

# **Отчет по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

Луцкая Алиса Витальевна

# 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 . . . . .	14
4.2.1	Ответы на вопросы по программе . . . . .	17
4.3	Выполнение заданий для самостоятельной работы . . . . .	18
<b>5</b>	<b>Выводы</b>	<b>20</b>
<b>6</b>	<b>Список литературы</b>	<b>21</b>

# Список иллюстраций

4.1	Создание директории . . . . .	9
4.2	Создание файла . . . . .	9
4.3	Создание копии файла . . . . .	9
4.4	Редактирование файла . . . . .	10
4.5	Запуск исполняемого файла . . . . .	10
4.6	Редактирование файла . . . . .	11
4.7	Запуск исполняемого файла . . . . .	11
4.8	Редактирование файла . . . . .	12
4.9	Запуск исполняемого файла . . . . .	12
4.10	Редактирование файла . . . . .	13
4.11	Запуск исполняемого файла . . . . .	13
4.12	Редактирование файла . . . . .	13
4.13	Запуск исполняемого файла . . . . .	14
4.14	Редактирование файла . . . . .	14
4.15	Запуск исполняемого файла . . . . .	15
4.16	Изменение программы . . . . .	15
4.17	Запуск исполняемого файла . . . . .	16
4.18	Редактирование файла . . . . .	16
4.19	Запуск исполняемого файла . . . . .	17
4.20	Написание программы . . . . .	18
4.21	Запуск исполняемого файла . . . . .	18
4.22	Повторный запуск исполняемого файла . . . . .	19

## **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 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. - 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. 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. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно



## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

1. Создаю каталог для программ лабораторной работы № 6 и перехожу в него (рис. 4.1).

```
avluckaya@vbox:~$ mkdir ~/work/arch-pc/lab06  
avluckaya@vbox:~$ cd ~/work/arch-pc/lab06
```

Рис. 4.1: Создание директории

Создаю файл lab6-1.asm (рис. 4.2).

```
avluckaya@vbox:~/work/arch-pc/lab06$ touch lab6-1.asm  
avluckaya@vbox:~/work/arch-pc/lab06$ ls  
lab6-1.asm
```

Рис. 4.2: Создание файла

2. Перед созданием исполняемого файла создаю копию файла in\_out.asm в каталоге ~/work/arch-pc/lab06 (рис. 4.3).

```
avluckaya@vbox:~/work/arch-pc/lab06$ cp ~/Загрузки/in_out.asm in_out.asm  
avluckaya@vbox:~/work/arch-pc/lab06$ ls  
in_out.asm lab6-1.asm
```

Рис. 4.3: Создание копии файла

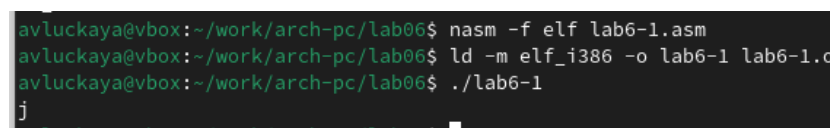
Открываю созданный файл lab6-1.asm, вставляю в него программу (рис. 4.4).



```
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: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 4.5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



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

Рис. 4.5: Запуск исполняемого файла

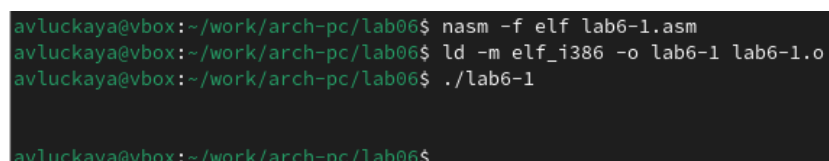
3. Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.6).

A screenshot of a text editor window. The title bar shows 'lab6-...' and the file path '~/.work...'. The editor contains assembly code with line numbers 1 through 13. The code includes a directive to include 'in\_out.asm', defines a .bss section with a buffer 'buf1' of 80 bytes, and a .text section with a global '\_start' symbol. The assembly instructions are: 'mov eax, 6', 'mov ebx, 4', 'add eax, ebx', 'mov [buf1], eax', 'mov eax, buf1', 'call sprintf', and 'call quit'.

```
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. 4.6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.7). Выводится символ с кодом 10, это символ перевода строки, он не отображается при выводе на экран.

A screenshot of a terminal window showing the compilation and execution of the assembly file. The user runs 'nasm -f elf lab6-1.asm', then 'ld -m elf\_i386 -o lab6-1 lab6-1.o', and finally './lab6-1'. The prompt shows the user is at the directory '~/.work/arch-pc/lab06'.

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

Рис. 4.7: Запуск исполняемого файла

4. Создаю файл lab6-2.asm в каталоге ~/.work/arch-pc/lab06 и ввожу в него текст программы из листинга 6.2 (рис. 4.8).

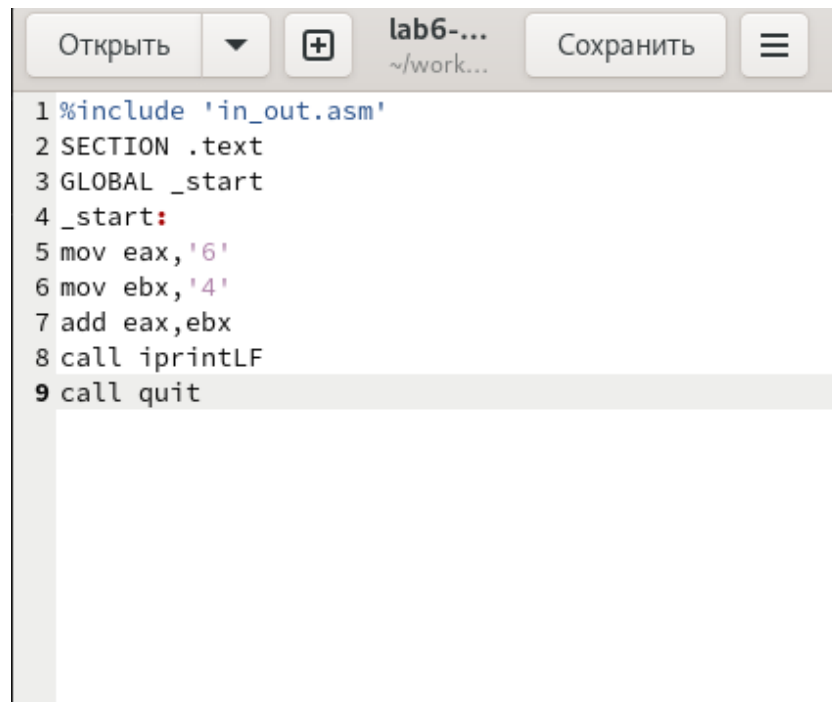


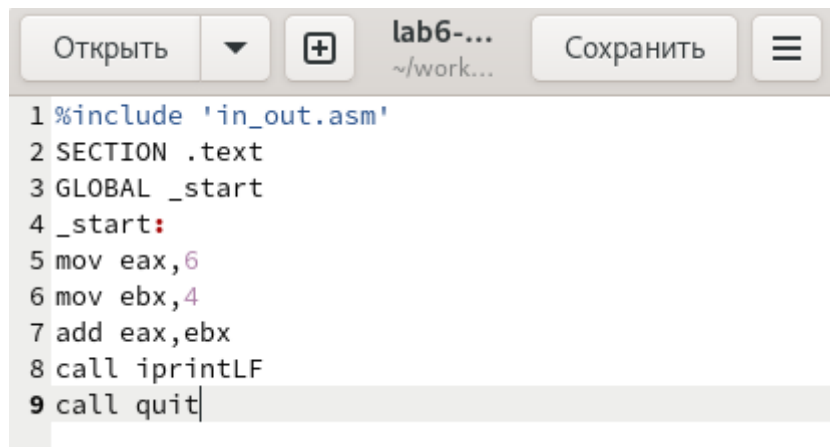
Рис. 4.8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.9). 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”.

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

Рис. 4.9: Запуск исполняемого файла

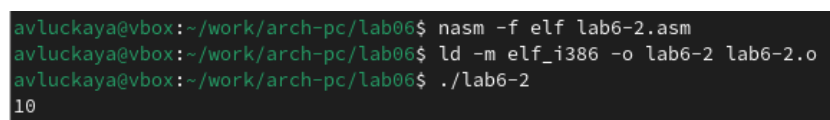
5. Аналогично предыдущему примеру заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.10).



```
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. 4.10: Редактирование файла

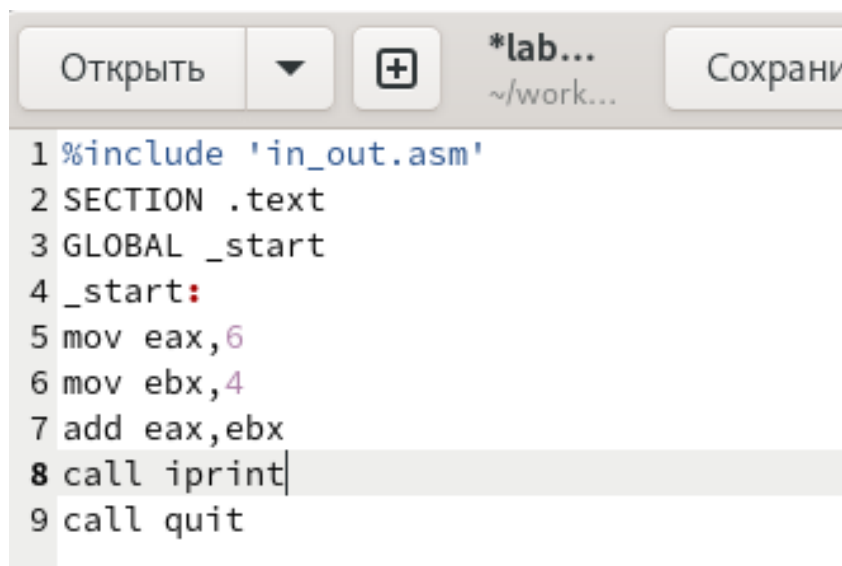
Создаю и запускаю новый исполняемый файл (рис. 4.11).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.



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

Рис. 4.11: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.12).



```
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. 4.12: Редактирование файла

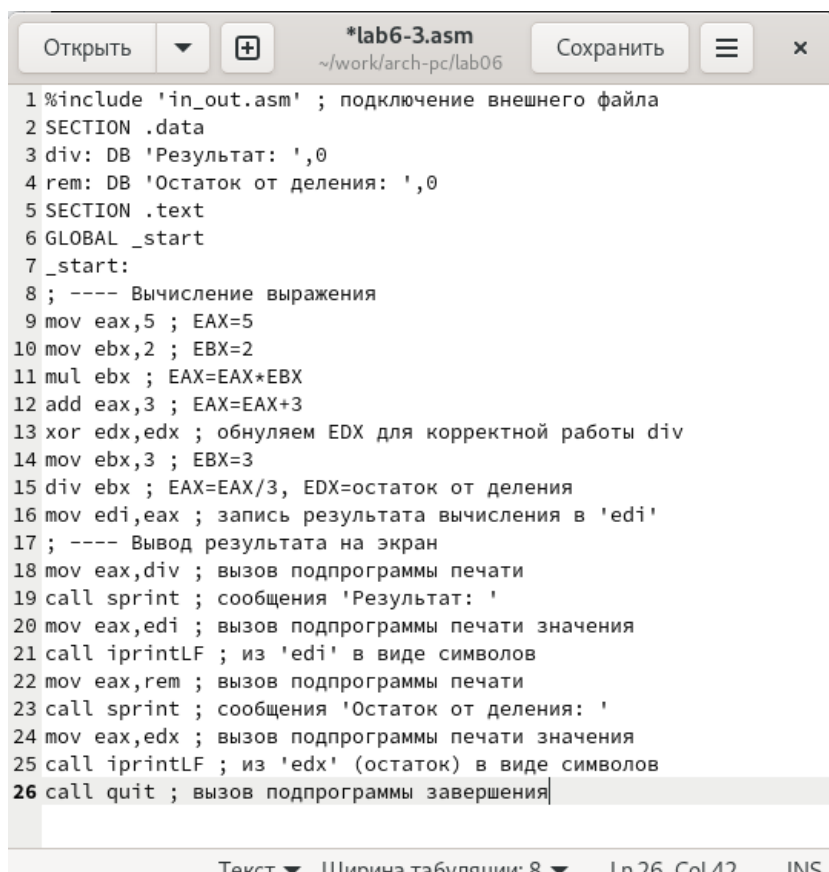
Создаю и запускаю новый исполняемый файл (рис. 4.13). Вывод не изменился, единственное `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`, поэтому командная строка отображается на той же строке, что и результат программы.

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

Рис. 4.13: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

6. Создаю файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`, внимательно изучаю текст программы из листинга 6.3 и ввожу в `lab6-3.asm` (рис. 4.14).



```
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
```

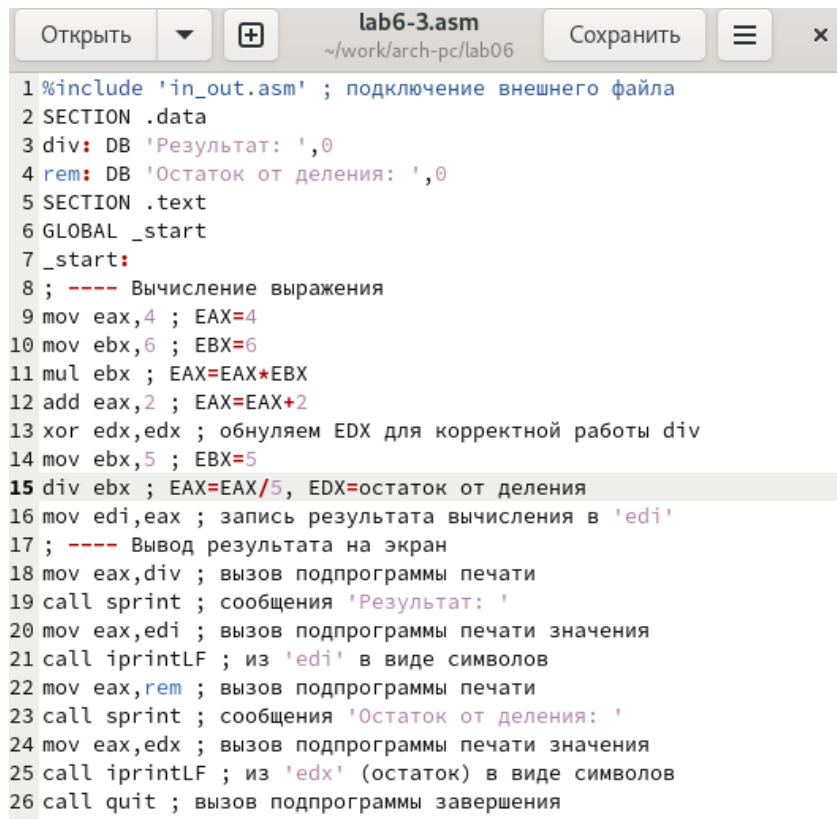
Рис. 4.14: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.15).

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

Рис. 4.15: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 4.16).



```
Открыть  lab6-3.asm  Сохранить  x
~/work/arch-pc/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,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. 4.16: Изменение программы

7. Создаю и запускаю новый исполняемый файл (рис. 4.17). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

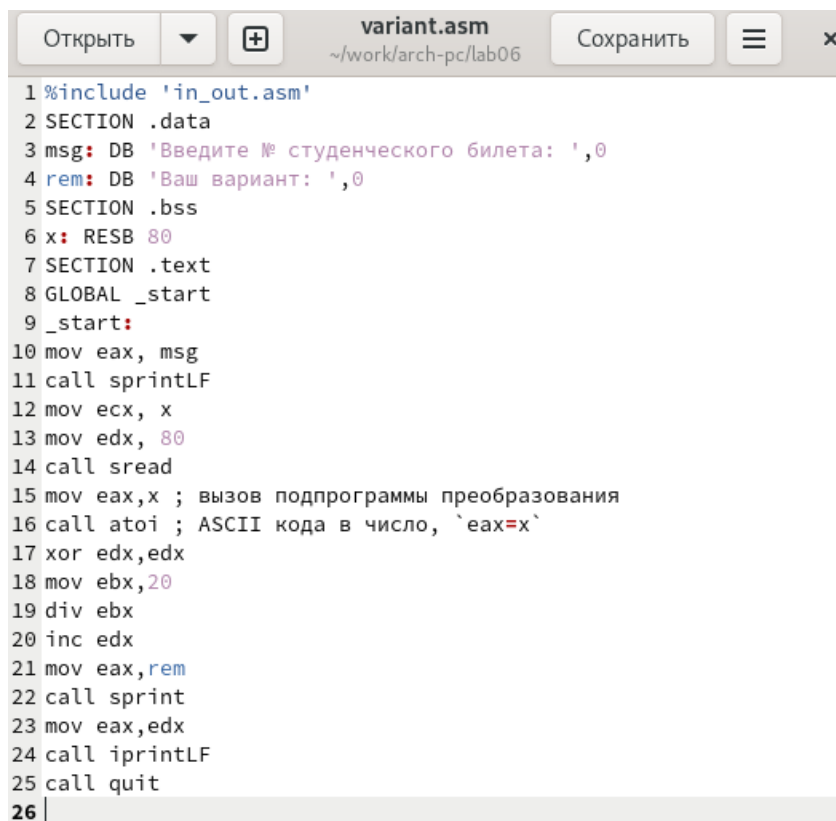
```

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

```

Рис. 4.17: Запуск исполняемого файла

Создаю файл `variant.asm` и ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.18).



```

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № 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
26

```

Рис. 4.18: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.19). Ввожу номер своего студенческого билета, программа вывела мой вариант - 11.



```

avluckaya@vbox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
avluckaya@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
avluckaya@vbox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246750
Ваш вариант: 11
avluckaya@vbox:~/work/arch-pc/lab06$

```

Рис. 4.19: Запуск исполняемого файла

### 4.2.1 Ответы на вопросы по программе

1. За вывод на экран сообщения “Ваш вариант” отвечают строки:

```

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

```

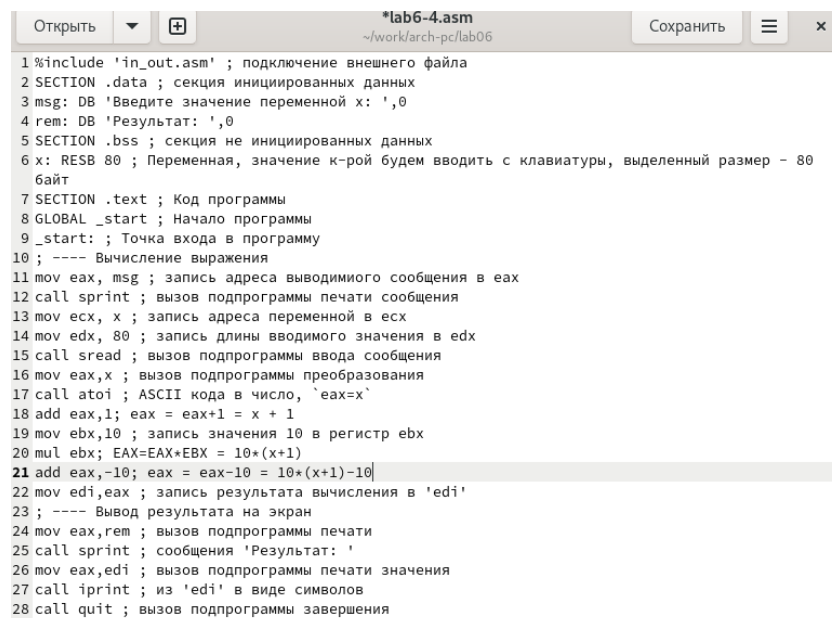
5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1

7. За вывод на экран результатов вычислений отвечают строки:

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

## 4.3 Выполнение заданий для самостоятельной работы

С помощью touch Создаю файл lab6-4.asm, открываю в редакторе gedit, ввожу в него текст программы для вычисления значения выражения  $10 \cdot (1 + x) - 10$ . Это выражение было под вариантом 11. (рис. 4.20).

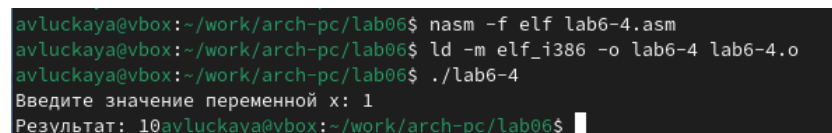


```
*lab6-4.asm
~/.work/arch-pc/lab06
Сохранить

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data ; секция инициализированных данных
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss ; секция не инициализированных данных
6 x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80
байт
7 SECTION .text ; Код программы
8 GLOBAL _start ; Начало программы
9 _start: ; Точка входа в программу
10 ; ---- Вычисление выражения
11 mov eax, msg ; запись адреса выводимого сообщения в eax
12 call sprint ; вызов подпрограммы печати сообщения
13 mov ecx, x ; запись адреса переменной в ecx
14 mov edx, 80 ; запись длины вводимого значения в edx
15 call sread ; вызов подпрограммы ввода сообщения
16 mov eax,x ; вызов подпрограммы преобразования
17 call atoi ; ASCII кода в число, `eax=x`
18 add eax,1; eax = eax+1 = x + 1
19 mov ebx,10 ; запись значения 10 в регистр ebx
20 mul ebx; EAX=EAX*EBX = 10*(x+1)
21 add eax,-10; eax = eax-10 = 10*(x+1)-10
22 mov edi,eax ; запись результата вычисления в 'edi'
23 ; ---- Вывод результата на экран
24 mov eax,rem ; вызов подпрограммы печати
25 call sprint ; сообщения 'Результат: '
26 mov eax,edi ; вызов подпрограммы печати значения
27 call iprint ; из 'edi' в виде символов
28 call quit ; вызов подпрограммы завершения
```

Рис. 4.20: Написание программы

Создаю и запускаю исполняемый файл (рис. 4.21). При вводе значения 1, вывод - 10.



```
avluckaya@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
avluckaya@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
avluckaya@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 10avluckaya@vbox:~/work/arch-pc/lab06$
```

Рис. 4.21: Запуск исполняемого файла

Проверяю корректность работы программы с другим значением на вводе. Провожу запуск исполняемого файла, на вводе значение 5, вывод- 50. Программы работает корректно (рис. 4.22).

```
Результат: 10avluckaya@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
avluckaya@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
avluckaya@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 5
Результат: 50avluckaya@vbox:~/work/arch-pc/lab06$
```

Рис. 4.22: Повторный запуск исполняемого файла

## **5 Выводы**

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

## **6 Список литературы**

1. <https://esystem.rudn.ru/course/view.php?id=112>