

# **Отчёт по лабораторной работе №6**

**Дисциплина: Архитектура Компьютера**

Азарцова Вероника Валерьевна

# 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>  |
| 3.1      | Адресация в NASM . . . . .                                     | 7         |
| 3.2      | Арифметические операции в NASM . . . . .                       | 7         |
| 3.3      | Перевод символа числа в десятичную символьную запись . . . . . | 9         |
| <b>4</b> | <b>Выполнение лабораторной работы</b>                          | <b>11</b> |
| 4.1      | Символьные и численные данные в NASM . . . . .                 | 11        |
| 4.2      | Выполнение арифметических операций в NASM . . . . .            | 17        |
| <b>5</b> | <b>Задания для самостоятельной работы</b>                      | <b>23</b> |
| <b>6</b> | <b>Выводы</b>  | <b>26</b> |
|          | <b>Список литературы</b>                                       | <b>27</b> |

# Список иллюстраций

|      |  |    |
|------|--|----|
| 4.1  | Создание каталога для лабораторной работы и lab6-1.asm . . . . . | 11 |
| 4.2  | Интерфейс Midnight Commander . . . . .                           | 12 |
| 4.3  | Текст программы в lab6-1.asm . . . . .                           | 12 |
| 4.4  | Запуск lab6-1 . . . . .  | 13 |
| 4.5  | Измененный текст lab6-1.asm . . . . .                            | 13 |
| 4.6  | Запуск измененного lab6-1 . . . . .                              | 14 |
| 4.7  | Создание lab6-2.asm . . . . .                                    | 14 |
| 4.8  | Текст программы в lab6-2.asm . . . . .                           | 15 |
| 4.9  | Запуск lab6-2 . . . . .  | 15 |
| 4.10 | Измененный текст программы в lab6-2.asm . . . . .                | 16 |
| 4.11 | Запуск измененного lab6-2 . . . . .                              | 16 |
| 4.12 | Замена iprintLF на iprint . . . . .                              | 17 |
| 4.13 | Запуск еще раз измененного lab6-2 . . . . .                      | 17 |
| 4.14 | Создание lab6-3.asm . . . . .                                    | 18 |
| 4.15 | Создание lab6-3.asm . . . . .                                    | 18 |
| 4.16 | Запуск lab6-3 . . . . .  | 19 |
| 4.17 | Измененный текст программы в lab6-3.asm . . . . .                | 19 |
| 4.18 | Запуск измененного lab6-3 . . . . .                              | 20 |
| 4.19 | Создание variant.asm . . . . .                                   | 20 |
| 4.20 | Текст программы в variant.asm . . . . .                          | 21 |
| 4.21 | Запуск измененного lab6-3 . . . . .                              | 21 |
| 5.1  | Создание var12.asm . . . . .                                     | 23 |
| 5.2  | Текст программы в var12.asm . . . . .                            | 24 |
| 5.3  | Запуск var12.asm . . . . .                                       | 25 |

## Список таблиц

|     |   |   |
|-----|---|---|
| 3.1 | Регистры, используемые командами умножения в NASM . . . . . | 8 |
| 3.2 | Регистры, используемые командами деления в NASM . . . . .   | 9 |

# 1 Цель работы

Цель лабораторной работы - освоить арифметические инструкции в языке ассемблера NASM.

## **2 Задание**

1. Ознакомление с теоретическим введением
2. Выполнение лабораторной работы
3. Выполнение заданий для самостоятельной работы

## 3 Теоретическое введение

### 3.1 Адресация в NASM

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.

### 3.2 Арифметические операции в NASM

1. Схема команды целочисленного сложения `add` (addition) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака:  
`add (операнд_1), (операнд_2)`

2. Команда целочисленного вычитания `sub` (subtraction) работает аналогично команде `add`:  
`sub (операнд_1), (операнд_2)`
3. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: `inc` (increment) и `dec` (decrement), которые увеличивают и уменьшают на 1 свой операнд:  
`inc (операнд) dec (операнд)`
4. Команда изменения знака `neg`: `neg (операнд)`
5. Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды - `mul` (multiply) и `imul`:  
`mul (операнд)` - Беззнаковое умножение  
`imul (операнд)` - Знаковое умножение  
 Для команд умножения один из сомножителей указывается в команде и должен находиться в регистре или в памяти, но не может быть непосредственным операндом. Второй сомножитель в команде явно не указывается и должен находиться в регистре `EAX`, `AX` или `AL`, а результат помещается в регистры `EDX:EAX`, `DX:AX` или `AX`, в зависимости от размера операнда (табл. 3.1).

Таблица 3.1: Регистры, используемые командами умножения в NASM

| Размер операнда | Неявный множитель | Результат умножения |
|-----------------|-------------------|---------------------|
| 1 Байт          | AL                | AX                  |
| 2 Байта         | AX                | DX:AX               |
| 3 Байта         | EAX               | EDX:EAX             |

6. Для деления, как и для умножения, существует две команды - `div` и `idiv`: `div (делитель)` - Беззнаковое деление `idiv (делитель)` - Знаковое деление



В командах указывается только один операнд – делитель, который может быть регистром или ячейкой памяти, но не может быть непосредственным операндом. Местоположение делимого и результата для команд деления зависит от размера делителя. Кроме того, так как в результате деления получается два числа – частное и остаток, то эти числа помещаются в определённые регистры (табл. 3.2).

Таблица 3.2: Регистры, используемые командами деления в NASM

| Размер операнда (делителя) | Делимое | Частное | Остаток |
|----------------------------|---------|---------|---------|
| 1 Байт                     | AX      | AL      | AH      |
| 2 Байта                    | DX:AX   | AX      | DX      |
| 3 Байта                    | EDX:EAX | EAX     |         |

### 3.3 Перевод символа числа в десятичную символьную запись

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. Для этого действия при выполнении лабораторных работ в файле `in_out.asm` реализованы подпрограммы. Это:

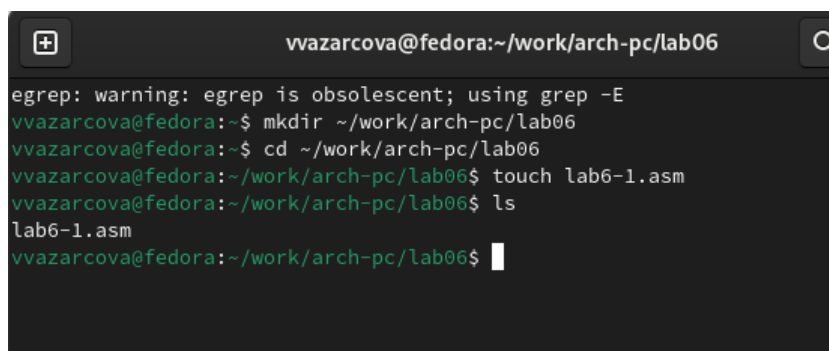
- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число - `mov eax,(int)`.
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки.
- `atoi` – функция преобразует `ascii`-код символа в целое число и запишет результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать

число - mov eax, (int)

## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

1. Создаю каталог для программы лабораторной работы №6, перехожу в него, создаю файл lab6-1.asm и проверяю действия с помощью ls (рис. 4.1).



```
egrep: warning: egrep is obsolescent; using grep -E
vvazarcova@fedora:~$ mkdir ~/work/arch-pc/lab06
vvazarcova@fedora:~$ cd ~/work/arch-pc/lab06
vvazarcova@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
lab6-1.asm
vvazarcova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание каталога для лабораторной работы и lab6-1.asm

2. Ввожу в файл lab6-1.asm текст программы, в которой в регистр еах записывается символ 6 (mov еах,'6'), в регистр еbх символ 4 (mov еbх,'4'), к значению в регистре еах прибавляется значение регистра еbх (add еах,еbх, результат сложения запишется в регистр еах), потом выводится результат. Так как для работы функции sprintLF в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Значение регистра еах записывается в переменную buf1 (mov [buf1],еах), а затем адрес переменной buf1 записывается в регистр еах (mov еах,buf1) и вызывается функция sprintLF.  
Для этого, открываю Midnight Commander с помощью команды mc (рис. 4.2).

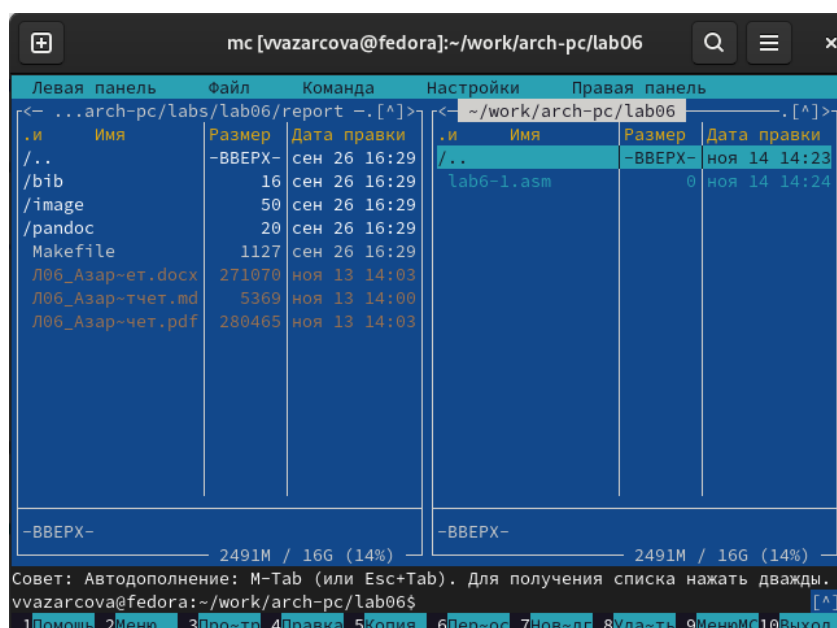


Рис. 4.2: Интерфейс Midnight Commander

Далее, открываю lab6-1.asm и ввожу текст программы (рис. 4.3).

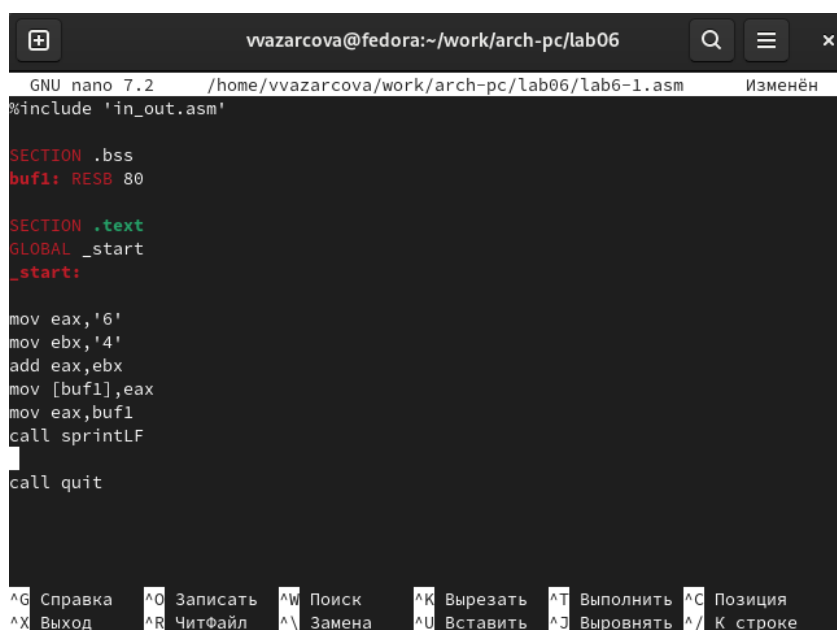


Рис. 4.3: Текст программы в lab6-1.asm

Создаю исполняемый файл, проверяю создание файла командой ls и запускаю его (рис. 4.4).

```

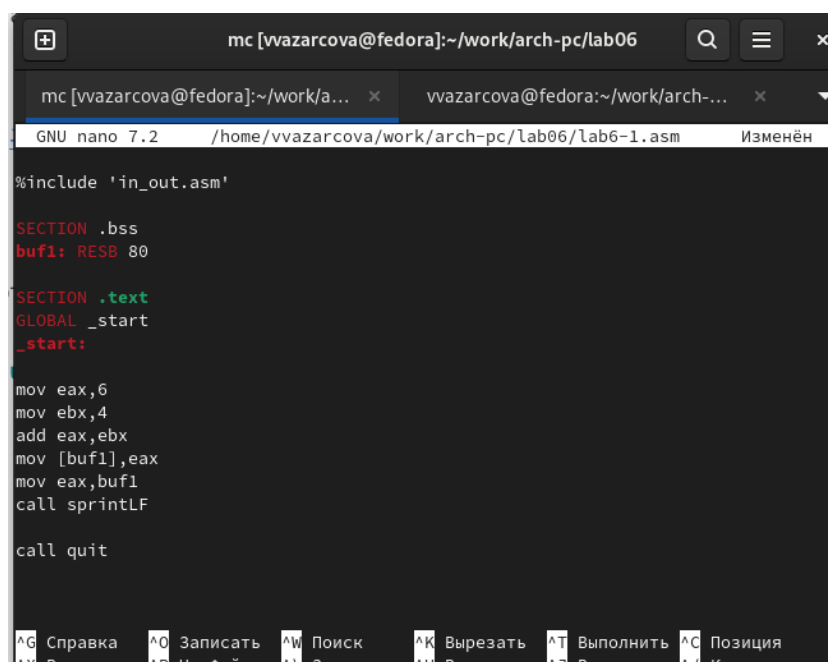
vvezorj
vvazarcova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
vvazarcova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.4: Запуск lab6-1

Программа выводит значение регистра `eax`, ожидается число 10, но результатом будет символ `j`, потому что код символа 6 равен 00110110, а код символа 4 - 00110100, и команда `add` записывает в регистр `eax` сумму кодов 01101010, что является кодом символа `j`.

3. Изменяю текст программы так, чтобы вместо символов записать в регистр цифры (рис. 4.5).



```

%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. 4.5: Измененный текст lab6-1.asm

Создаю исполняемый файл, проверяю создание файла командой `ls` и запускаю его (рис. 4.6).

```

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

```

Рис. 4.6: Запуск измененного lab6-1

Программа должна выводить символ на новой строке, выводятся две пустые строки. Согласно таблице ASCII, символ с кодом 10 это LF, т.е. символ перехода на новую строку, т.е. символ корректно отображается при выводе на экран (в виде новой, второй строки).

4. Создаю файл lab6-2.asm командой touch и проверяю его наличие командой ls (рис. 4.7).

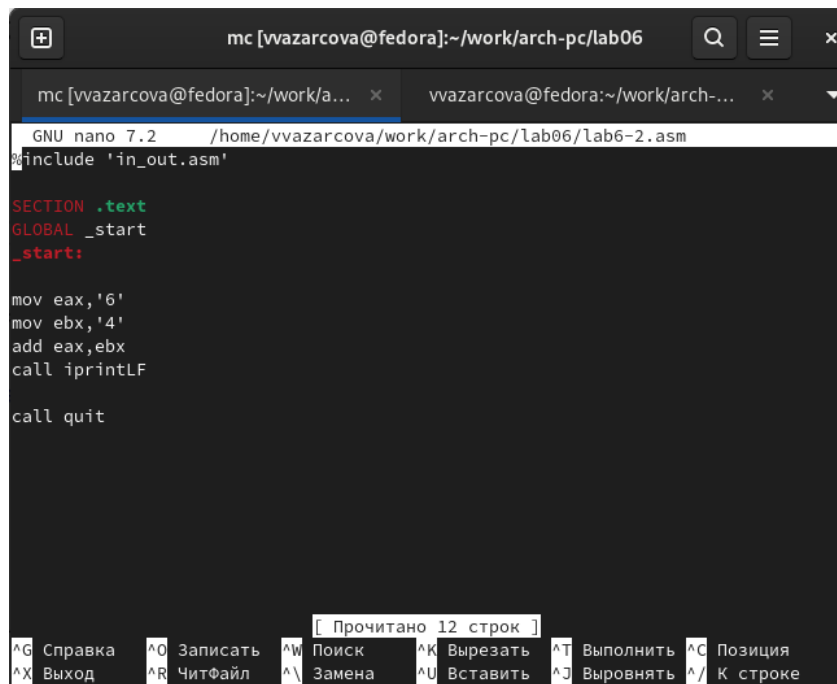
```

vvazarcova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
vvazarcova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.7: Создание lab6-2.asm

Открываю файл с помощью МС и ввожу текст программы (рис. 4.8).



```
mc [vvazarcova@fedora]:~/work/arch-pc/lab06
GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

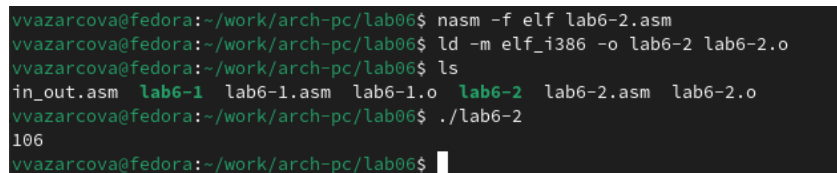
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF

call quit

[ Прочитано 12 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. 4.8: Текст программы в lab6-2.asm

Создаю исполняемый файл, проверяю создание файла командой ls и запускаю его (рис. 4.9).

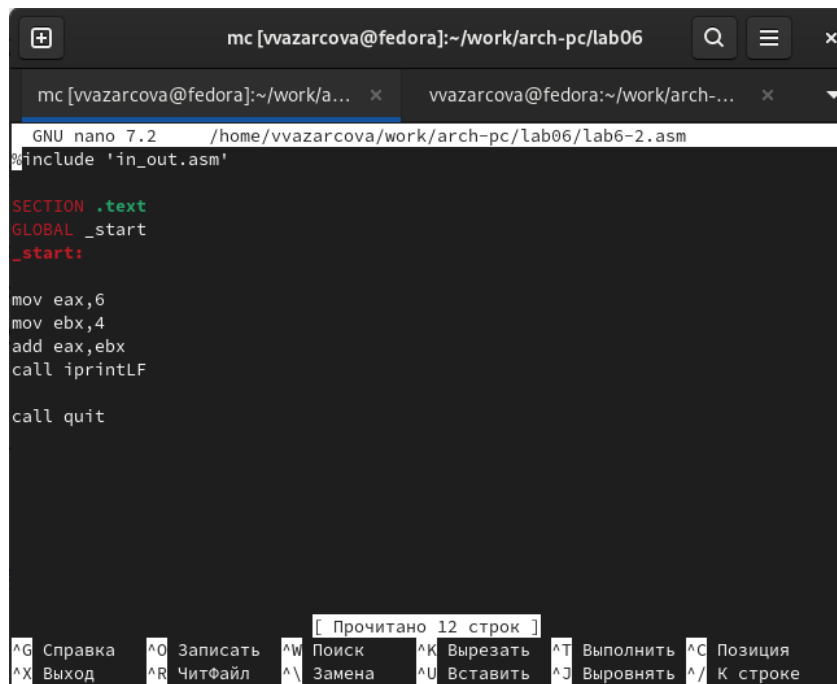


```
vvazarcova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
vvazarcova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.9: Запуск lab6-2

В результате работы программы выводится число 106. Аналогично первой программе, команда add складывает коды символов '6' и '4', но в отличии от первой программы функция iprintLF выводит число, а не символ, кодом которого является это число.

5. Аналогично номеру три, изменю символы на числа в тексте программы (рис. 4.10).



```
mc [vvazarcova@fedora]:~/work/arch-pc/lab06
GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

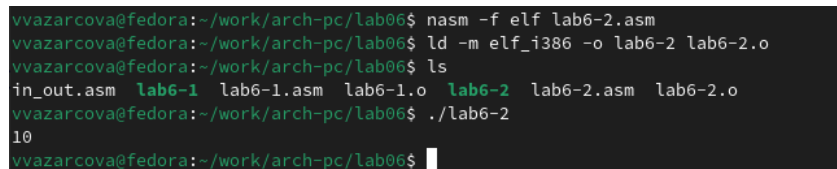
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF

call quit

[ Прочитано 12 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. 4.10: Измененный текст программы в lab6-2.asm

Создаю исполняемый файл, проверяю создание файла командой ls и запускаю его (рис. 4.11).



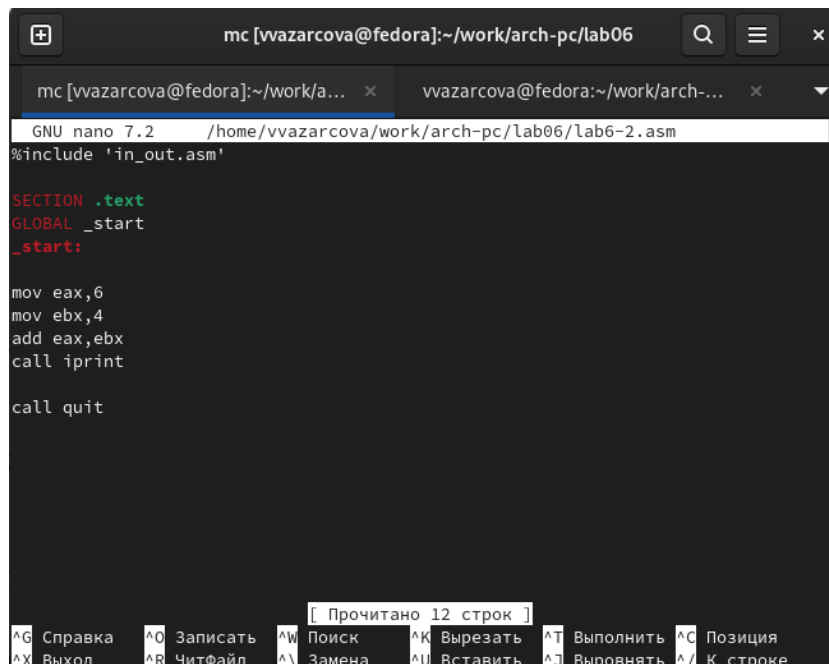
```
vvazarcova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
vvazarcova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.11: Запуск измененного lab6-2

Программа выводит число 10, т.к. теперь команда add складывает числа 4 и 6 и выводит результат в виде числа - 10.

Заменяю функцию iprintLF на iprint (рис. 4.12).





```
mc [vvazarcova@fedora]:~/work/arch-pc/lab06
GNU nano 7.2 /home/vvazarcova/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 iprint

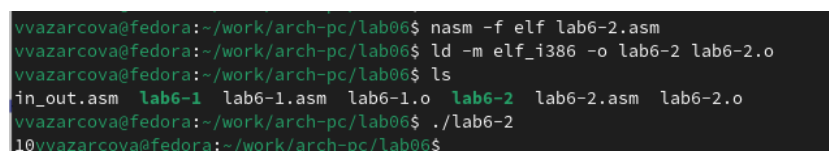
call quit
```

[ Прочитано 12 строк ]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция  
^X Выход ^R ЧитФайл ^M Замена ^U Вставить ^J Выровнять ^/\_ К строке

Рис. 4.12: Замена iprintLF на iprint

Создаю исполняемый файл, проверяю создание файла командой ls и запускаю его (рис. 4.13).



```
vvazarcova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ./lab6-2
10vvazarcova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.13: Запуск еще раз измененного lab6-2

Программа выводит 10, но не на новой строке. В этом состоит разница между iprintLF и iprint - первая подпрограмма выводит результат на новую строку, а вторая нет.

## 4.2 Выполнение арифметических операций в NASM

6. Создаю файл lab6-3.asm в каталоге лабораторной работы (рис. 4.14).

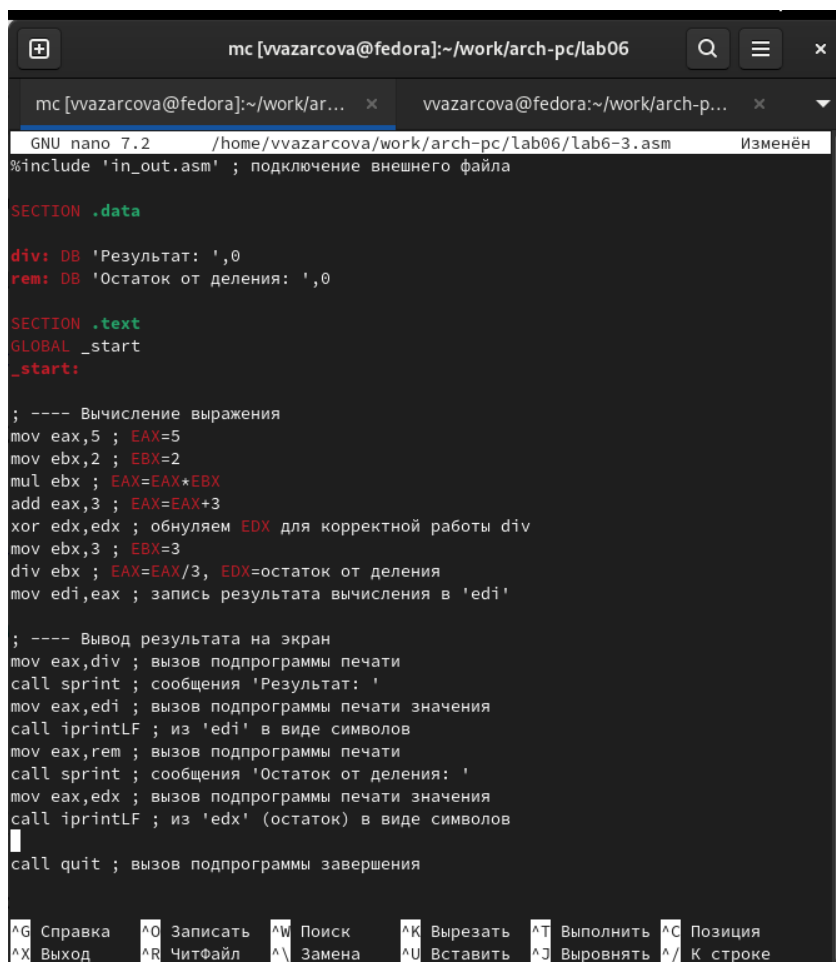
```

vvazarcova@fedora: ~/work/arch-pc/lab06$ ./lab6-2
10vvazarcova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
vvazarcova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.14: Создание lab6-3.asm

Ввожу в lab6-3.asm программу вычисления выражения  $f(x)=(5*2+3)/3$  (рис. 4.15).



```

mc [vvazarcova@fedora]:~/work/arch-pc/lab06
GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab06/lab6-3.asm Изменён
#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
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.15: Создание lab6-3.asm

Создаю исполняемый файл, проверяю создание файла командой ls и запускаю его (рис. 4.16).

```

vvazarcova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
vvazarcova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.16: Запуск lab6-3

Результат программы соответствует ожидаемому.

Изменяю текст программы для вычисления выражения  $f(x)=(4*6+2)/5$  (рис. 4.17).

```

GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab06/lab6-3.asm
#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
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax

; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintf ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintf ; из 'edx' (остаток) в виде символов

call quit ; вызов подпрограммы завершения

```

[ Прочитано 32 строки ]

<sup>^</sup>G Справка    <sup>^</sup>O Записать    <sup>^</sup>W Поиск    <sup>^</sup>K Вырезать    <sup>^</sup>T Выполнить    <sup>^</sup>C Позиция  
<sup>^</sup>X Выход    <sup>^</sup>R ЧитФайл    <sup>^</sup>\ Замена    <sup>^</sup>U Вставить    <sup>^</sup>J Выводить    <sup>^</sup>/ К строке

Рис. 4.17: Измененный текст программы в lab6-3.asm

Создаю исполняемый файл, проверяю создание файла командой ls и запускаю его (рис. 4.18).

```

vvazarcova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1.asm lab6-2 lab6-2.o lab6-3.asm
lab6-1 lab6-1.o lab6-2.asm lab6-3 lab6-3.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
vvazarcova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.18: Запуск измененного lab6-3

Программа выводит результат 5 и остаток 1, значит, она работает верно ( $4*6=24$ ;  $24+2=26$ ;  $26/5=5$ , ост. 1).

7. Создаю файл variant.asm и проверяю его наличие с помощью ls (рис. 4.19).

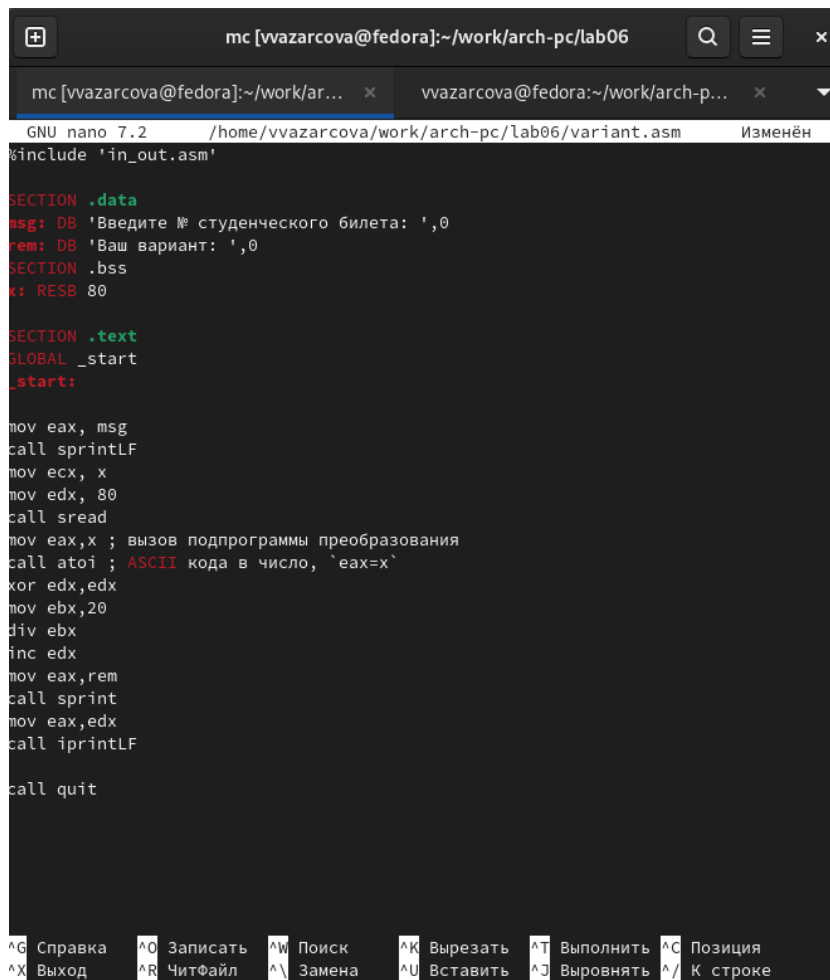
```

vvazarcova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1.asm lab6-2 lab6-2.o lab6-3.asm variant.asm
lab6-1 lab6-1.o lab6-2.asm lab6-3 lab6-3.o
vvazarcova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.19: Создание variant.asm

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



```
GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab06/variant.asm
#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:

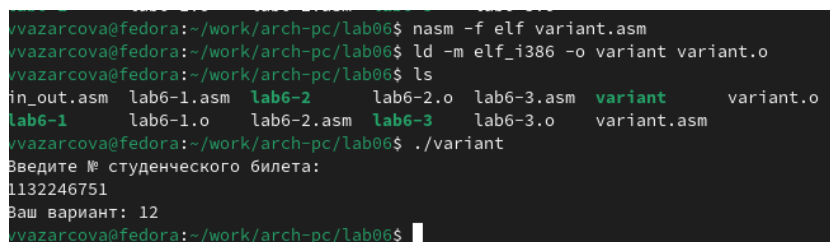
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintf

call quit

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

Рис. 4.20: Текст программы в variant.asm

Создаю исполняемый файл, проверяю создание файла командой ls и запускаю его (рис. 4.21).



```
vvazarcova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1.asm lab6-2 lab6-2.o lab6-3.asm variant variant.o
lab6-1 lab6-1.o lab6-2.asm lab6-3 lab6-3.o variant.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246751
Ваш вариант: 12
vvazarcova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.21: Запуск измененного lab6-3

Программа запрашивает ввод. Я ввожу свой номер студенческого билета (1132246751), и программа выводит, что мой вариант - 12.

Ответы на вопросы по выполнению лабораторной работы:

1. За вывод на экран сообщения 'Ваш вариант' отвечают следующие строки:

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

2. Следующие инструкции

```
mov ecx, x  
mov edx, 80  
call sread
```

используются для того, чтобы загрузить адрес переменной x в регистр ecx.

3. Инструкция "call atoi" вызывает подпрограмму из файла in\_out.asm, которая переводит строку ASCII в целое число.

4. За вычисление варианта отвечают следующие строки:

```
xor edx, edx  
mov ebx, 20  
div ebx
```

Эти команды ищут остаток от деления номера студенческого билета на 20.

5. Остаток от деления при выполнении инструкции "div ebx" записывается в регистр edx.

6. Инструкция inc edx увеличивает значение в регистре edx на 1, для того, чтобы список возможных вариантов задания, вычисленных программой, начинался не с 0 а с 1.

7. За вывод на экран результата вычислений отвечают строки:

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

## 5 Задания для самостоятельной работы

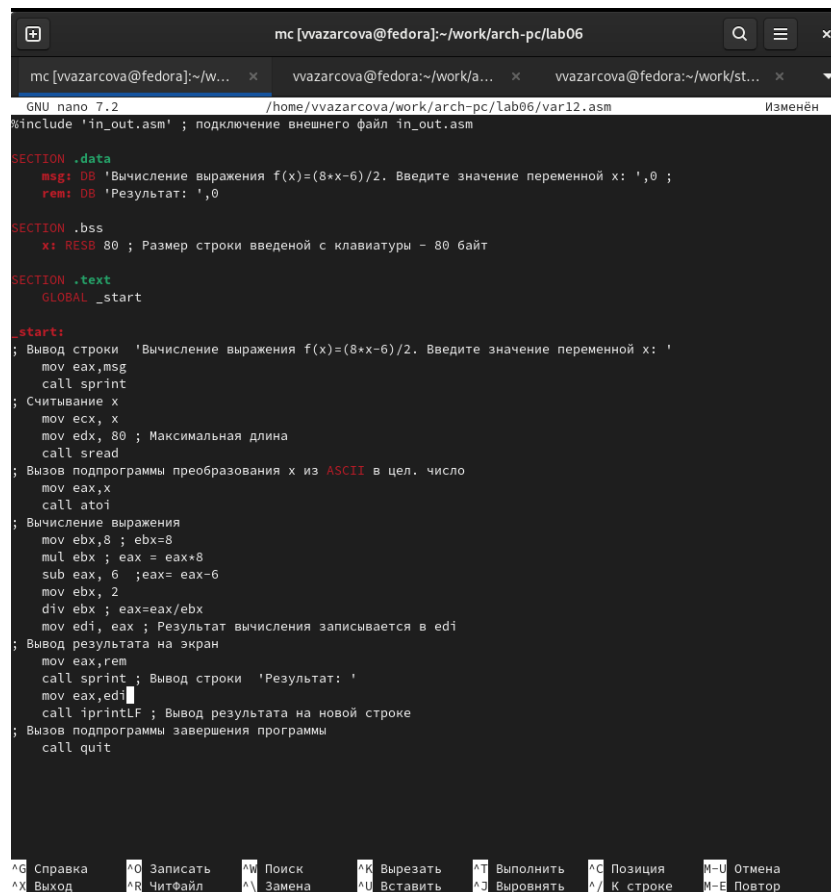
1. Т.к. мой вариант задания - 12, мне нужно написать программу для вычисления выражения  $f(x)=(8*x-6)/2$ , которая будет выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$  и выводить результат вычислений.

Создаю файл `var12.asm` в каталоге для выполнения лабораторной работы с помощью команды `touch` и проверяю его наличие командой `ls` (рис. 5.1).

```
vvazarcova@fedora:~/work/arch-pc/lab06$ touch var12.asm
vvazarcova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  var12.asm  variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o    variant    variant.o
```

Рис. 5.1: Создание `var12.asm`

2. Открываю `var12.asm` с помощью МС и пишу код программы с пояснениями, аналогично программам, приведенным в пример в ходе лабораторной работы (рис. 5.2).



```
mc [vazarcova@fedora]:~/work/arch-pc/lab06
GNU nano 7.2 /home/vazarcova/work/arch-pc/lab06/var12.asm
#include 'in_out.asm' ; подключение внешнего файла in_out.asm

SECTION .data
msg: DB 'Вычисление выражения f(x)=(8*x-6)/2. Введите значение переменной x: ',0 ;
rem: DB 'Результат: ',0

SECTION .bss
x: RESB 80 ; Размер строки введенной с клавиатуры - 80 байт

SECTION .text
GLOBAL _start

_start:
; Вывод строки 'Вычисление выражения f(x)=(8*x-6)/2. Введите значение переменной x: '
mov eax,msg
call sprint
; Считывание x
mov ecx, x
mov edx, 80 ; Максимальная длина
call sread
; Вызов подпрограммы преобразования x из ASCII в цел. число
mov eax,x
call atoi
; Вычисление выражения
mov ebx,8 ; ebx=8
mul ebx ; eax = eax*8
sub eax, 6 ; eax= eax-6
mov ebx, 2
div ebx ; eax=eax/ebx
mov edi, eax ; Результат вычисления записывается в edi
; Вывод результата на экран
mov eax,rem
call sprint ; Вывод строки 'Результат: '
mov eax,edi
call iprintLF ; Вывод результата на новой строке
; Вызов подпрограммы завершения программы
call quit
```

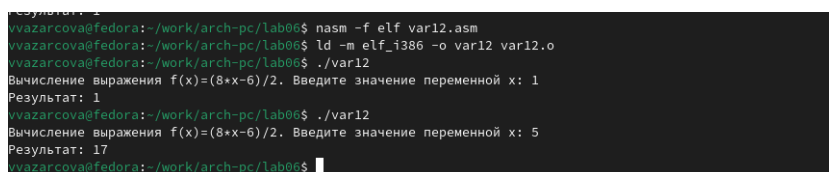
Рис. 5.2: Текст программы в var12.asm

- Сначала в программе я задаю переменные с текстом, который нужно будет вывести в её ходе, затем выделяю память на неинициализированные данные, далее прописываю команду `_start` для начала работы программы.
- В тексте программы я вызываю подпрограмму `sprint` вывода строки и вывожу текст вычисляемого выражения и просьбу ввести `x`, задаю максимальную длину `x`, вызываю подпрограмму ввода с клавиатуры `sread` и записываю `x` в `ecx`. Затем, вызываю подпрограмму `atoi` для перевода `x` из строчки ASCII в целое число - целое число `x` записывается в `eax`.
- Далее, с помощью подпрограмм `mul`, `sub` и `div` (умножения, вычитания, деления), провожу нужные арифметические действия, при этом записывая значения с которыми работаю в `ebx`. В конце записываю значение результа-



та в edi.

- Последними я вызываю подпрограммы `sprint` и `iprintLF`, чтобы вывести строку, объявляющую результат, и затем вывести результат выражения на новой строке, переместив выводимое значение в еах перед каждой из подпрограмм.
3. Создаю исполняемый файл, проверяю создание файла командой `ls` и запускаю его (рис. 5.3).



```
vazarcova@fedora: ~/work/arch-pc/lab06$ nasm -f elf var12.asm
vazarcova@fedora: ~/work/arch-pc/lab06$ ld -m elf_i386 -o var12 var12.o
vazarcova@fedora: ~/work/arch-pc/lab06$ ./var12
Вычисление выражения f(x)=(8*x-6)/2. Введите значение переменной x: 1
Результат: 1
vazarcova@fedora: ~/work/arch-pc/lab06$ ./var12
Вычисление выражения f(x)=(8*x-6)/2. Введите значение переменной x: 5
Результат: 17
vazarcova@fedora: ~/work/arch-pc/lab06$
```

Рис. 5.3: Запуск `var12.asm`

В итоге, программа выводит результат 1 при вводе числа 1 и результат 17 при вводе числа 5; Это верно, поскольку:

- $(8*1-6)/2=2/2=1$
- $(8*5-6)/2=(40-6)/2=34/2=17$

Значит, программа написана корректно.

## 6 Выводы

Подводя итоги данной лабораторной работы, я закрепила знания работы подфункций из внешнего файла `in_out.asm`, научилась пользоваться арифметическими функциями и успешно написала и запустила несколько программ, производящих арифметические действия в NASM.

## **Список литературы**