

Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Учаева Алёна Сергеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	13
4.2.1	Ответы на вопросы	15
4.3	Выполнение заданий для самостоятельной работы	16
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Создание файла	8
4.2	Редактирование файла	9
4.3	Запуск исполняемого 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	12
4.11	Запуск исполняемого файла	13
4.12	Создание файла	13
4.13	Редактирование файла	13
4.14	Запуск исполняемого файла	14
4.15	Редактирование файла	14
4.16	Запуск исполняемого файла	14
4.17	Создание файла	15
4.18	Редактирование файла	15
4.19	Запуск исполняемого файла	15
4.20	Создание файла	16
4.21	Редактирование файла	17
4.22	Запуск исполняемого файла	17

1 Цель работы

Цель данной лабораторной работы освоить арифметические инструкции 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:

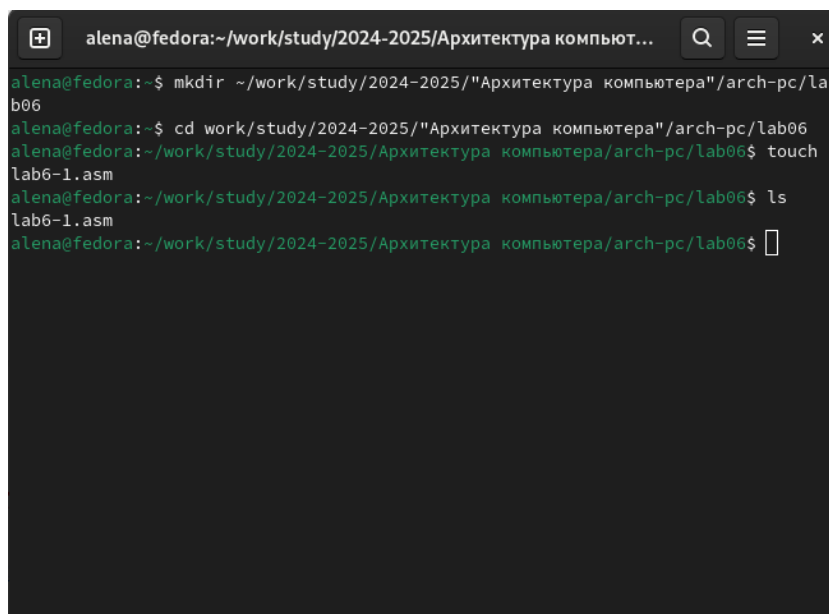
- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается 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. 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 делает 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

С помощью команды `mkdir` создаю каталог для программ лабораторной работы №6, перехожу в него и создаю файл `lab6-1.asm` (рис. 4.1).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютер...  
alena@fedora:~$ mkdir ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06  
alena@fedora:~$ cd work/study/2024-2025/Архитектура компьютера/arch-pc/lab06  
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-1.asm  
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ls  
lab6-1.asm  
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.1: Создание файла

Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.2).

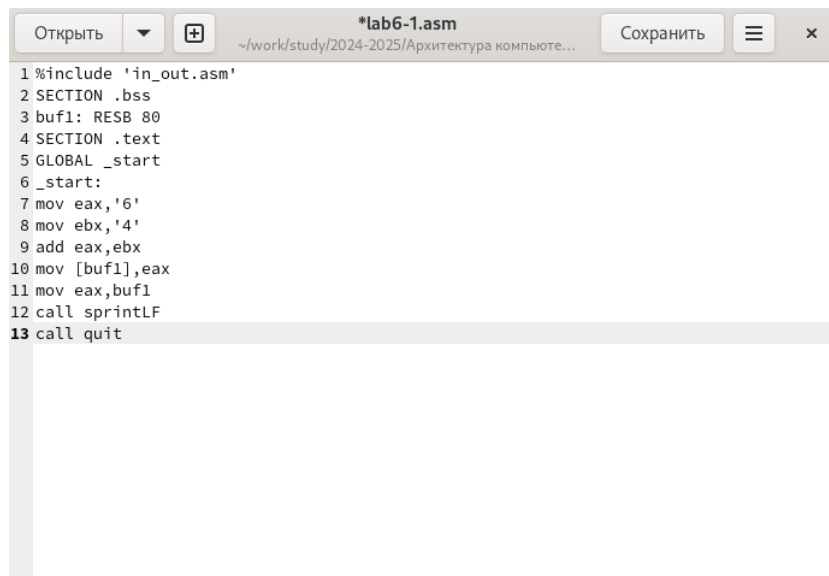


Рис. 4.2: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 4.3).

```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-1.o
ld: отсутствуют входные файлы
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-1
j
```

Рис. 4.3: Запуск исполняемого файла

Далее изменяю текст программы, вместо символов записываю в регистры числа (рис. 4.4).

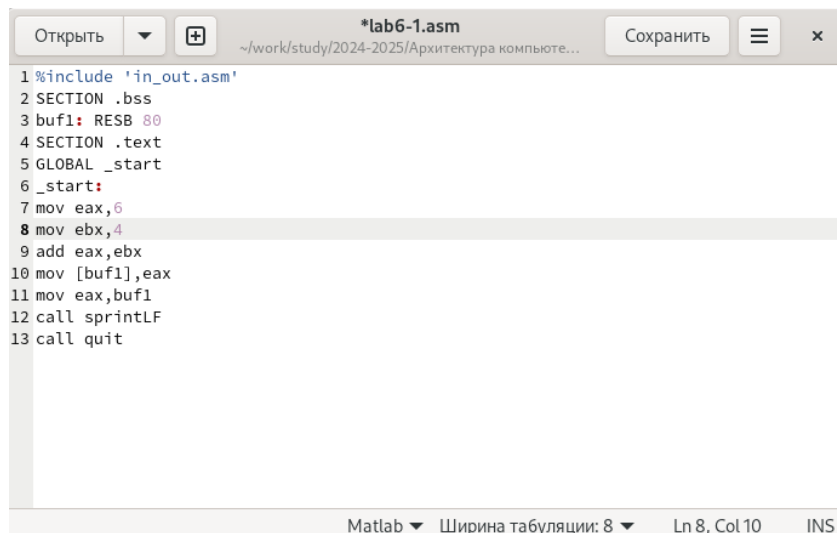


Рис. 4.4: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.5). Вывелся симьюл с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

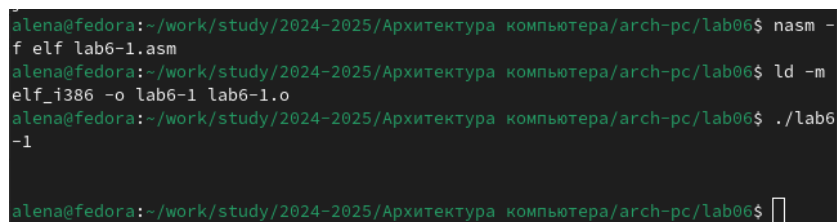


Рис. 4.5: Запуск исполняемого файла

С помощью команды `touch` создаю файл `lab6-2.asm` и ввожу в него текст программы (рис. 4.6).

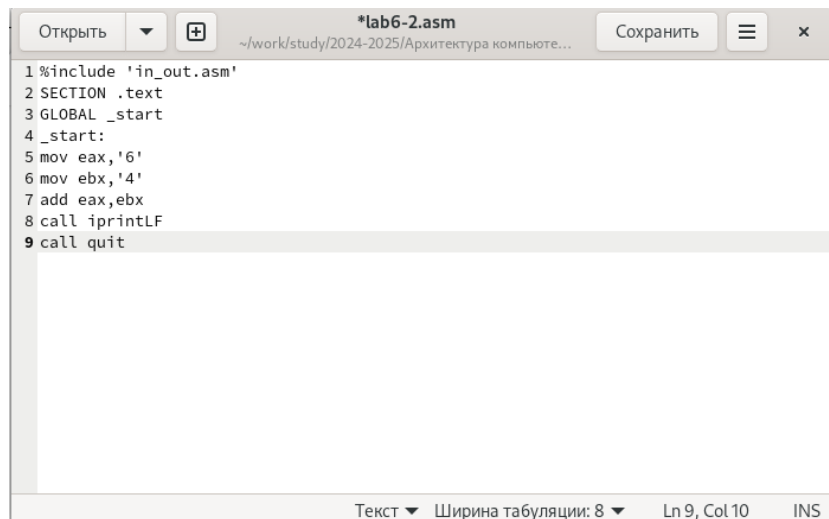


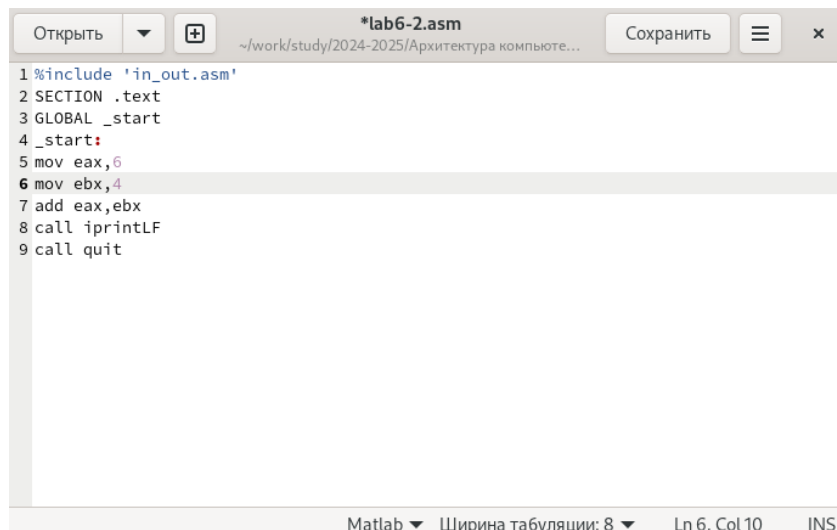
Рис. 4.6: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.7).

```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-2.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
106
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.7: Запуск исполняемого файла

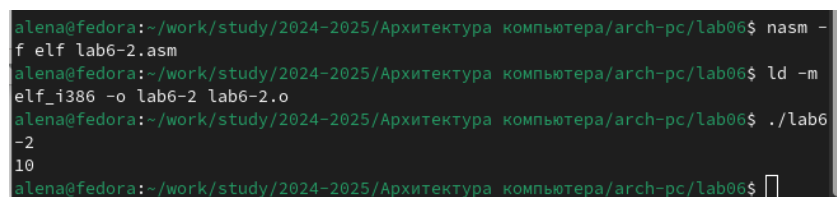
Далее изменяю символы на числа (рис. 4.8).



```
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. 4.8: Редактирование файла

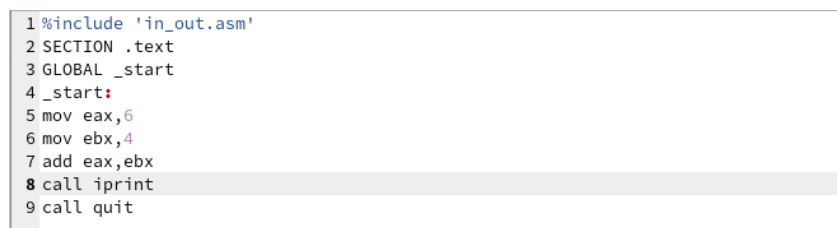
Создаю и запускаю исполняемый файл (рис. 4.9). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому выводит 10.



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
10
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.9: Запуск исполняемого файла

Заменяю функцию iprintLF на Iprint (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 iprint
9 call quit
```

Рис. 4.10: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.11). 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`.

```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
-2
10
```

Рис. 4.11: Запуск исполняемого файла

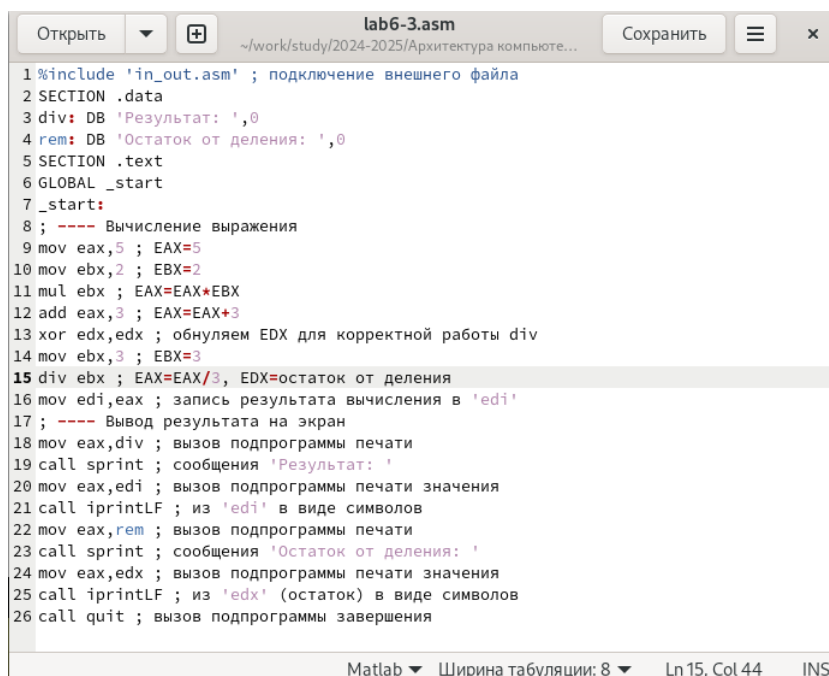
4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3` с помощью команды `touch` (рис. 4.12).

```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-3.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.12: Создание файла

Ввожу текст программы для вычисления выражения $\lfloor (5 \cdot 2 + 3) / 3 \rfloor$ в исполняемый файл (рис. 4.13).



```
lab6-3.asm
~/work/study/2024-2025/Архитектура компьюте...
Сохранить

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

Matlab  Ширина табуляции: 8  Ln 15, Col 44  INS
```

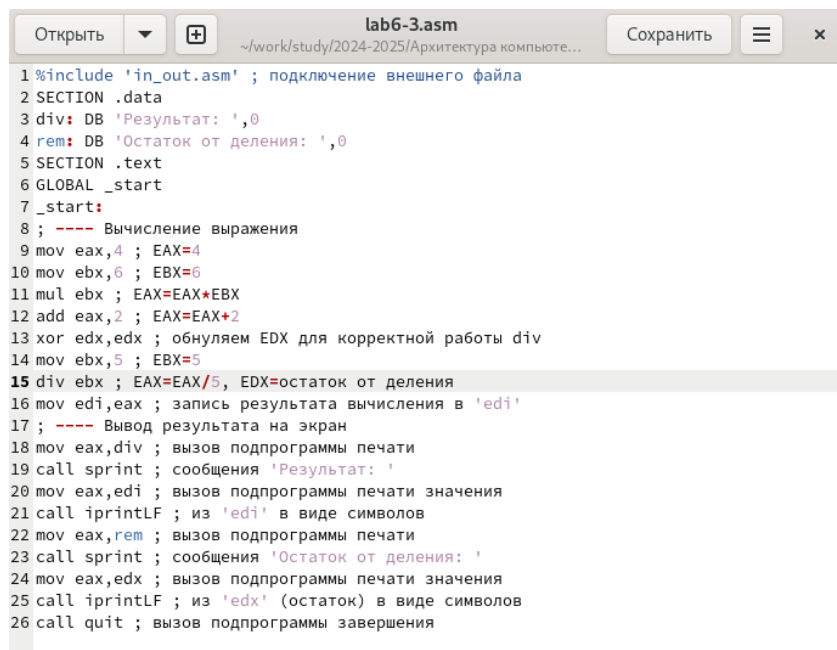
Рис. 4.13: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.14).

```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-3.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.14: Запуск исполняемого файла

Далее изменяю текст программы для вычисления выражения $\square(\square) = (4 \cdot 6 + 2) / 5$ (рис. 4.15).



```
Открыть  +  lab6-3.asm  Сохранить  x
~/work/study/2024-2025/Архитектура компьюте...
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. 4.15: Редактирование файла

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

```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-3.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

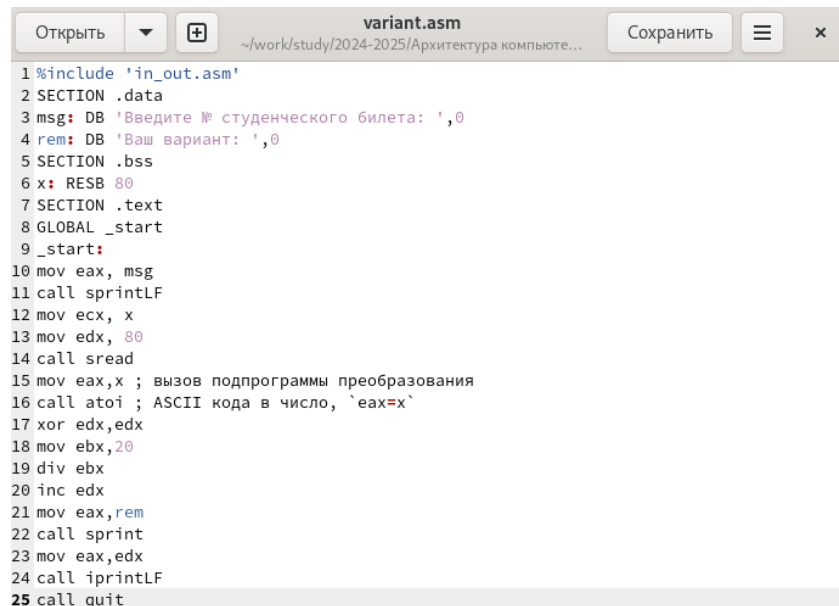
Рис. 4.16: Запуск исполняемого файла

Создаю файл variant.asm (рис. 4.17).

```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch variant.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.17: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.18).



```
variant.asm
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
```

Рис. 4.18: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.19).

```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf variant.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246728
Ваш вариант: 9
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.19: Запуск исполняемого файла

4.2.1 Ответы на вопросы

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`

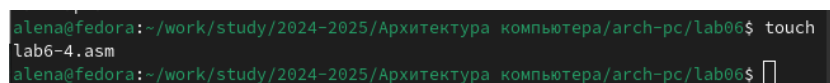
```
mov ebx,20 ; ebx=20  
div ebx ; eax=eax/20, edx-остаток от деления  
inc edx ; edx=edx+1
```

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

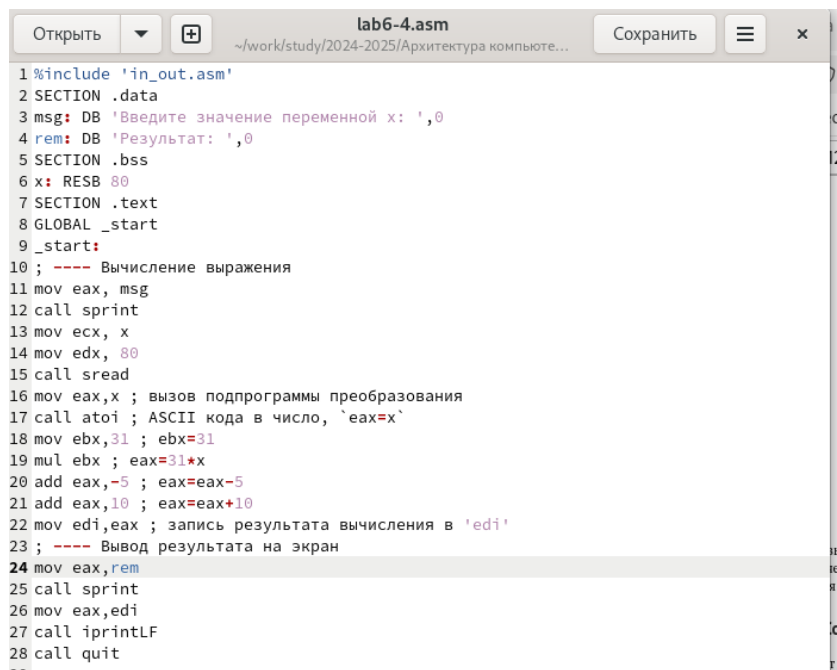
Создаю файл `lab6-4.asm` (рис. 4.20).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch  
lab6-4.asm  
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.20: Создание файла

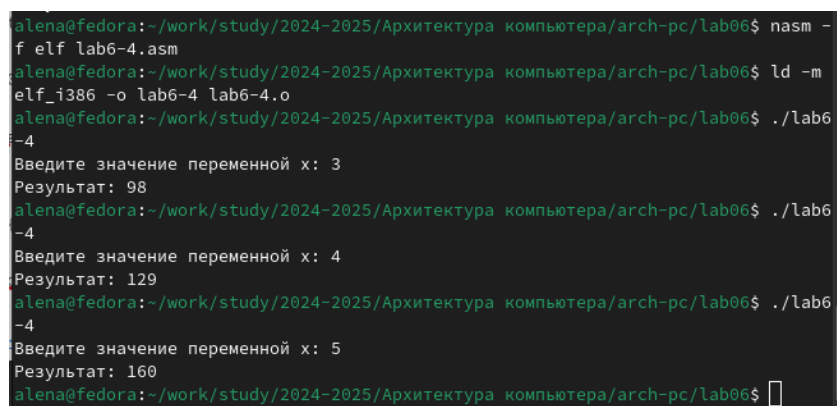
Ввожу в созданный файл текст программы для вычисления $10+(31*x-5)$ (вариант №9) (рис. 4.21).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 ; ---- Вычисление выражения
11 mov eax, msg
12 call sprint
13 mov ecx, x
14 mov edx, 80
15 call sread
16 mov eax, x ; вызов подпрограммы преобразования
17 call atoi ; ASCII кода в число, `eax=x`
18 mov ebx, 31 ; ebx=31
19 mul ebx ; eax=31*x
20 add eax, -5 ; eax=eax-5
21 add eax, 10 ; eax=eax+10
22 mov edi, eax ; запись результата вычисления в `edi`
23 ; ---- Вывод результата на экран
24 mov eax, rem
25 call sprint
26 mov eax, edi
27 call iprintLF
28 call quit
```

Рис. 4.21: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.22).



```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-4.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-4
-4
Введите значение переменной x: 3
Результат: 98
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-4
-4
Введите значение переменной x: 4
Результат: 129
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-4
-4
Введите значение переменной x: 5
Результат: 160
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.22: Запуск исполняемого файла

Программа сработала верно

**Листинг программы для вычисления выражения $10+(31*x-5)$

```
%include 'in_out.asm'
```

```

SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
mov ebx,31 ; ebx=31
mul ebx ; eax=31*x
add eax, -5 ; eax=eax-5
add eax,10 ; eax=eax+10
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,rem
call sprint
mov eax,edi
call iprintLF
call quit

```

5 Выводы

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

Список литературы

1. Архитектура ЭВМ