_out.asm  lab6-1.o    lab6-2.o    lab6-3.o    variant.asm
lab6-1      lab6-2      lab6-3      lab6-4.asm  variant.o
lab6-1.asm  lab6-2.asm  lab6-3.asm  variant

```

Рис. 4.14: Создаю файл lab6-4.asm, начинаю заполнять для формулы варианта 3



```

[yimirzoyan@fedora lab06]$ gedit lab6-4.asm
[yimirzoyan@fedora lab06]$ nasm -f elf lab6-4.asm
[yimirzoyan@fedora lab06]$ ld -m elf_i386 -o lab6-4 lab6-4.o
[yimirzoyan@fedora lab06]$ ./lab6-4
Введите значение переменной x: 4
Результат: 36[yimirzoyan@fedora lab06]$ ./lab6-4
Введите значение переменной x: 5
Результат: 49[yimirzoyan@fedora lab06]$

```

Рис. 4.15: Проверяю работоспособность программы для  $x_1=4$  и  $x_2=5$  из пункта 6.3, значения получены верные



## **5 Выводы**

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

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-

- е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).