

Отчет по выполнению лабораторной работы №6

Дисциплина: Архитектура компьютера

Ефремова Полина Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Символьные и численные данные в NASM.	9
4.2	Выполнение арифметических операций в NASM	16
4.2.1	Ответы на вопросы по программе	23
4.3	Выполнение заданий для самостоятельной работы.	24
5	Выводы	28
	Список литературы	29

Список иллюстраций

4.1	Создание каталога и файла 1 для работы	9
4.2	Ввод программы 1	10
4.3	Запуск файла 1	11
4.4	Изменение файла 1	12
4.5	Запуск файла 1	13
4.6	Создание файла 2	13
4.7	Ввод программы 2	14
4.8	Запуск файла 2	15
4.9	Изменение файла 2	15
4.10	Запуск измененного файла 2	16
4.11	Создание файла 3	16
4.12	Ввод программы 3	17
4.13	Запуск файла 3	17
4.14	Изменение файла 3	19
4.15	Запуск измененного файла 3	20
4.16	Создание файла variant.asm	21
4.17	Ввод программы	22
4.18	Запуск файла	23
4.19	Создание файла lab6-4.asm	24
4.20	Ввод программы в файл lab6-4.asm	25
4.21	Запуск файла lab6-4.asm	27

Список таблиц

1 Цель работы

Основная цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Изучение теоретического материала по заданной теме
2. Определение существующих арифметических инструкций через ассемблер.
3. Работа с примерами кодов, решающих математические выражения.
4. Создание собственного кода.

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.

Поговорим про арифметические операции.

- 1) Схема команды целочисленного сложения `add` (*от англ. addition - добавление*) выполняет сложение двух операндов и записывает результат по адресу первого операнда.
- 2) Команда целочисленного вычитания `sub` (*от англ. subtraction – вычитание*) работает аналогично команде `add`.
- 3) Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкре-

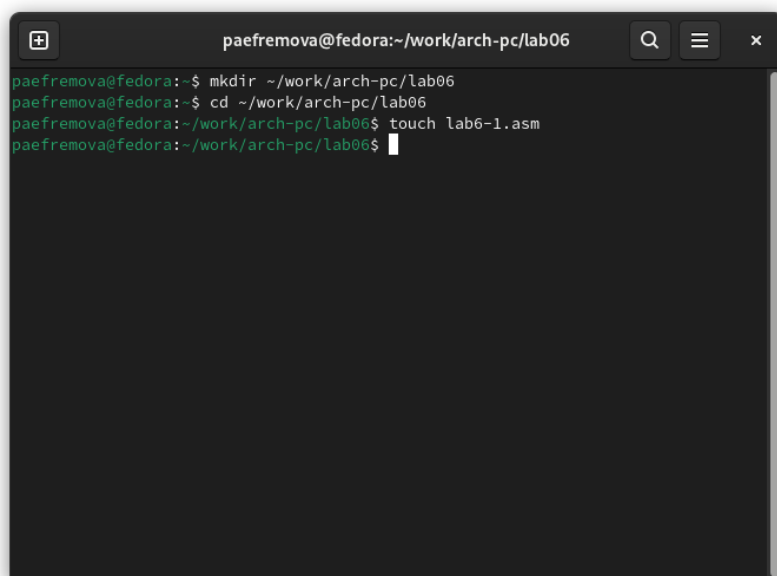
ментом, а вычитание — декрементом. Для этих операций существуют специальные команды: `inc` (*от англ. increment*) и `dec` (*от англ. decrement*), которые увеличивают и уменьшают на 1 свой операнд.

- 4) Команда `neg` рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.
- 5) Для беззнакового умножения используется команда `mul` (*от англ. multiply – умножение*). Для знакового умножения используется команда `imul`.
- 6) Для деления, как и для умножения, существует 2 команды `div` (*от англ. divide – деление*) и `idiv`

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM.

1. Создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm (рис. 4.1).



```
paefremova@fedora:~/work/arch-pc/lab06
paefremova@fedora:~$ mkdir ~/work/arch-pc/lab06
paefremova@fedora:~$ cd ~/work/arch-pc/lab06
paefremova@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
paefremova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание каталога и файла 1 для работы

2. Ввожу в файл lab6-1.asm текст программы из листинга 6.1. (рис. 4.2).

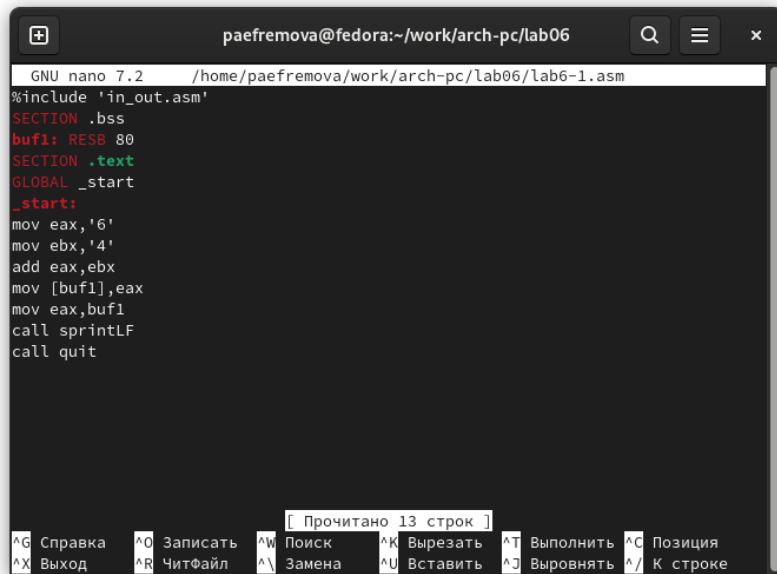
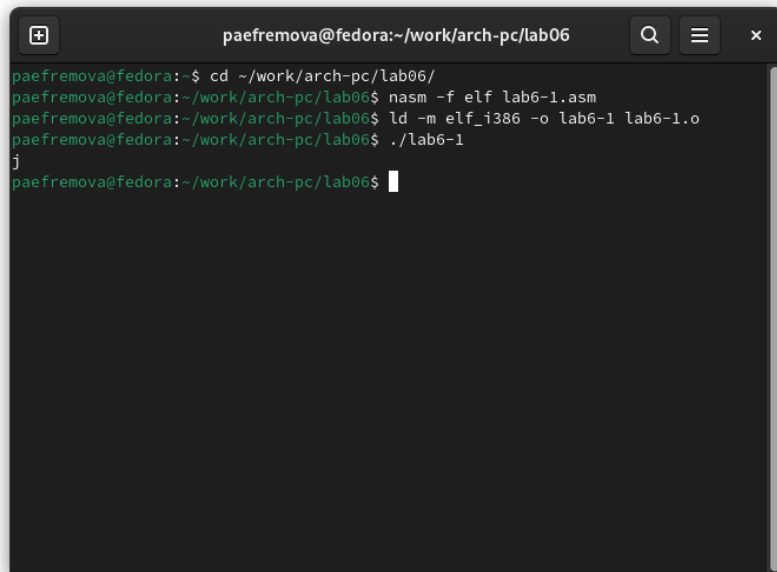


Рис. 4.2: Ввод программы 1

Ниже прикрепляю текст программы:

```
%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 printf
call quit
```

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

A terminal window titled 'paefremova@fedora:~/work/arch-pc/lab06'. The window shows the following commands and output:

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

Рис. 4.3: Запуск файла 1

Вывод: В данном случае при выводе значения регистра `eax` вместо числа 10 мы видим символ `j` как результат программы. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении, а код символа 4 – 00110100. Команда `add eax,ebx` запишет в регистр `eax` сумму кодов – 01101010 (106), что в свою очередь является кодом символа `j`.

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



Рис. 4.4: Изменение файла 1

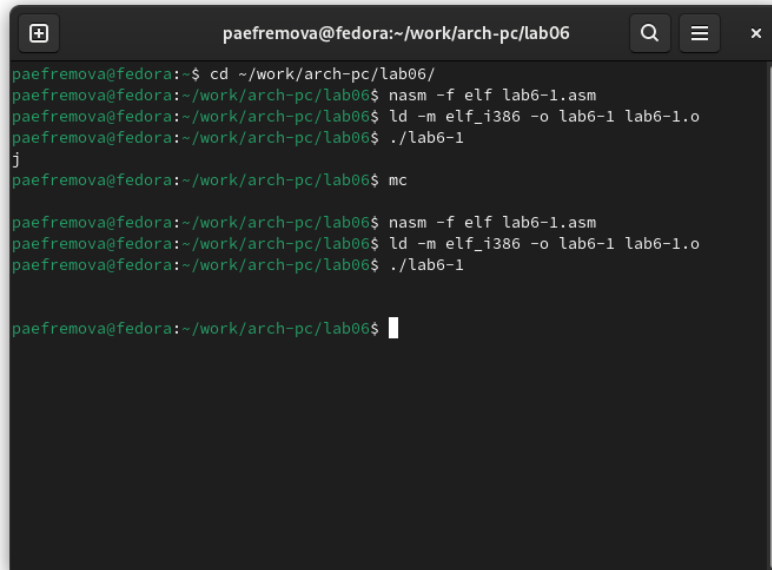
Текст программы:

```

#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

```

5. Создаю исполняемый файл и запускаю его. (рис. 4.5).



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

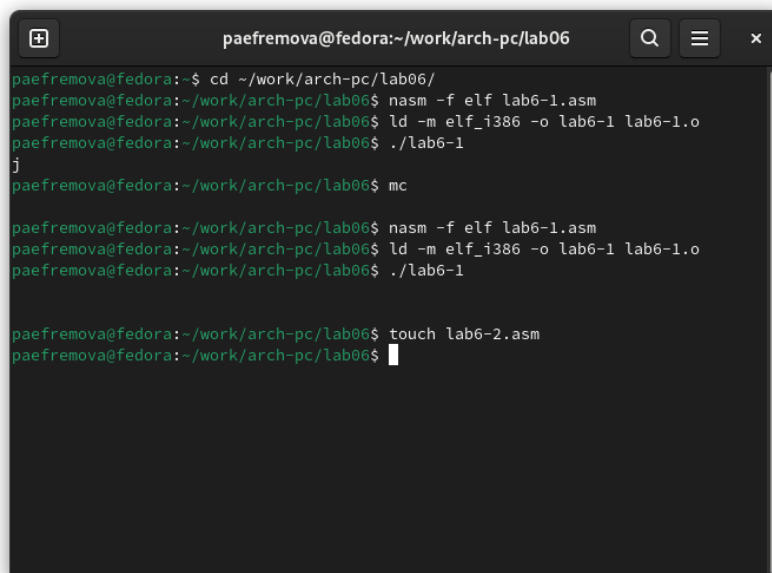
paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-1

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

Рис. 4.5: Запуск файла 1

Вывод: Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

6. Создаю файл lab6-2.asm. (рис. 4.6).



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

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

paefremova@fedora:~/work/arch-pc/lab06$ touch lab6-2.asm
paefremova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.6: Создание файла 2

7. Ввожу в файл 2 текст программы из листинга 6.2. (рис. 4.7).



The screenshot shows a terminal window with the GNU nano 7.2 text editor. The title bar indicates the user is 'paefremova@fedora' in the directory '~/work/arch-pc/lab06'. The editor is editing a file named 'lab6-2.asm'. The code entered is as follows:

```
GNU nano 7.2 /home/paefremova/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

At the bottom of the window, there is a status bar with the text '[Прочитано 9 строк]' and a list of keyboard shortcuts: ^G Справка, ^O Записать, ^W Поиск, ^K Вырезать, ^T Выполнить, ^C Позиция, ^X Выход, ^R ЧитФайл, ^\ Замена, ^U Вставить, ^J Выровнять, ^/ К строке.

Рис. 4.7: Ввод программы 2

Текст программы вывода значения регистра `eax`.

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

8. Создаю исполняемый файл 2 и запускаю его. (рис. 4.8).

```

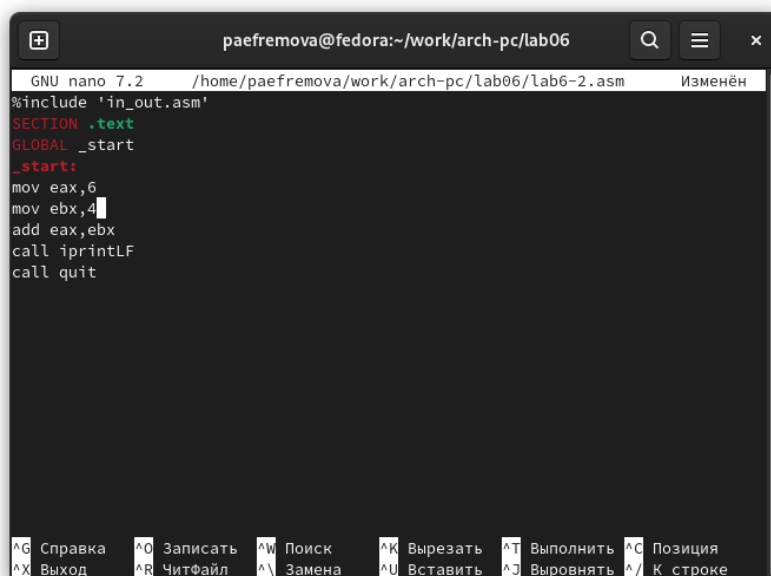
paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
paefremova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.8: Запуск файла 2

В результате работы данной программы я получила число 106. Данная программа схожа с предыдущей, только здесь `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

9. Аналогично предыдущему примеру изменяю символы на числа. (рис. 4.9).



```

GNU nano 7.2 /home/paefremova/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit

```

Рис. 4.9: Изменение файла 2

Текст измененной программы файла 2:

```

#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6

```

```

mov ebx,4
add eax,ebx
call iprintLF
call quit

```

10. Создаю исполняемый измененный файл 2 и запускаю его. (рис. 4.10).

```

paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-1

paefremova@fedora:~/work/arch-pc/lab06$ touch lab6-2.asm
paefremova@fedora:~/work/arch-pc/lab06$ mc

paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-1.o
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-1

paefremova@fedora:~/work/arch-pc/lab06$ ^C
paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
paefremova@fedora:~/work/arch-pc/lab06$ mc

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

```

Рис. 4.10: Запуск измененного файла 2

Вывод: Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

4.2 Выполнение арифметических операций в NASM

11. Создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. 4.11).

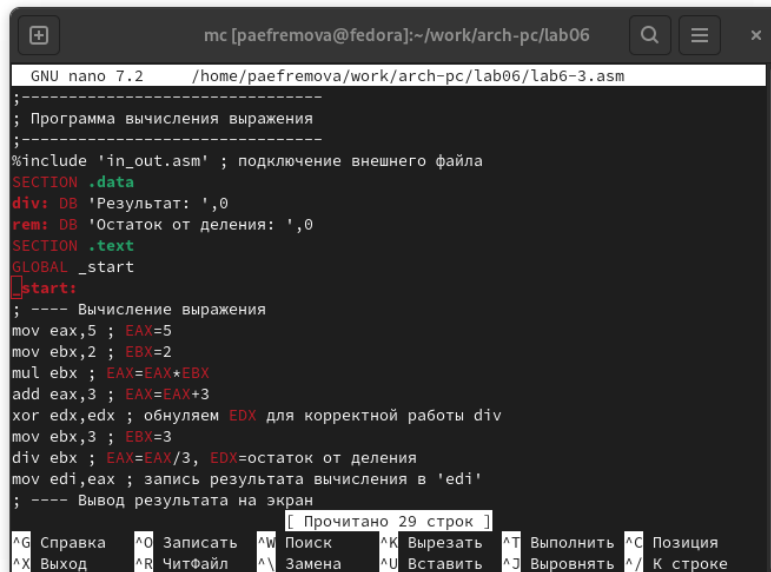
```

10
paefremova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
paefremova@fedora:~/work/arch-pc/lab06$ mc

```

Рис. 4.11: Создание файла 3

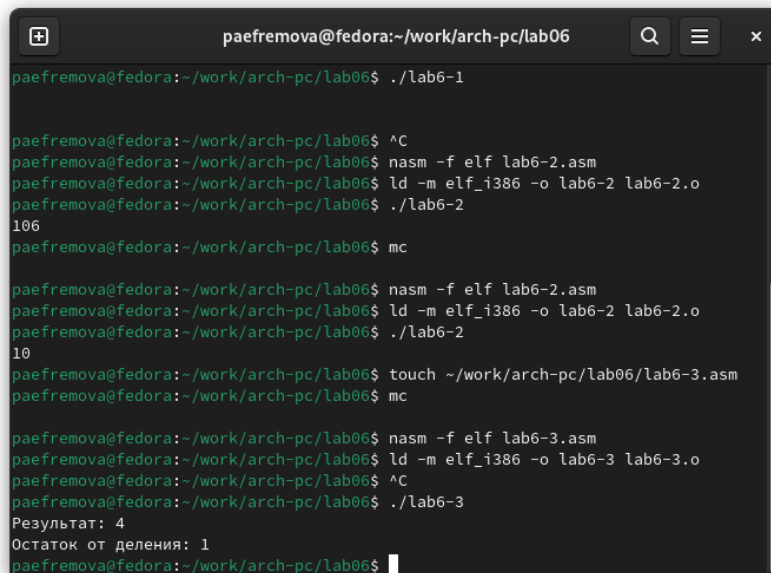
12. Внимательно изучаю текст программы из листинга 6.3 и ввожу в lab6-3.asm (рис. 4.12).



```
GNU nano 7.2 /home/paefremova/work/arch-pc/lab06/lab6-3.asm
;-----
; Программа вычисления выражения
;-----
#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
[ Прочитано 29 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^J Вставить ^_ Выровнять ^/_ К строке
```

Рис. 4.12: Ввод программы 3

13. Создаю исполняемый файл 3 и запускаю его. (рис. 4.13).



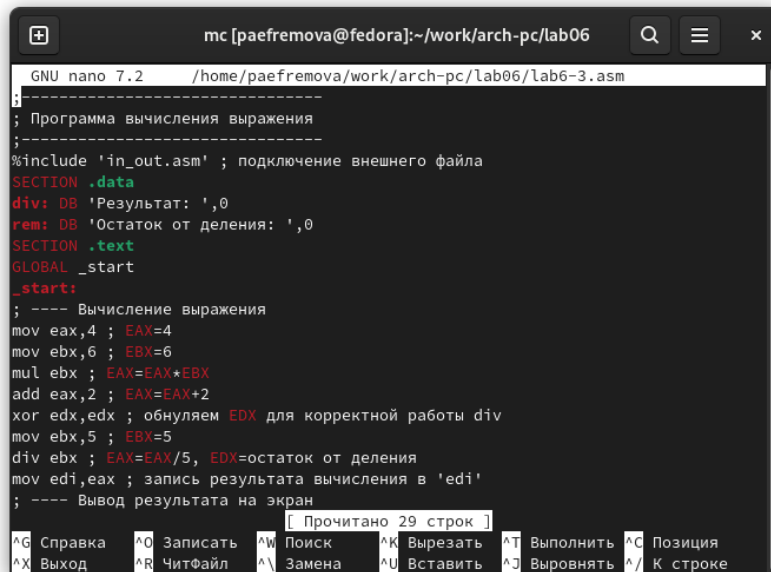
```
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-1
paefremova@fedora:~/work/arch-pc/lab06$ ^C
paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
paefremova@fedora:~/work/arch-pc/lab06$ mc
paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
paefremova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
paefremova@fedora:~/work/arch-pc/lab06$ mc
paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
paefremova@fedora:~/work/arch-pc/lab06$ ^C
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
paefremova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.13: Запуск файла 3

Текст программы 3: Программа вычисления выражения $f(x) = (5 * 2 + 3)/3$:

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: 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,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

14. Изменяю программу, чтобы она была способна вычислить выражение Программа вычисления выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.14).



```
mc [paefremova@fedora]~/.work/arch-pc/lab06
GNU nano 7.2 /home/paefremova/work/arch-pc/lab06/lab6-3.asm
;-----
; Программа вычисления выражения
;-----
#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 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
[ Прочитано 29 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке
```

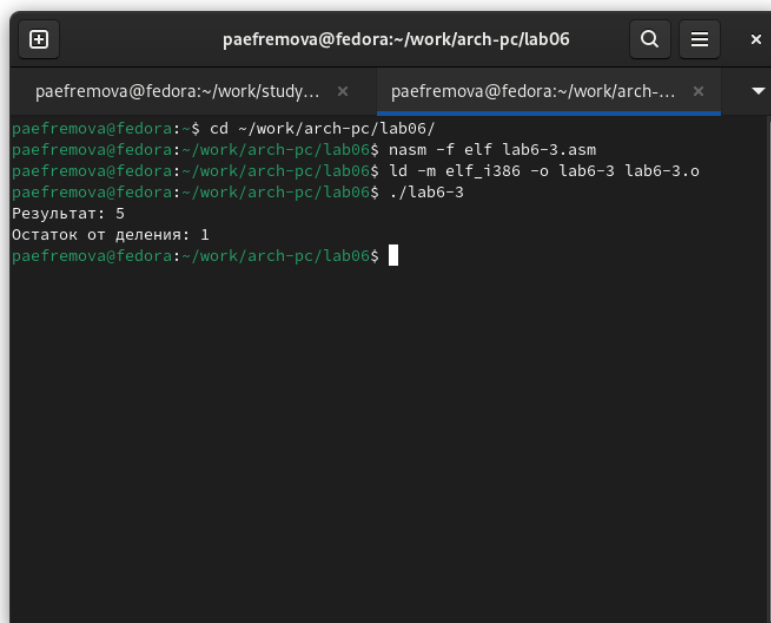
Рис. 4.14: Изменение файла 3

Текст измененной программы 3: Программа вычисления выражения $f(x) = (4 * 6 + 2) / 5$:

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
```

```
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

15. Создаю исполняемый измененный файл 3 и запускаю его. (рис. 4.15).



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

Рис. 4.15: Запуск измененного файла 3

16. Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06 (рис. 4.16).

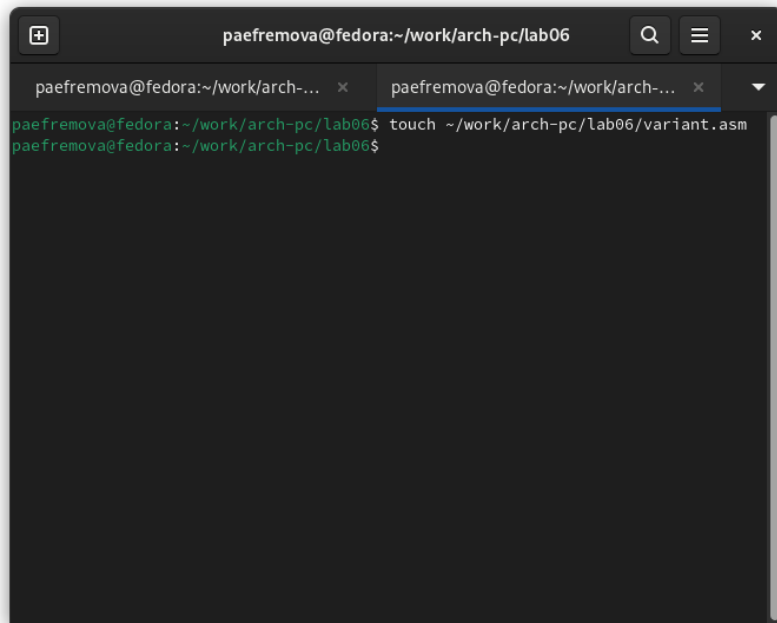
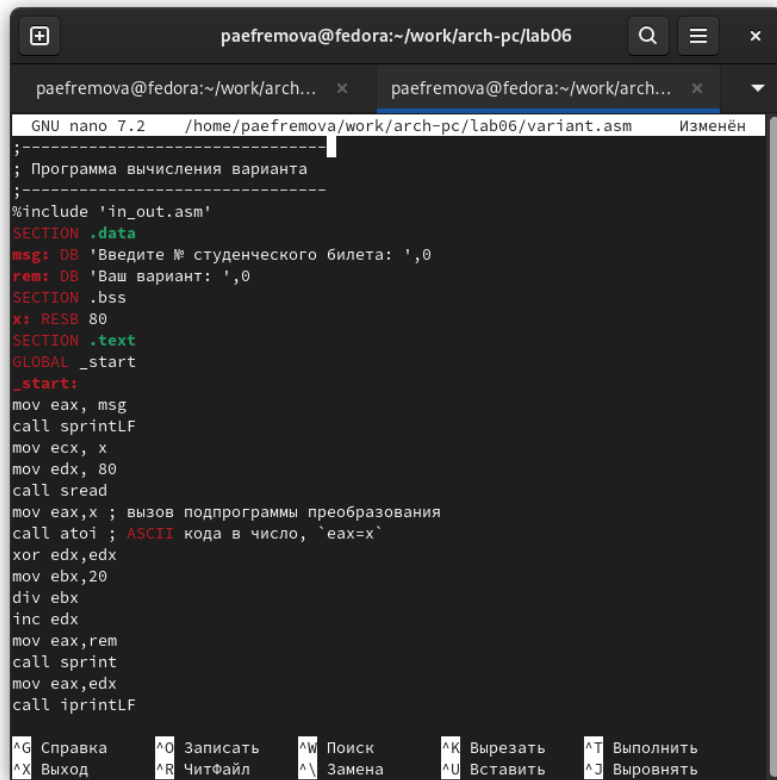


Рис. 4.16: Создание файла variant.asm

17. Внимательно изучаю текст программы из листинга 6.4 и ввожу в файл variant.asm. (рис. 4.17).

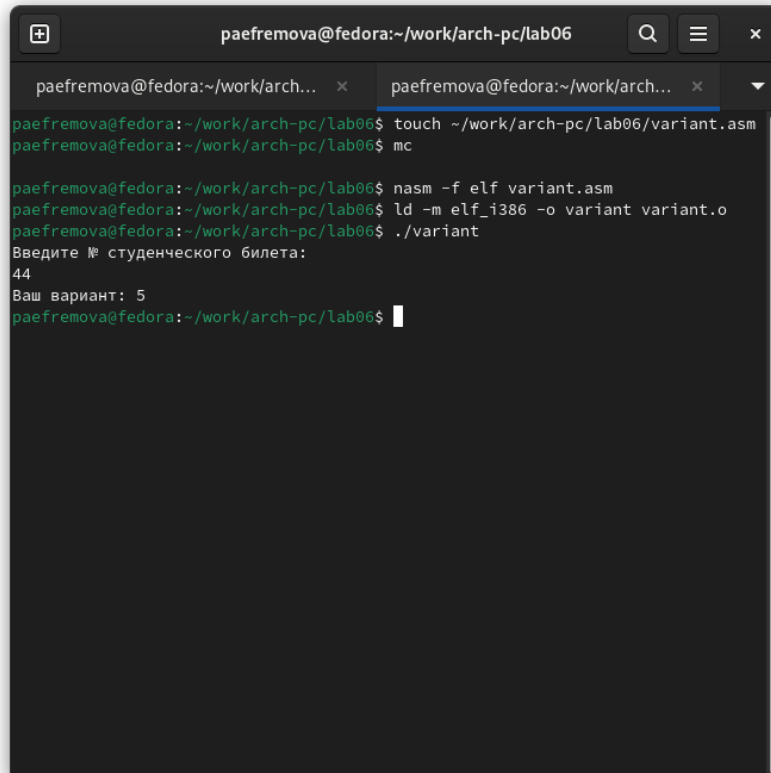


```
paefremova@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/paefremova/work/arch-pc/lab06/variant.asm
;-----
; Программа вычисления варианта
;-----
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; Вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintLF

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выводить
```

Рис. 4.17: Ввод программы

18. Создаю исполняемый измененный файл и запускаю его. (рис. 4.18).



```
paefremova@fedora:~/work/arch-pc/lab06
paefremova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
paefremova@fedora:~/work/arch-pc/lab06$ mc
paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
paefremova@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
44
Ваш вариант: 5
paefremova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.18: Запуск файла

Программа показывает, что в самостоятельной работе я буду писать программу для выражения из варианта 5.

4.2.1 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`

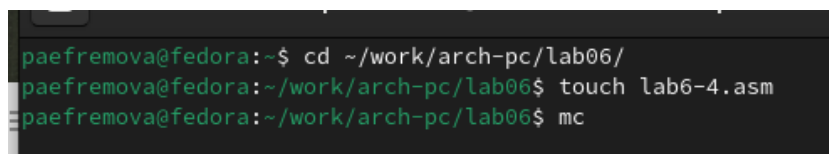
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1

7. За вывод на экран результатов вычислений отвечают строки:

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

4.3 Выполнение заданий для самостоятельной работы.

1. Создаю файл `lab6-4.asm` в том же каталоге, что и другие файлы `asm`. (рис. 4.19).



```
paefremova@fedora:~$ cd ~/work/arch-pc/lab06/
paefremova@fedora:~/work/arch-pc/lab06$ touch lab6-4.asm
paefremova@fedora:~/work/arch-pc/lab06$ mc
```

Рис. 4.19: Создание файла `lab6-4.asm`

2. Ввожу программу для выражения из варианта 5 $f(x) = (9x - 8)/8$ (рис. 4.20).

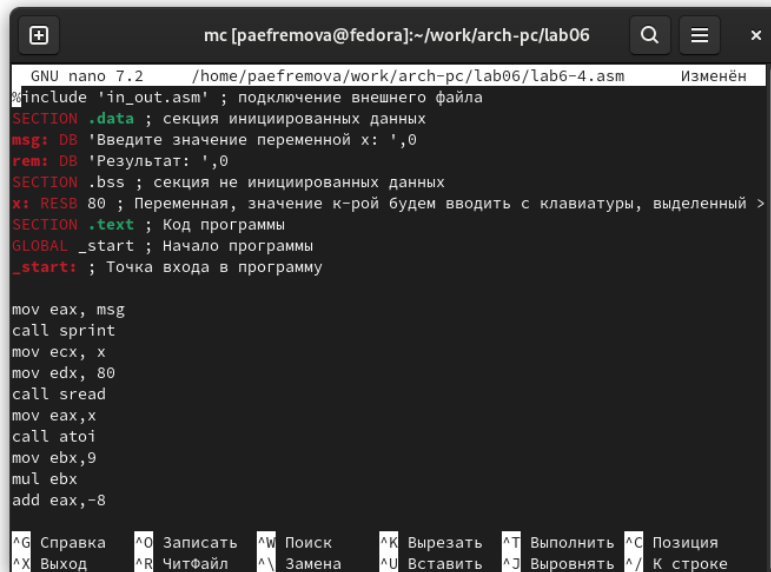


Рис. 4.20: Ввод программы в файл lab6-4.asm

Текст программы из файла lab6-4.asm:

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB 'Введите значение переменной x: ',0
res: DB 'Результат: ',0
SECTION .bss ; секция не инициализированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный >
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
  
```

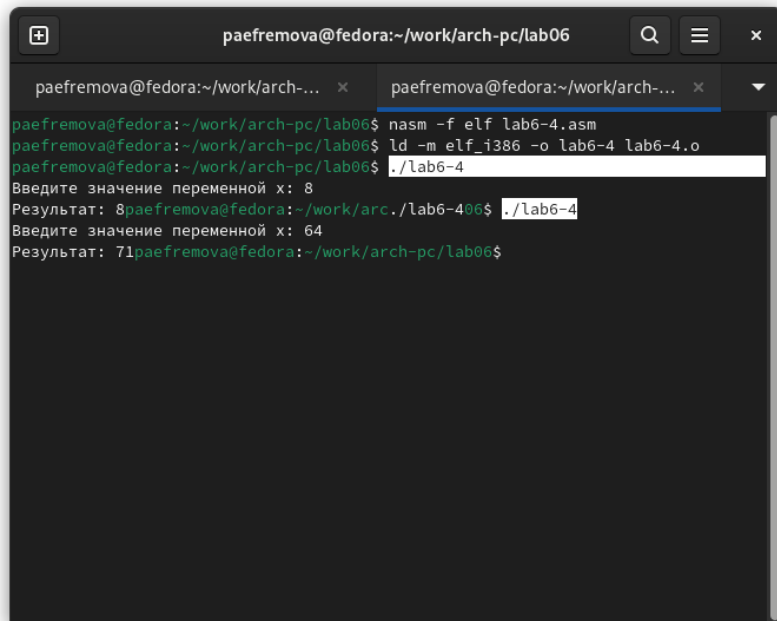
```

mov eax,x
call atoi
mov ebx,9
mul ebx
add eax,-8
mov ebx, 8
div ebx
mov edi,eax ; запись результата вычисления в 'edi'

mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

```

3. Создаю исполняемый измененный файл и запускаю его, проверяю его работу для значений x1 и x2 из 6.3. (рис. 4.21).



A terminal window titled 'paefremova@fedora:~/work/arch-pc/lab06' with search and menu icons. It shows the compilation of 'lab6-4.asm' using 'nasm' and 'ld' to create 'lab6-4.o' and then 'lab6-4'. The user then runs './lab6-4', which prompts for a variable value. The first run with value 8 results in 8. The second run with value 64 results in 71. The prompt is in Russian: 'Введите значение переменной x:'. The terminal has two tabs open, both showing the same directory path.

```
paefremova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
paefremova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 8
Результат: 8paefremova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 64
Результат: 71paefremova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.21: Запуск файла lab6-4.asm

5 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM. Я увидела разницу между символьными и численными данными в NASM, научилась выполнять арифметические операции, а также сама смогла написать программу нахождения значения выражения.

Список литературы

1. Архитектура ЭВМ
2. Синтаксис Markdown: подробная шпаргалка для веб-разработчиков / Skillbox Media
3. Руководство по NASM