

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Бородин Дмитрий Алексеевич

Содержание

1	Цель работы	1
2	Задание	1
3	Теоретическое введение	1
4	Выполнение лабораторной работы	2
4.1	Символьные и численные данные в NASM	2
4.2	Выполнение арифметических операций в NASM	10
4.3	Ответы на контрольные вопросы	16
4.4	Задание для самостоятельной работы	17
5	Выводы	18
6	Список литературы	18

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

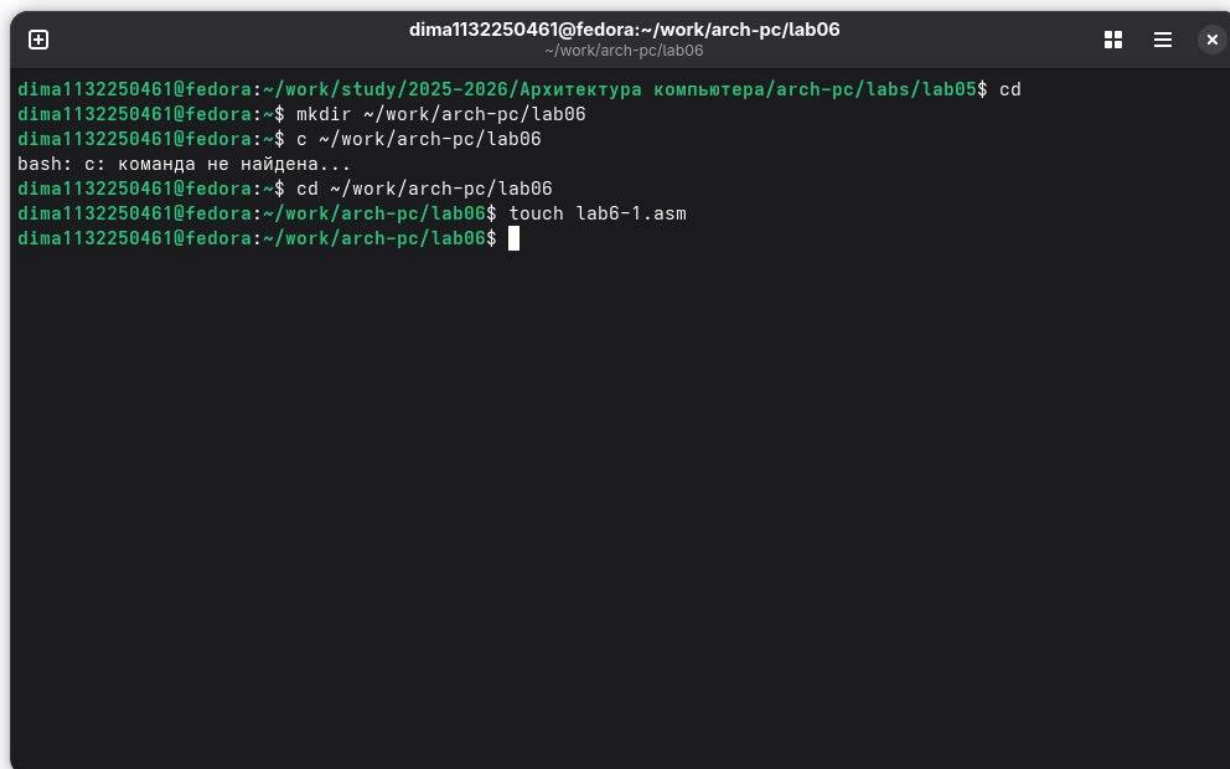
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена 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

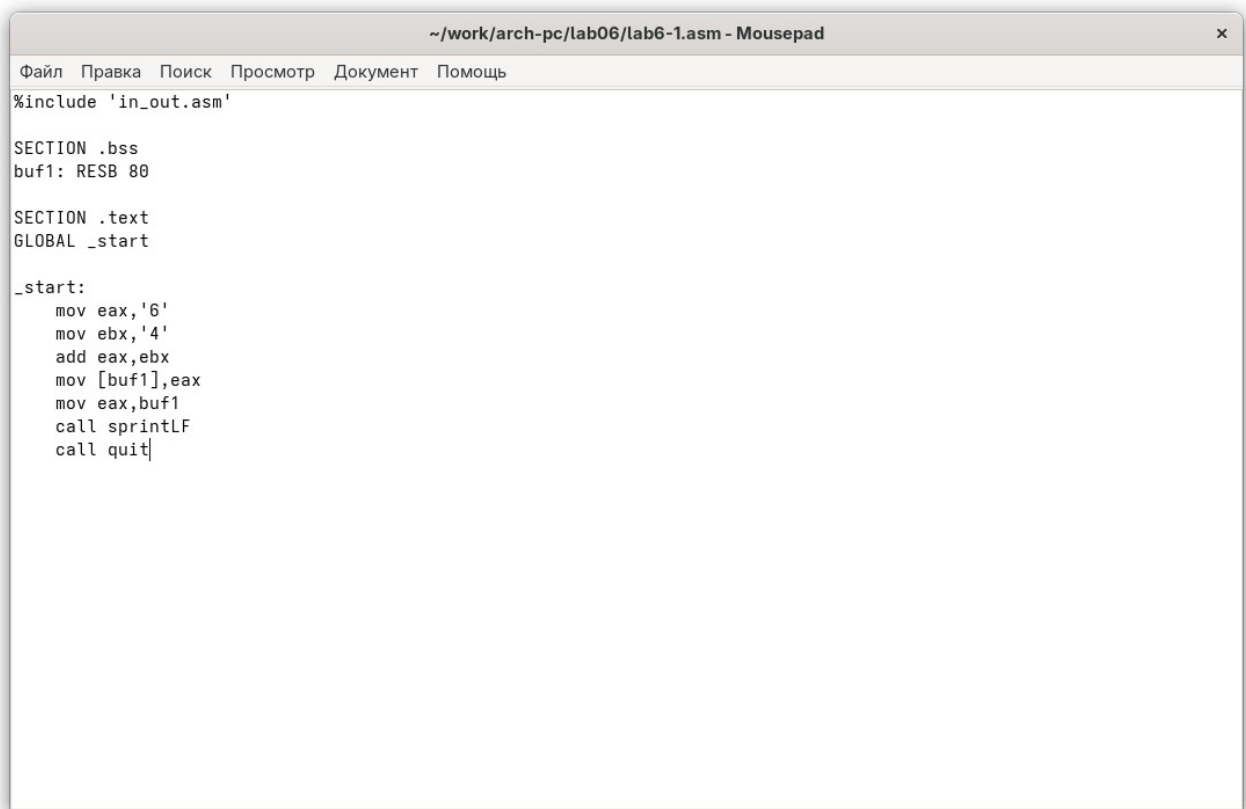
Создаю каталог для программ лабораторной работы №6 и перехожу в него, создаю там файл (рис. 1).



```
dima1132250461@fedora:~/work/arch-pc/lab06
dima1132250461@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab05$ cd
dima1132250461@fedora:~$ mkdir ~/work/arch-pc/lab06
dima1132250461@fedora:~$ cd ~/work/arch-pc/lab06
bash: cd: команда не найдена...
dima1132250461@fedora:~$ cd ~/work/arch-pc/lab06
dima1132250461@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 1: Создание нового каталога

В созданном файле ввожу программу из листинга (рис. 2).

A screenshot of a text editor window titled '~/.work/arch-pc/lab06/lab6-1.asm - Mousepad'. The window has a menu bar with 'Файл', 'Правка', 'Поиск', 'Просмотр', 'Документ', and 'Помощь'. The code inside is assembly language. It includes a file 'in_out.asm', defines a section '.bss' with a buffer 'buf1' of 80 bytes, and a section '.text' with a global symbol '_start'. The '_start' function contains instructions to move '6' to 'eax', '4' to 'ebx', add them, store the result in 'buf1', and then call 'sprintf' and 'quit'.

```
~/.work/arch-pc/lab06/lab6-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'

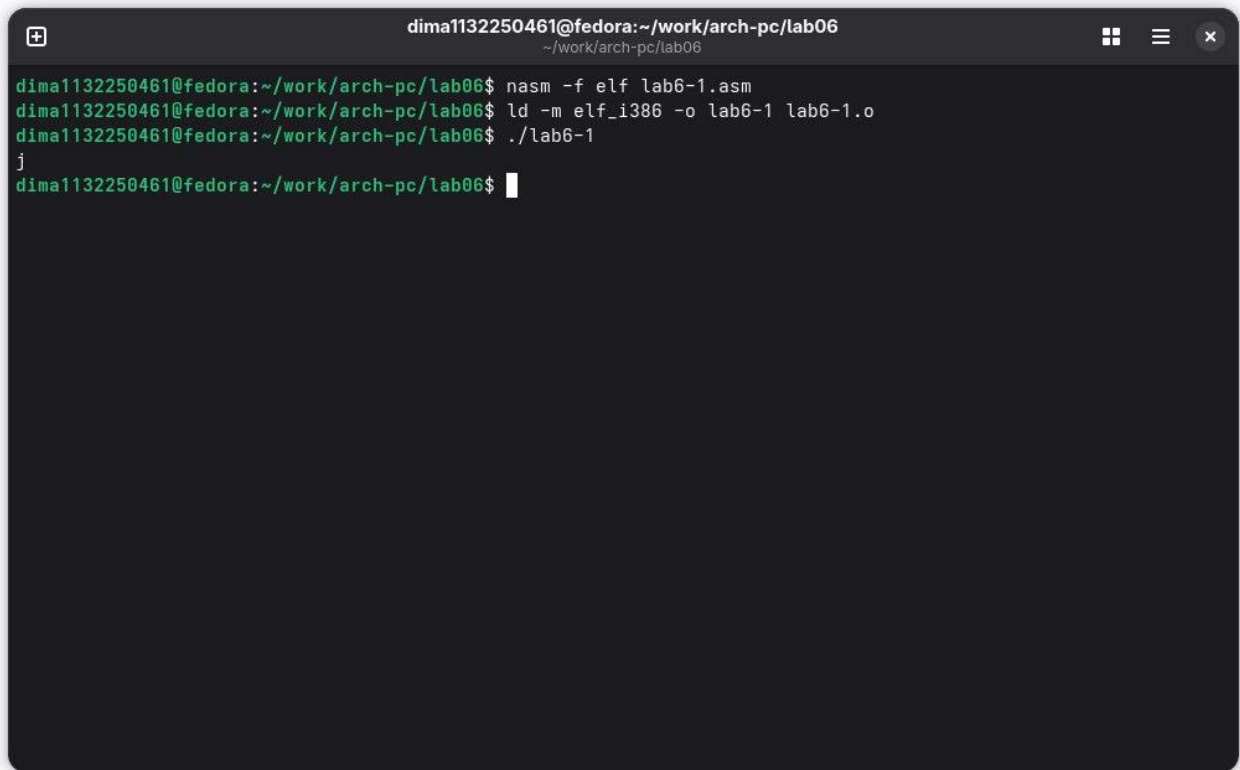
SECTION .bss
buf1: RESB 80

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. 2: Сохранение новой программы

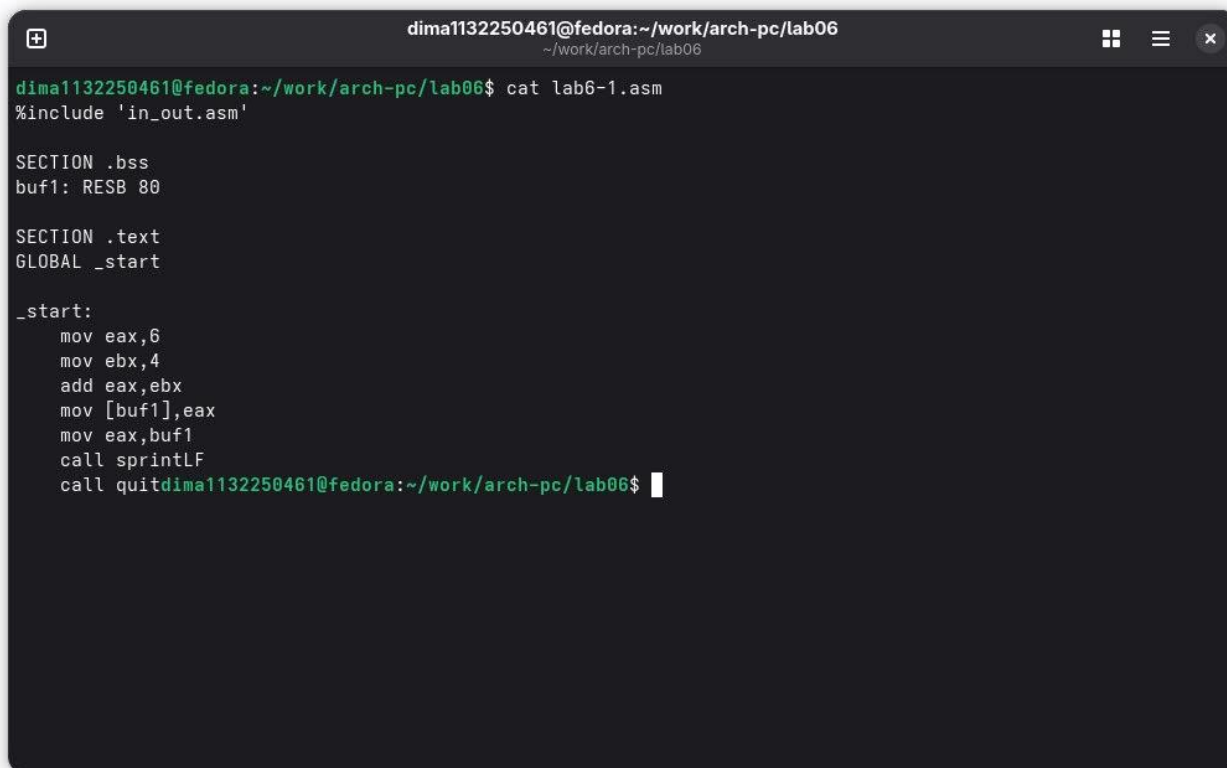
Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, ибо коды символов в сумме дают символ j по таблице ASCII. {#fig:003 width=70%}

A terminal window with a dark background and light green text. The window title is 'dima1132250461@fedora:~/work/arch-pc/lab06'. The terminal shows the following commands and output:

```
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 3: Запуск изначальной программы

Изменяю текст изначальной программы, убрав кавычки (рис. 4).

A terminal window with a dark background and light text. The title bar at the top shows the user 'dima1132250461@fedora' and the directory '~/work/arch-pc/lab06'. The terminal content shows the command 'cat lab6-1.asm' being executed. The output is assembly code: '%include \'in_out.asm\'', 'SECTION .bss', 'buf1: RESB 80', 'SECTION .text', 'GLOBAL _start', and a function '_start:' containing instructions 'mov eax,6', 'mov ebx,4', 'add eax,ebx', 'mov [buf1],eax', 'mov eax,buf1', 'call sprintf', and 'call quit'. The prompt 'dima1132250461@fedora:~/work/arch-pc/lab06\$' is visible at the end of the last line.

```
dima1132250461@fedora:~/work/arch-pc/lab06$ cat lab6-1.asm
%include 'in_out.asm'

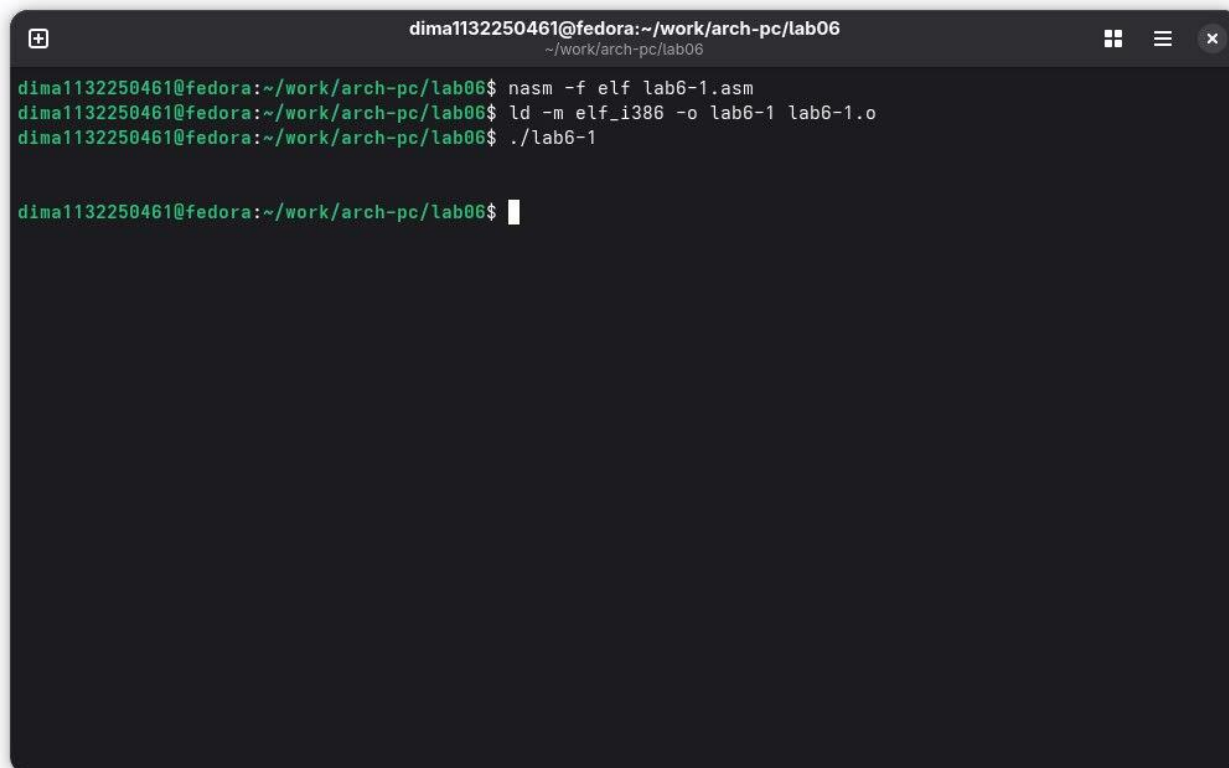
SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start

_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    mov [buf1],eax
    mov eax,buf1
    call sprintf
    call quitdima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 4: Измененная программа

На этот раз программа выдала пустую строку, это связано с тем, что символ 10 означает переход на новую строку (рис. 5).

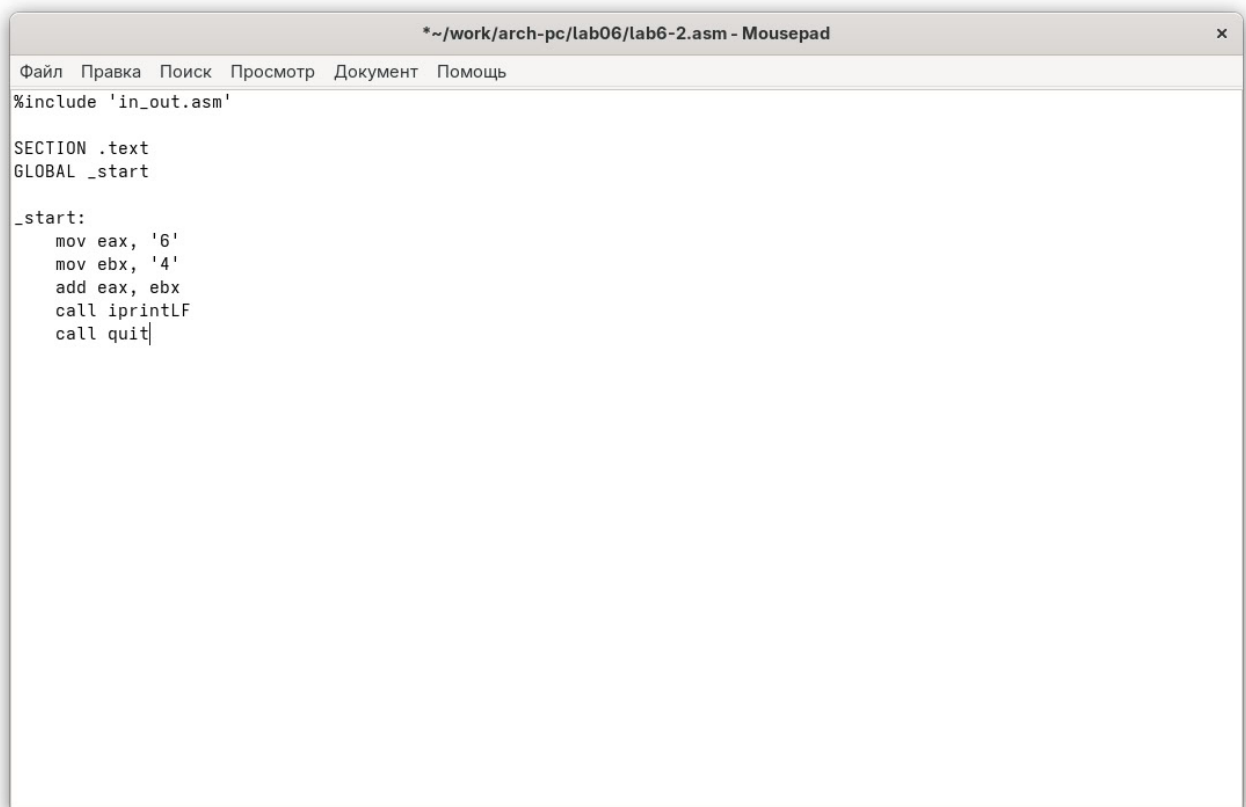
A terminal window with a dark background and light green text. The window title is 'dima1132250461@fedora:~/work/arch-pc/lab06'. The terminal shows three commands being executed in sequence: 'nasm -f elf lab6-1.asm', 'ld -m elf_i386 -o lab6-1 lab6-1.o', and './lab6-1'. The prompt 'dima1132250461@fedora:~/work/arch-pc/lab06\$' is visible at the start of each line.

```
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-1

dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 5: Запуск измененной программы

Создаю новый файл для будущей программы и записываю в нее код из листинга (рис. 6).

A screenshot of a text editor window titled '*~/work/arch-pc/lab06/lab6-2.asm - Mousepad'. The window has a menu bar with 'Файл', 'Правка', 'Поиск', 'Просмотр', 'Документ', and 'Помощь'. The code inside is as follows:

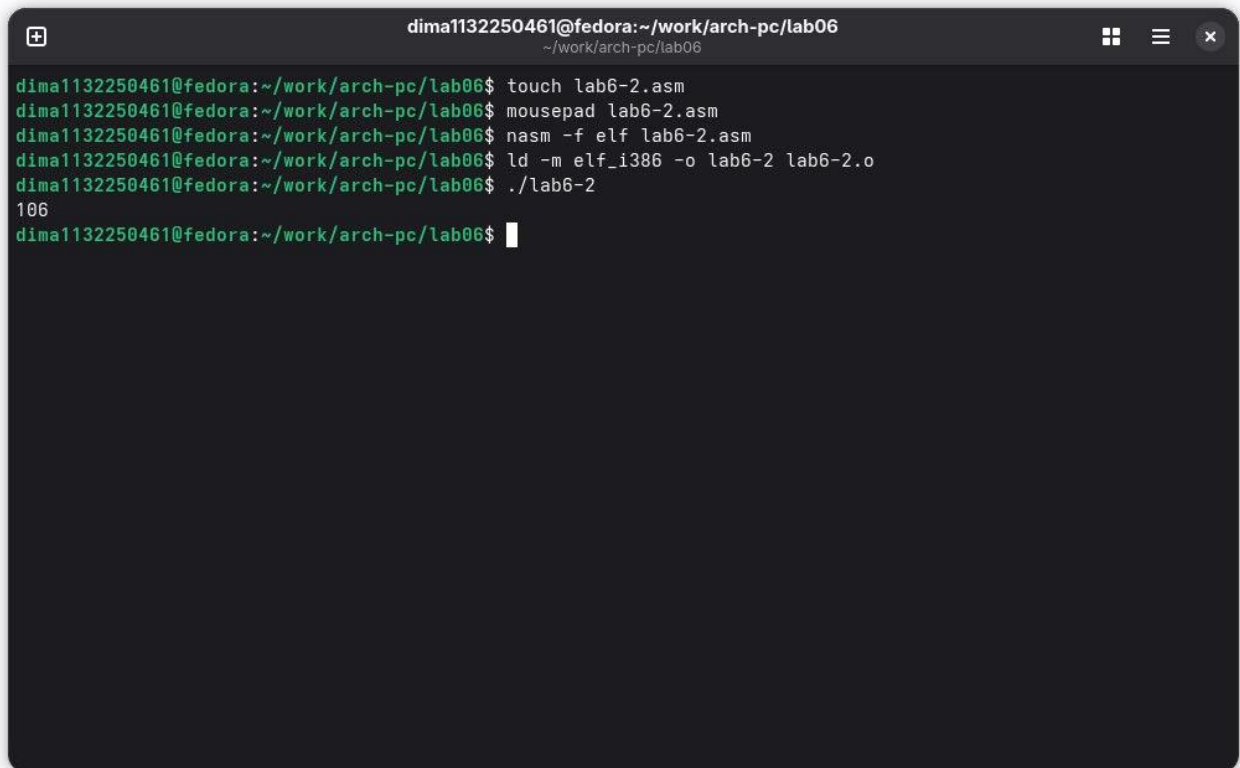
```
%include 'in_out.asm'

SECTION .text
GLOBAL _start

_start:
    mov eax, '6'
    mov ebx, '4'
    add eax, ebx
    call iprintLF
    call quit
```

Рис. 6: Вторая программа

Создаю исполняемый файл и запускаю его, теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на `iprintLF` (рис. 7).

A terminal window with a dark background and light green text. The window title is 'dima1132250461@fedora:~/work/arch-pc/lab06'. The terminal shows a sequence of commands: 'touch lab6-2.asm', 'mousepad lab6-2.asm', 'nasm -f elf lab6-2.asm', 'ld -m elf_i386 -o lab6-2 lab6-2.o', and './lab6-2'. The output of the last command is '106'. The prompt 'dima1132250461@fedora:~/work/arch-pc/lab06\$' is visible at the end of the line.

```
dima1132250461@fedora:~/work/arch-pc/lab06$ touch lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ mousepad lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 7: Вывод второй программы

Убрав кавычки в программе, я снова ее запускаю и получаю предполагаемый изначально результат. (рис. 8).

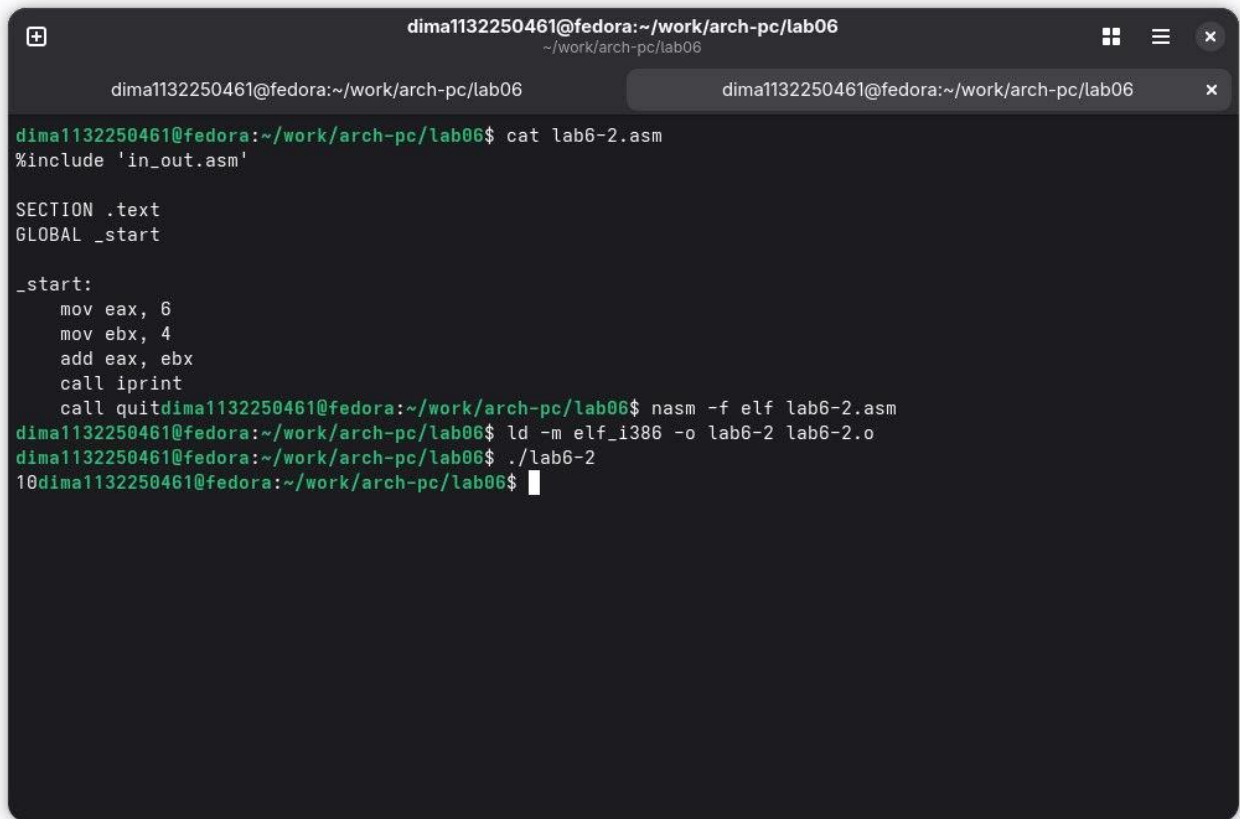

```
dima1132250461@fedora:~/work/arch-pc/lab06
~/work/arch-pc/lab06
dima1132250461@fedora:~/work/arch-pc/lab06$ touch lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ mousepad lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
dima1132250461@fedora:~/work/arch-pc/lab06$ mousepad lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
dima1132250461@fedora:~/work/arch-pc/lab06$ cat lab6-2.asm
#include 'in_out.asm'

SECTION .text
GLOBAL _start

_start:
    mov eax, 6
    mov ebx, 4
    add eax, ebx
    call iprintLF
    call quitdima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 8: Вывод измененной второй программы

Заменив функцию вывода на `iprint`, я получаю тот же результат, но без переноса строки (рис. 9).



```
dima1132250461@fedora:~/work/arch-pc/lab06
~/work/arch-pc/lab06
dima1132250461@fedora:~/work/arch-pc/lab06
dima1132250461@fedora:~/work/arch-pc/lab06$ cat lab6-2.asm
#include 'in_out.asm'

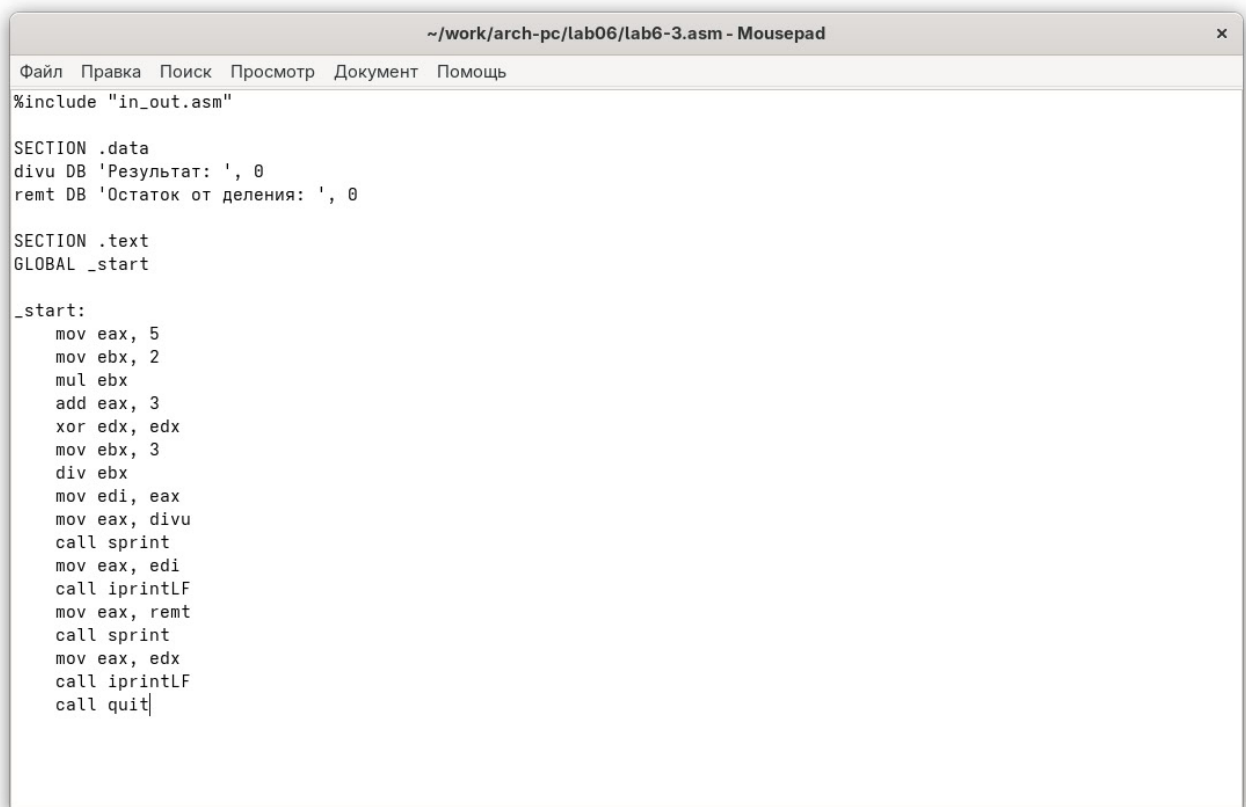
SECTION .text
GLOBAL _start

_start:
    mov eax, 6
    mov ebx, 4
    add eax, ebx
    call iprint
    call quitdima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-2
10dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 9: Замена функции вывода во второй программе

4.2 Выполнение арифметических операций в NASM

Создаю новый файл и копирую в него содержимое листинга (рис. 10).



```
~/work/arch-pc/lab06/lab6-3.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

#include "in_out.asm"

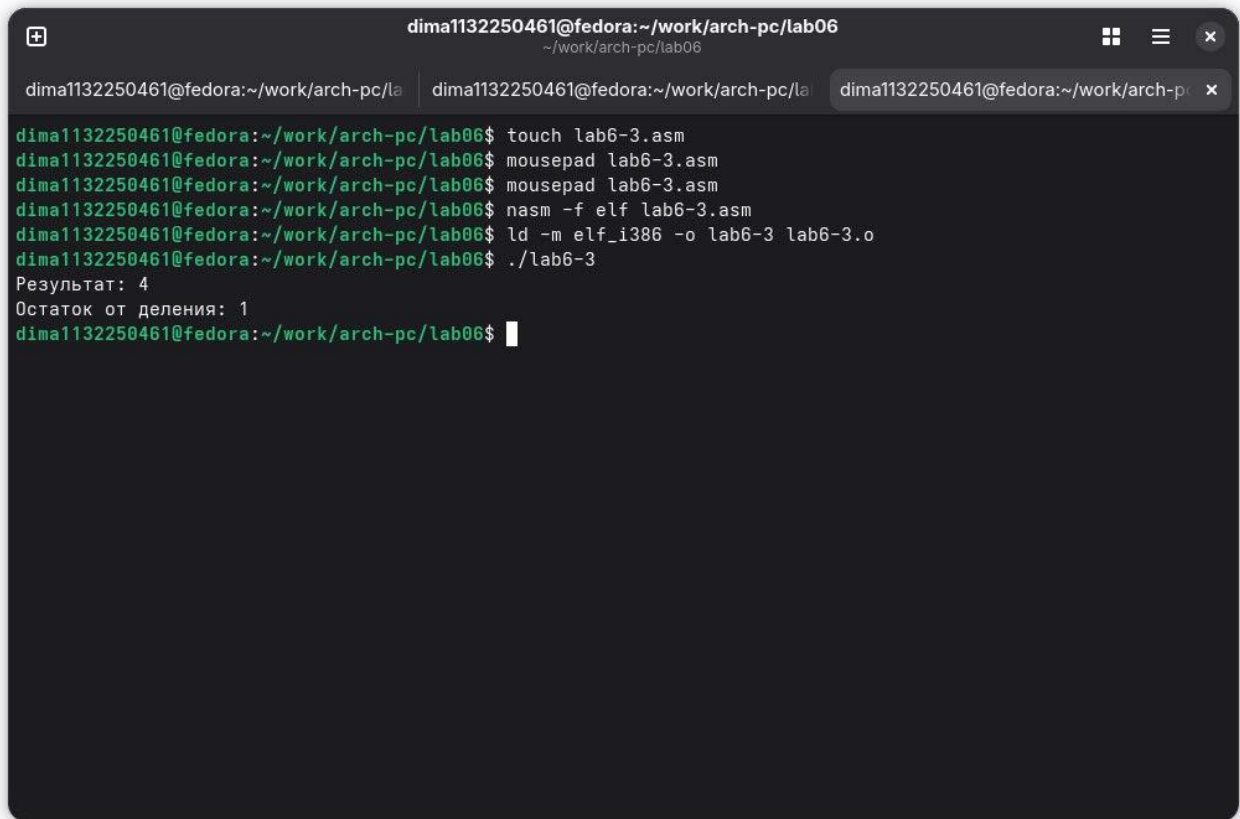
SECTION .data
divu DB 'Результат: ', 0
remt DB 'Остаток от деления: ', 0

SECTION .text
GLOBAL _start

_start:
    mov eax, 5
    mov ebx, 2
    mul ebx
    add eax, 3
    xor edx, edx
    mov ebx, 3
    div ebx
    mov edi, eax
    mov eax, divu
    call sprint
    mov eax, edi
    call iprintLF
    mov eax, remt
    call sprint
    mov eax, edx
    call iprintLF
    call quit
```

Рис. 10: Третья программа

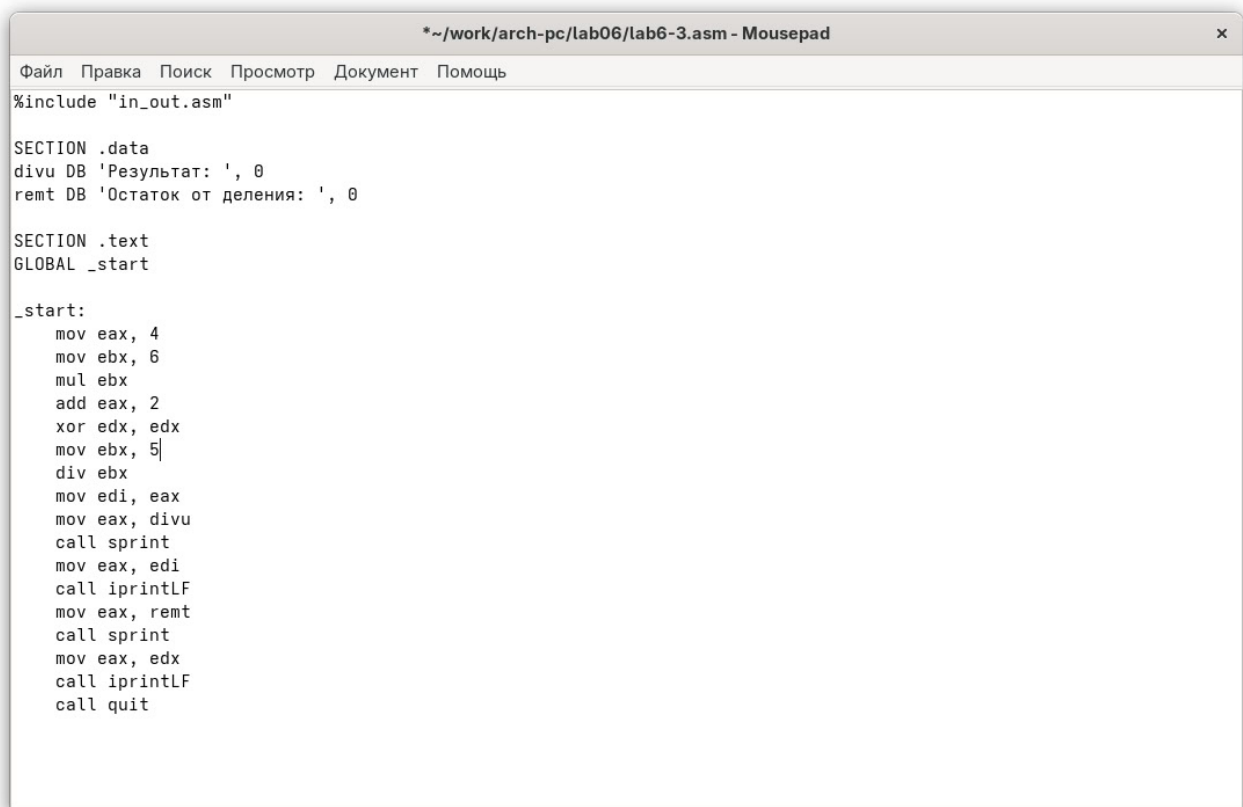
Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления (рис. 11).



```
dima1132250461@fedora:~/work/arch-pc/lab06
~/work/arch-pc/lab06
dima1132250461@fedora:~/work/arch-pc/la  dima1132250461@fedora:~/work/arch-pc/la  dima1132250461@fedora:~/work/arch-p
dima1132250461@fedora:~/work/arch-pc/lab06$ touch lab6-3.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ mousepad lab6-3.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ mousepad lab6-3.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 11: Запуск третьей программы

Заменив переменные в программе для выражения $f(x) = (4*6+2)/5$ (рис. 12).



```
*~/work/arch-pc/lab06/lab6-3.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include "in_out.asm"

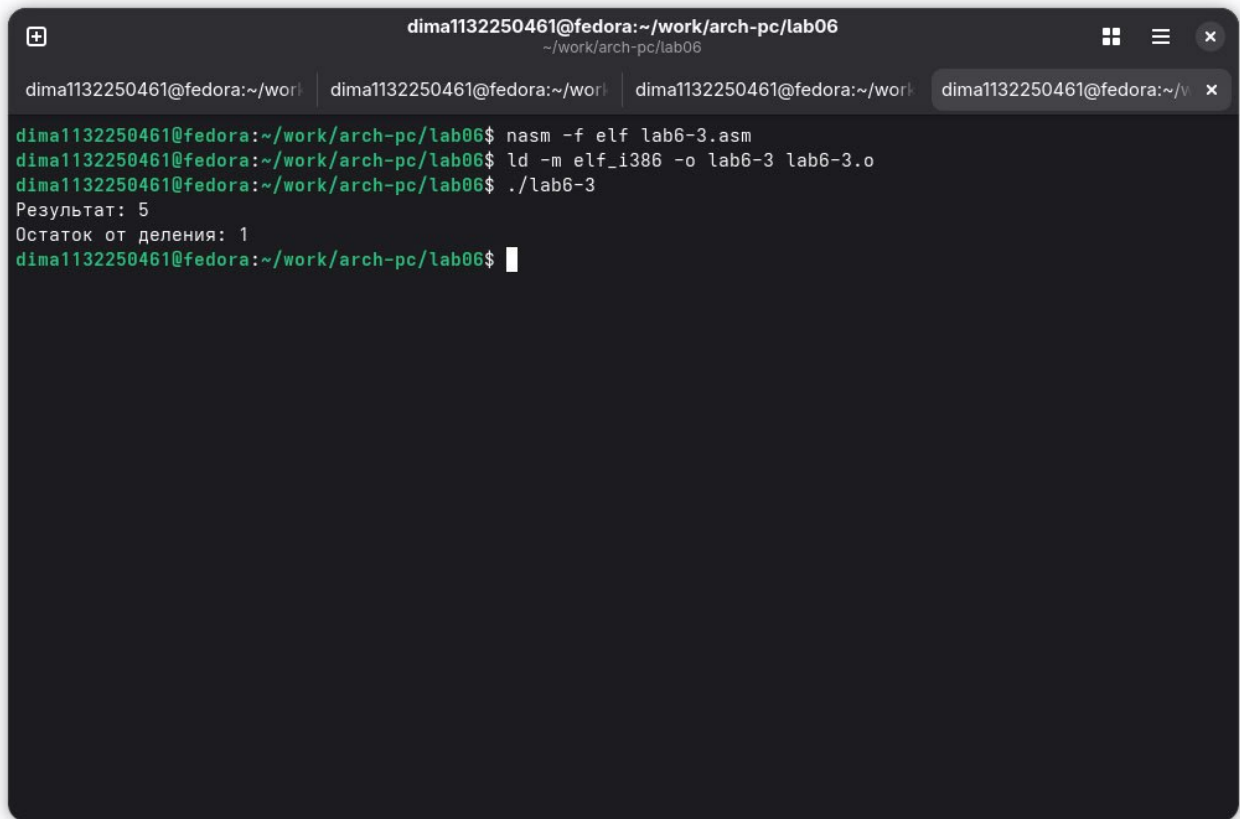
SECTION .data
divu DB 'Результат: ', 0
remt DB 'Остаток от деления: ', 0

SECTION .text
GLOBAL _start

_start:
    mov eax, 4
    mov ebx, 5
    mul ebx
    add eax, 2
    xor edx, edx
    mov ebx, 5
    div ebx
    mov edi, eax
    mov eax, divu
    call sprint
    mov eax, edi
    call iprintLF
    mov eax, remt
    call sprint
    mov eax, edx
    call iprintLF
    call quit
```

Рис. 12: Изменение третьей программы

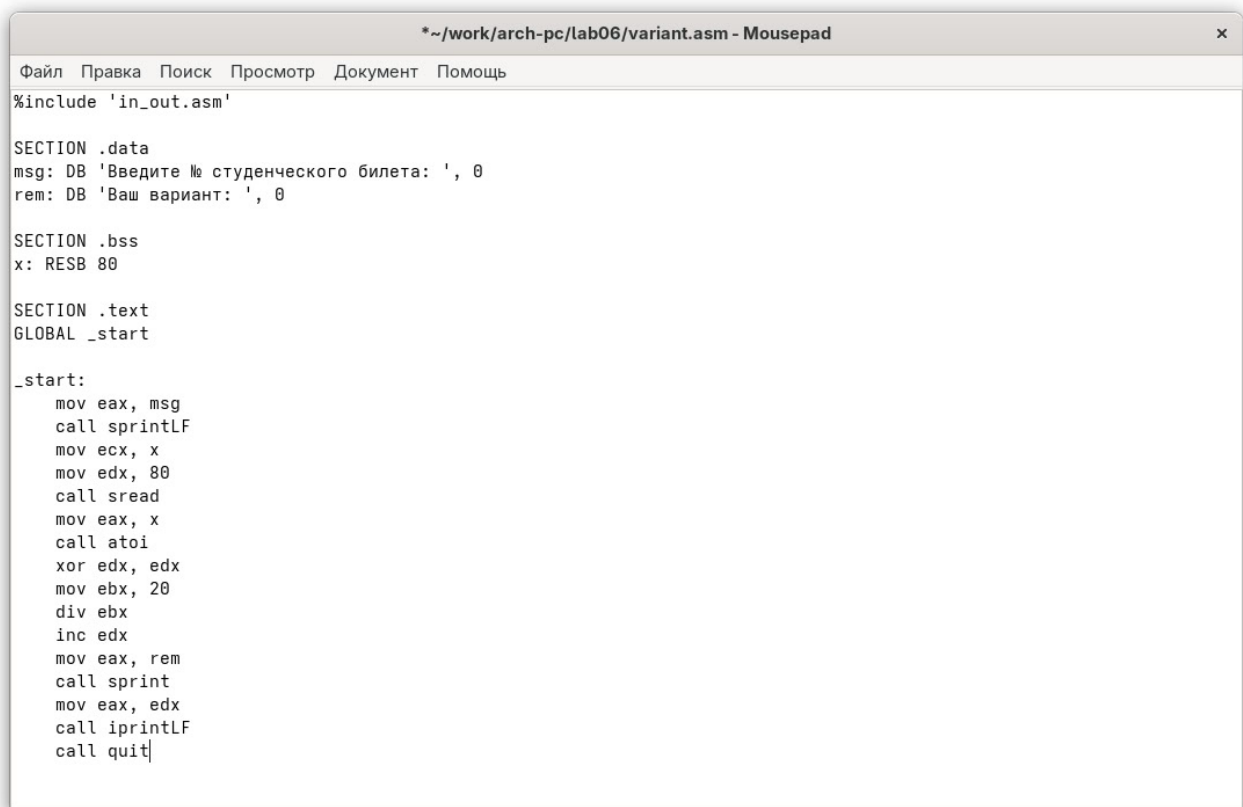
Запуск программы дает корректный результат (рис. 13).

A terminal window with a dark background and light green text. The window title is 'dima1132250461@fedora:~/work/arch-pc/lab06'. The terminal shows three lines of commands: 'nasm -f elf lab6-3.asm', 'ld -m elf_i386 -o lab6-3 lab6-3.o', and './lab6-3'. The output shows 'Результат: 5' and 'Остаток от деления: 1'.

```
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 13: Запуск измененной третьей программы

Создаю новый файл и помещаю текст из листинга (рис. 14).



```
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите № студенческого билета: ', 0
rem: DB 'Ваш вариант: ', 0

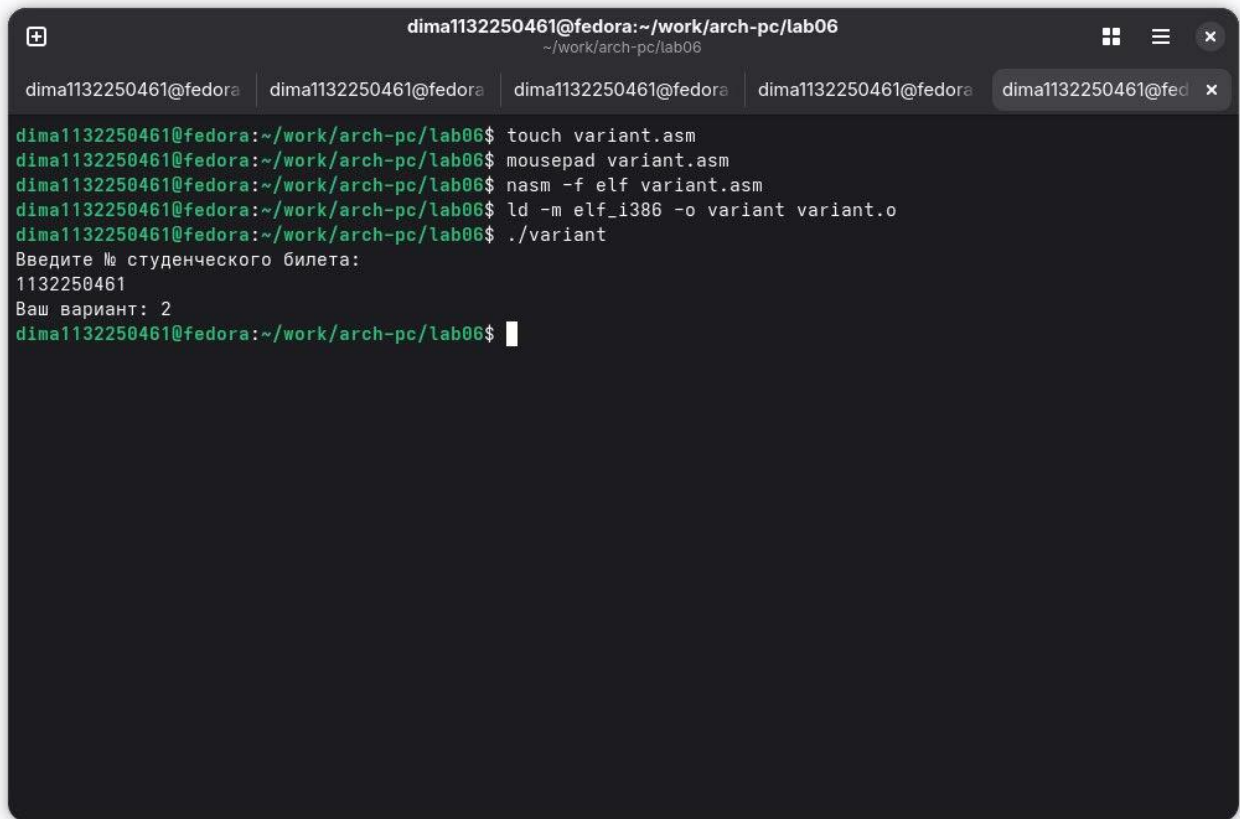
SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start

_start:
    mov eax, msg
    call sprintf
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    xor edx, edx
    mov ebx, 20
    div ebx
    inc edx
    mov eax, rem
    call sprintf
    mov eax, edx
    call iprintLF
    call quit
```

Рис. 14: Программа для подсчета варианта

Запустив программу и указав свой номер студенческого билета, я получил свой вариант для дальнейшей работы. (рис. 15).



```
dima1132250461@fedora:~/work/arch-pc/lab06
dima1132250461@fedora:~/work/arch-pc/lab06$ touch variant.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ mousepad variant.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132250461
Ваш вариант: 2
dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 15: Запуск программы для подсчета варианта

4.3 Ответы на контрольные вопросы

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

```
mov eax,rem
call sprint
```

2. Инструкция `mov esx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `esx` `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
```

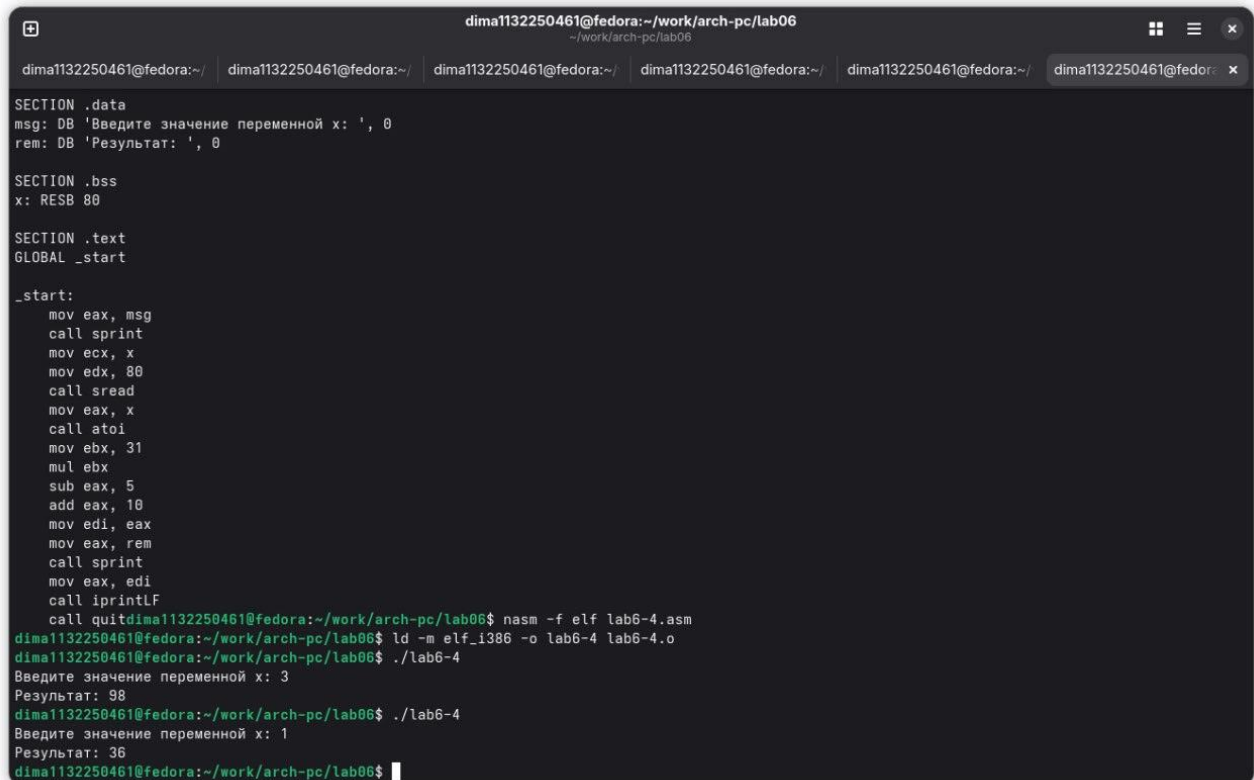
5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.

6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1.
7. За вывод на экран результатов вычислений отвечают строки:

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

4.4 Задание для самостоятельной работы

В соответствии с выбранным вариантом, я реализую программу для подсчета функции $f(x) = 10 + (31x - 5)$, проверка на нескольких переменных показывает корректное выполнение программы (рис. 16).



```
dima1132250461@fedora:~/work/arch-pc/lab06
SECTION .data
msg: DB 'Введите значение переменной x: ', 0
rem: DB 'Результат: ', 0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start

_start:
    mov eax, msg
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov ebx, 31
    mul ebx
    sub eax, 5
    add eax, 10
    mov edi, eax
    mov eax, rem
    call sprint
    mov eax, edi
    call iprintLF
    call quit
dima1132250461@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
dima1132250461@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 3
Результат: 98
dima1132250461@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 36
dima1132250461@fedora:~/work/arch-pc/lab06$
```

Рис. 16: Запуск и проверка программы

Прилагаю код своей программы:

```
%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:
mov eax, msg
call sprint
```

```
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 31
mul ebx
sub eax, 5
add eax, 10
mov edi, eax
mov eax, rem
call sprint
mov eax, edi
call iprint
```

5 Выводы

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

6 Список литературы

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №6
4. Программирование на языке ассемблера NASM Столяров А. В.