

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Абронина Алиса Кирилловна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.0.1	Ответы на вопросы по программе	16
4.1	Выполнение заданий для самостоятельной работы	17
5	Выводы	19

Список иллюстраций

4.1	Создание файла	9
4.2	Создание копии файла	9
4.3	Редактирование файла	10
4.4	Запуск исполняемого файла	10
4.5	Редактирование файла	11
4.6	Запуск исполняемого файла	11
4.7	Создание 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	Редактирование файла	13
4.13	Запуск исполняемого файла	13
4.14	Создание файла	13
4.15	Редактирование файла	14
4.16	Запуск исполняемого файла	14
4.17	Изменение программы	15
4.18	Запуск исполняемого файла	15
4.19	Создание файла	15
4.20	Редактирование файла	16
4.21	Запуск исполняемого файла	16
4.22	Создание файла	17
4.23	Написание программы	18
4.24	Запуск исполняемого файла	18
4.25	Запуск исполняемого файла	18

List of Tables

1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Выполнение арифметических операций в NASM
2. 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

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. ??). Перехожу в созданный ка-

талог с помощью утилиты `cd`.

```
akabronina@vbox:~$ mkdir ~/work/study/2024-2025/"Архитектура компьютера"/archpc/lab06
akabronina@vbox:~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/archpc/lab06
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. 4.1).

```
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ touch lab6-1.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ls
lab6-1.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

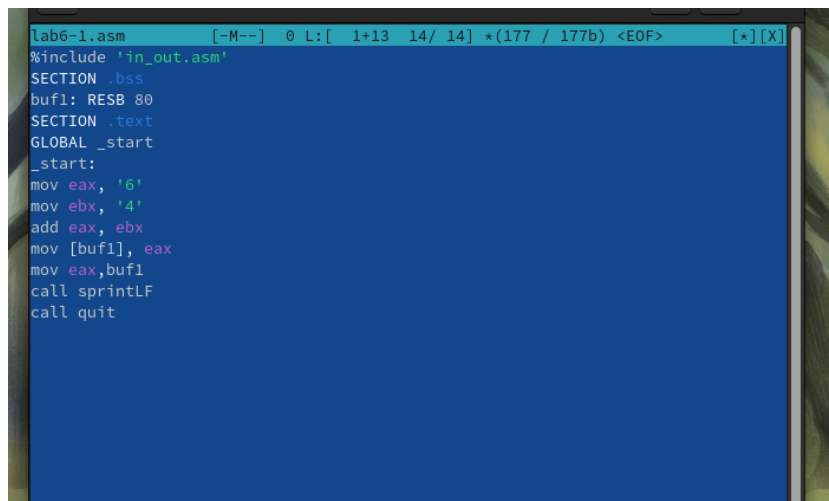
Рис. 4.1: Создание файла

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах (рис. 4.2).

```
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ cp ~/Загрузки/in_out.asm in_out.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ls
in_out.asm lab6-1.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.2: Создание копии файла

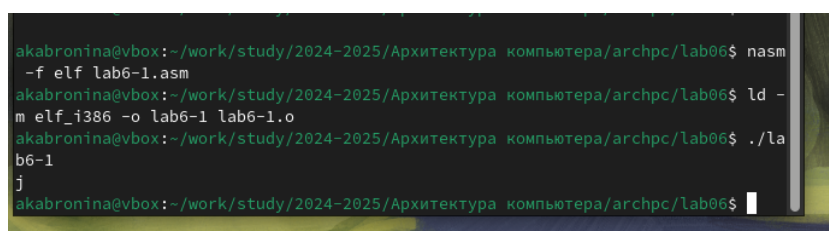
Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.3).

A screenshot of a text editor window titled 'lab6-1.asm'. The editor has a blue background and a light blue title bar. The code is as follows:

```
lab6-1.asm  [-M--]  0 L:[ 1+13 14/ 14] *(177 / 177b) <EOF>  [*][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 sprintf
call quit
```

Рис. 4.3: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 4.4). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

A screenshot of a terminal window with a dark background and green text. The user is 'akabronina@vbox' and the directory is '~/work/study/2024-2025/Архитектура компьютера/archpc/lab06'. The commands and output are:

```
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ nasm -f elf lab6-1.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ./lab6-1
j
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.4: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.5).

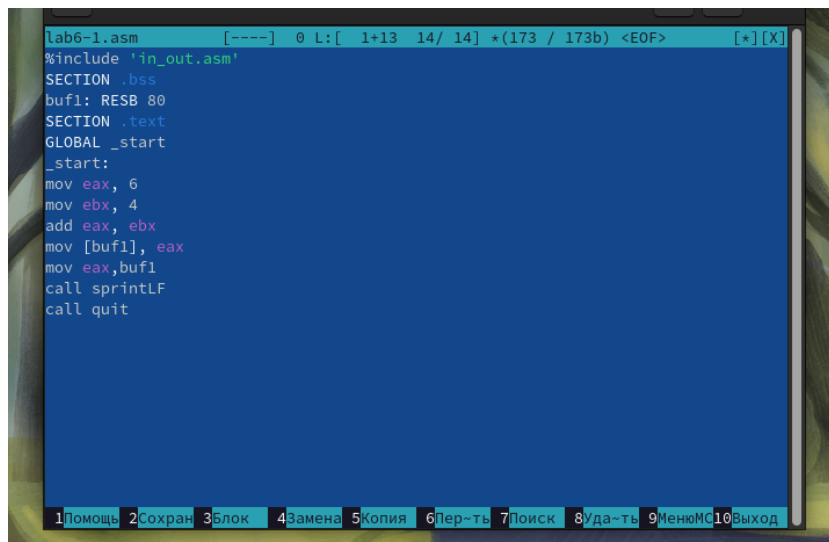


Рис. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.6). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

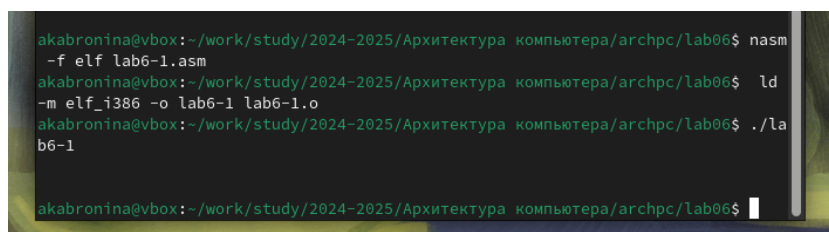


Рис. 4.6: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 4.7).

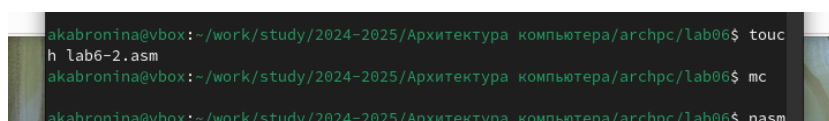
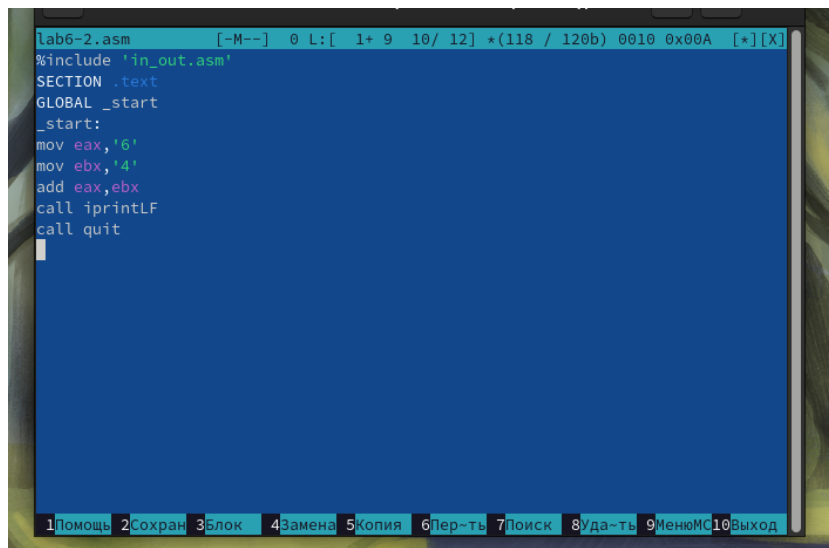


Рис. 4.7: Создание файла

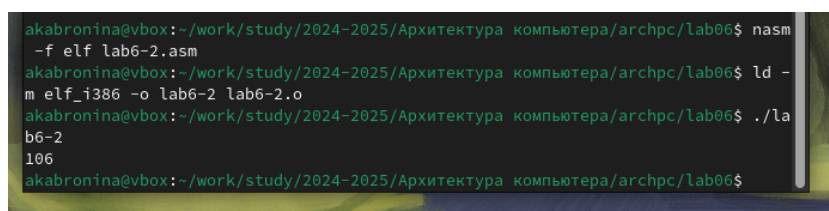
Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 4.8).



```
lab6-2.asm [-M--] 0 L: [ 1+ 9 10/ 12] *(118 / 120b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 4.8: Редактирование файла

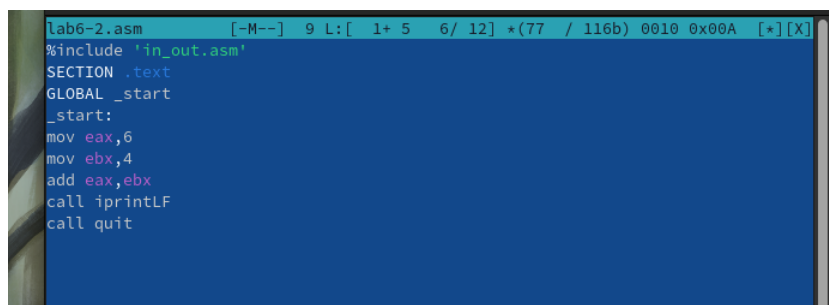
Создаю и запускаю исполняемый файл lab6-2 (рис. 4.9). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.



```
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ nasm
-f elf lab6-2.asm
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ld -
m elf_i386 -o lab6-2 lab6-2.o
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ./la
b6-2
106
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.9: Запуск исполняемого файла

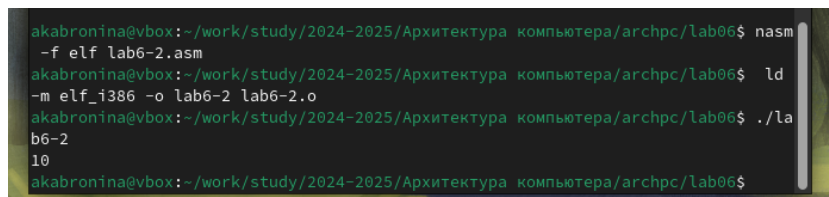
Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.10).



```
lab6-2.asm [-M--] 9 L: [ 1+ 5 6/ 12] *(77 / 116b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.10: 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.



```
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ nasm -f elf lab6-2.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ./lab6-2
10
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.11: Запуск исполняемого файла

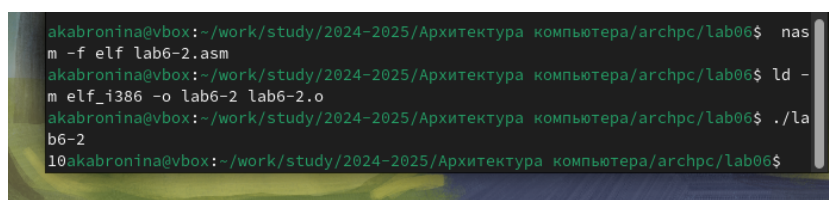
Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 4.12).



```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 4.12: Редактирование файла

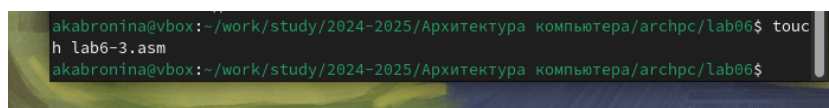
Создаю и запускаю новый исполняемый файл (рис. 4.13). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией `iprintLF`, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.



```
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ nasm -f elf lab6-2.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ./lab6-2
10
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.13: Запуск исполняемого файла

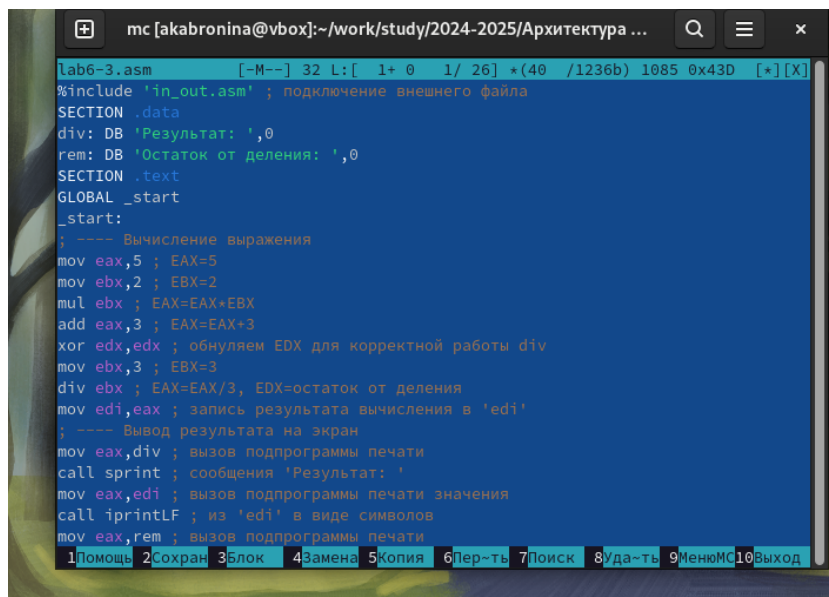
Создаю файл `lab6-3.asm` с помощью утилиты `touch` (рис. 4.14).



```
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ touch lab6-3.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.14: Создание файла

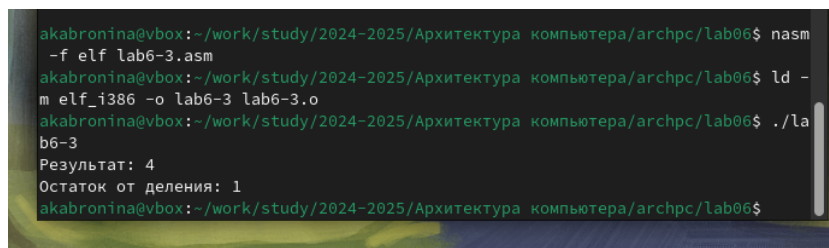
Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.15).



```
lab6-3.asm      [-M--] 32 L: [ 1+ 0 1/ 26] *(40 /1236b) 1085 0x43D [*] [X]
%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,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
```

Рис. 4.15: Редактирование файла

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



```
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ nasm
-f elf lab6-3.asm
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ld -
m elf_i386 -o lab6-3 lab6-3.o
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ./la
b6-3
Результат: 4
Остаток от деления: 1
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.17).

```
lab6-3.asm [-M--] 8 L: [ 1+14 15/ 26] *(404 /1149b) 0010 0x00A [*] [X]
%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
mov ebx,6
mul ebx ; EAX=EAX*EBX
add eax,2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5
div ebx.
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
```

Рис. 4.17: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 4.18). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ nasm
-f elf lab6-3.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ld -
m elf_i386 -o lab6-3 lab6-3.o
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ./la
b6-3
Результат: 5
Остаток от деления: 1
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.18: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 4.19).

```
В КОМ Остаток от деления: 1
СИМВО akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ touc
льном h variant.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ mous
epad variant.asm
akabronina@vbox:~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ mous
```

Рис. 4.19: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.20).

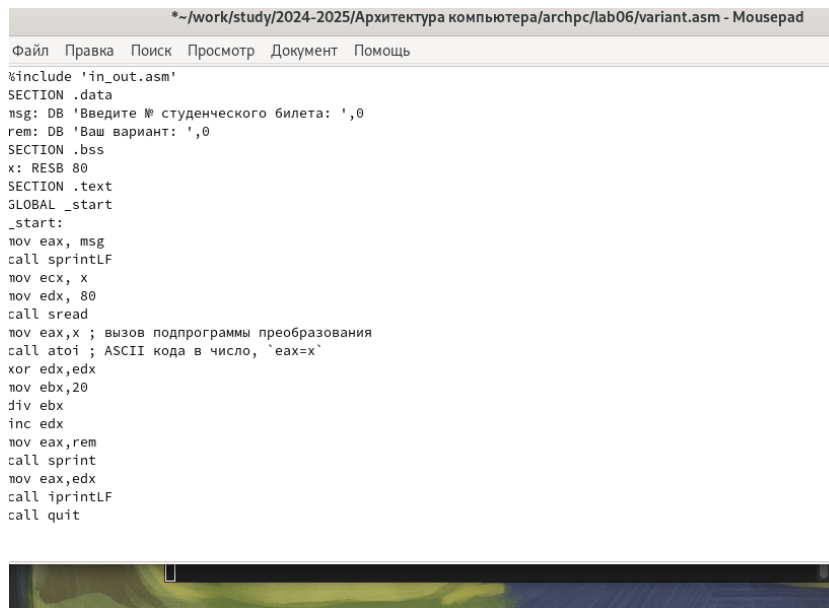


Рис. 4.20: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.21). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 18.

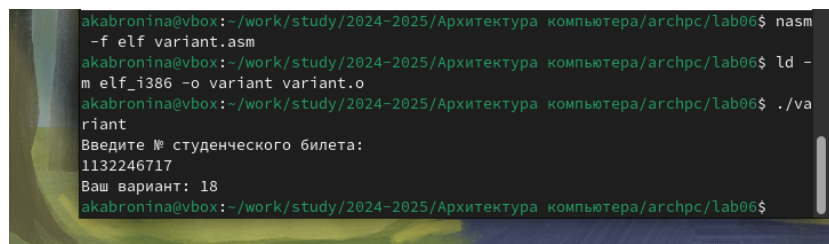


Рис. 4.21: Запуск исполняемого файла

4.0.1 Ответы на вопросы по программе

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
```

4.1 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch (рис. 4.22).

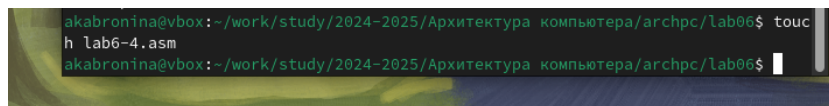
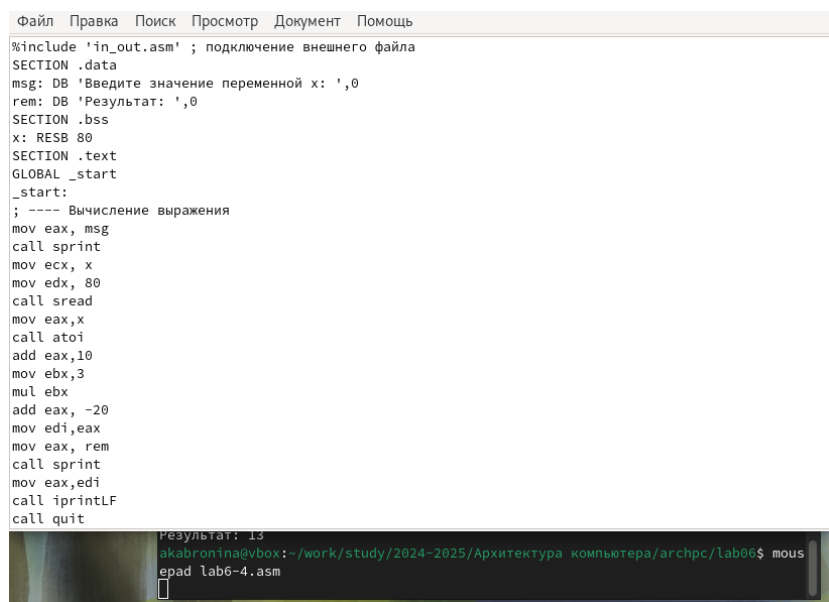


Рис. 4.22: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $3 \cdot (x+10) - 20$ (рис. 4.23). Это выражение было под вариантом 18.

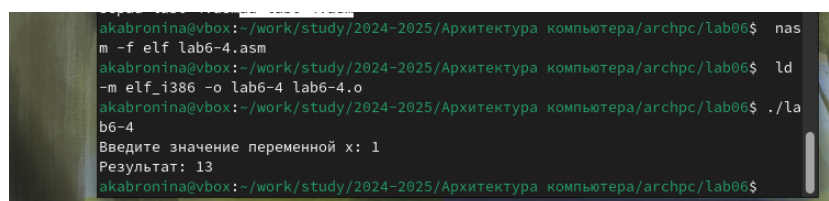


```
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
add eax, 10
mov ebx, 3
mul ebx
add eax, -20
mov edi, eax
mov eax, rem
call sprint
mov eax, edi
call iprintLF
call quit
```

Результат: 13
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06\$ mousepad lab6-4.asm

Рис. 4.23: Написание программы

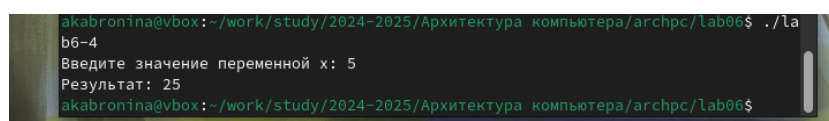
Создаю и запускаю исполняемый файл (рис. 4.24). При вводе значения 1, вывод - 13.



```
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ nas
-m -f elf lab6-4.asm
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ld
-m elf_i386 -o lab6-4 lab6-4.o
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 13
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.24: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 4.25). Программа отработала верно.



```
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$ ./lab6-4
Введите значение переменной x: 5
Результат: 25
akabronina@vbox: ~/work/study/2024-2025/Архитектура компьютера/archpc/lab06$
```

Рис. 4.25: Запуск исполняемого файла

5 Выводы

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