

# **Отчёт по лабораторной работе №7**

**Дисциплина: Архитектура компьютера**

Ганина Таисия, НКАбд-01-22

# 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>8</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>11</b>
<b>5</b>	<b>Листинги программ:</b>	<b>25</b>
<b>6</b>	<b>Выводы</b>	<b>31</b>
	<b>Список литературы</b>	<b>32</b>

# Список иллюстраций

4.1	Рис. 1	11
4.2	Рис. 2	12
4.3	Рис. 3	12
4.4	Рис. 4	12
4.5	Рис. 5	13
4.6	Рис. 6	13
4.7	Рис. 7	13
4.8	Рис. 8	14
4.9	Рис. 9	14
4.10	Рис. 10	15
4.11	Рис. 11	16
4.12	Рис. 12	16
4.13	Рис. 13	17
4.14	Рис. 14	17
4.15	Рис. 15	18
4.16	Рис. 16	18
4.17	Рис. 17	19
4.18	Рис. 18	19
4.19	Рис. 19	19
4.20	Рис. 20	20
4.21	Рис. 21	21
4.22	Рис. 22	23
4.23	Рис. 23	24

## **Список таблиц**

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Создать файл lab7-1.asm и ввести в него программу из листинга 1, создать исполняемый файл и запустить его.
2. Исправить листинг 1, заменив строки

```
mov eax, '6'
```

```
mov ebx, '4'
```

на строки

```
mov eax, 6
```

```
mov ebx, 4
```

Создать исполняемый файл и запустить его, пользуясь таблицей ASCII определить какому символу соответствует код 10.

3. Создать файл lab7-2.asm, ввести в него программу из листинга 2, создать исполняемый файл и запустить его.
4. Исправить листинг 2, заменив строки

```
mov eax, '6'
```

```
mov ebx, '4'
```

на строки

```
mov eax, 6
```

```
mov ebx, 4
```

Создать исполняемый файл и запустить его.

5. Заменить функцию `iprintln` на `iprint`. Создать исполняемый файл и запустить его. Выяснить чем отличается вывод функций `iprintln` и `iprint`.

6. Создать файл lab7-3.asm, заполнить его соответственно с листингом 3, создать исполняемый файл и запустить его.
7. Изменить файл так, чтобы программа вычисляла выражение  $f(x) = (4 * 6 + 2)/5$
8. Создать файл “вариант”, заполнить его соответственно с листингом 4, создать исполняемый файл и запустить его.
9. Ответить на вопросы по разделу.
10. Написать программу для вычисления выражения  $5 * (x + 18) - 28$  и проверить его при  $x=2$  и при  $x=3$ .

### 3 Теоретическое введение

Схема команды целочисленного сложения `add` (от англ. *addition* - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда.

1. Команда `add` работает как с числами со знаком, так и без знака и выглядит следующим образом:

```
add <операнд_1>, <операнд_2>
```

Допустимые сочетания операндов для команды `add` аналогичны сочетаниям операндов для команды `mov`. Так, например, команда `add eax,ebx` прибавит значение из регистра `eax` к значению из регистра `ebx` и запишет результат в регистр `eax`.

2. Команда целочисленного вычитания `sub` (от англ. *subtraction* – вычитание) работает аналогично команде `add` и выглядит следующим образом:

```
sub <операнд_1>, <операнд_2>
```

Так, например, команда `sub ebx,5` уменьшает значение регистра `ebx` на 5 и записывает результат в регистр `ebx`.

3. Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, 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. Эти команды содержат один операнд и имеет следующий вид:



`inc <операнд>`

`dec <операнд>`

Операндом может быть регистр или ячейка памяти любого размера. Команды инкремента и декремента выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. Так, например, команда `inc ebx` увеличивает значение регистра `ebx` на 1, а команда `inc ax` уменьшает значение регистра `ax` на 1.

4. Еще одна команда, которую можно отнести к арифметическим командам это команда изменения знака `neg`:

`neg <операнд>`

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. Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды.

Для беззнакового умножения используется команда `mul` (от англ. *multiply* – умножение):

`mul <операнд>`

Для знакового умножения используется команда `imul`:

`imul <операнд>`

Для команд умножения один из сомножителей указывается в команде и должен находиться в регистре или в памяти, но не может быть непосредственным операндом. Вторым сомножителем 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.

6. Для деления, как и для умножения, существует 2 команды `div` (от англ. *divide* – деление) и `idiv`:

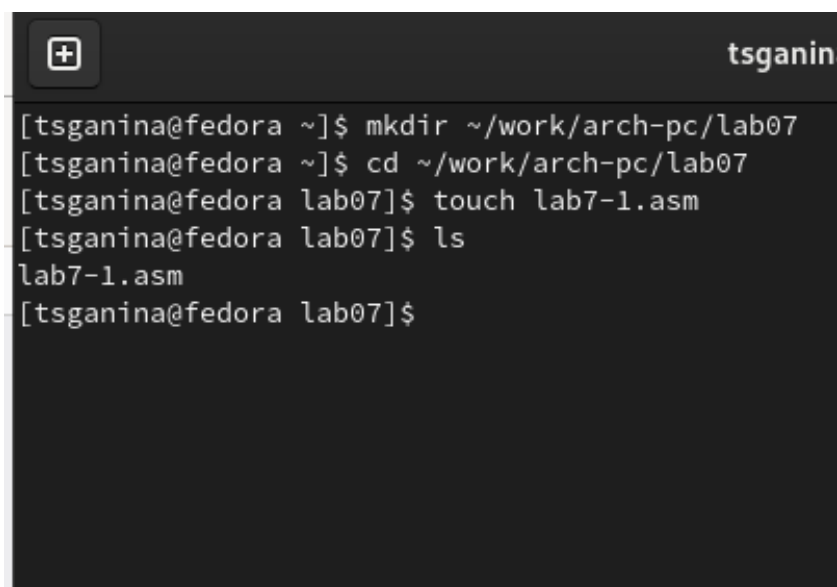
`div <делитель>`

`idiv <делитель>`

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

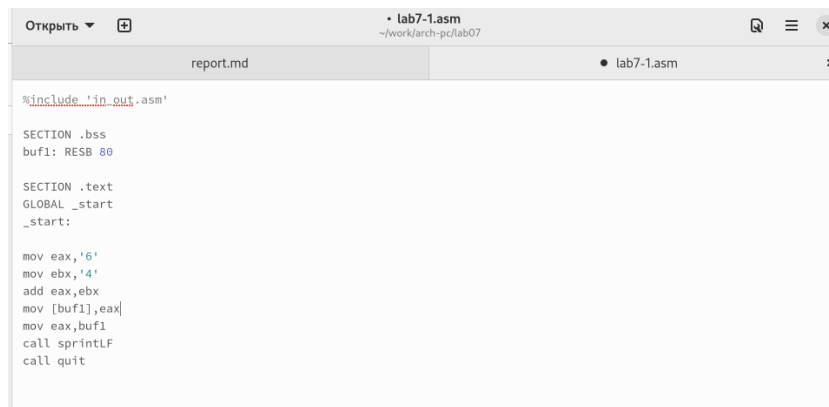
## 4 Выполнение лабораторной работы

1. Я создала файл lab7-1.asm и ввела в него программу из листинга 1, создала исполняемый файл и запустила его (рис. 4.1, 4.2, 4.3).

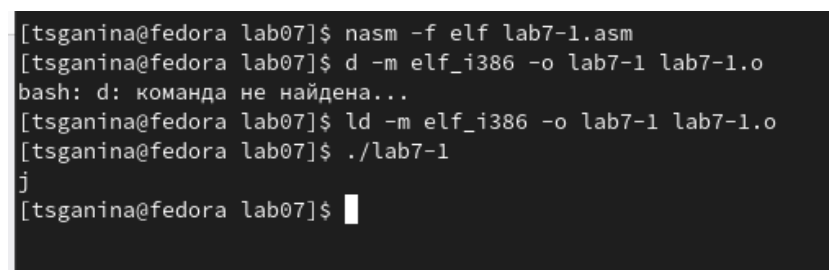
A terminal window with a dark background. The title bar shows a window icon on the left and the name 'tsganin' on the right. The terminal contains the following text:

```
[tsganina@fedora ~]$ mkdir ~/work/arch-pc/lab07  
[tsganina@fedora ~]$ cd ~/work/arch-pc/lab07  
[tsganina@fedora lab07]$ touch lab7-1.asm  
[tsganina@fedora lab07]$ ls  
lab7-1.asm  
[tsganina@fedora lab07]$
```

4.1: Рис. 1



4.2: Рис. 2



4.3: Рис. 3

## 2. Исправила листинг 1, заменив строки

```
mov eax, '6'
```

```
mov ebx, '4'
```

на строки

```
mov eax, 6
```

```
mov ebx, 4
```

Создала исполняемый файл и запустила его, пользуясь таблицей ASCII определила какому символу соответствует код 10, (рис. 4.4, 4.5, 4.6).



4.4: Рис. 4

```
mov eax,6
mov ebx,4
add eax,ebx
```

4.5: Рис. 5

```
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[tsganina@fedora lab07]$ ./lab7-1
j
[tsganina@fedora lab07]$ nasm -f elf lab7-1.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[tsganina@fedora lab07]$ ./lab7-1

[tsganina@fedora lab07]$
```

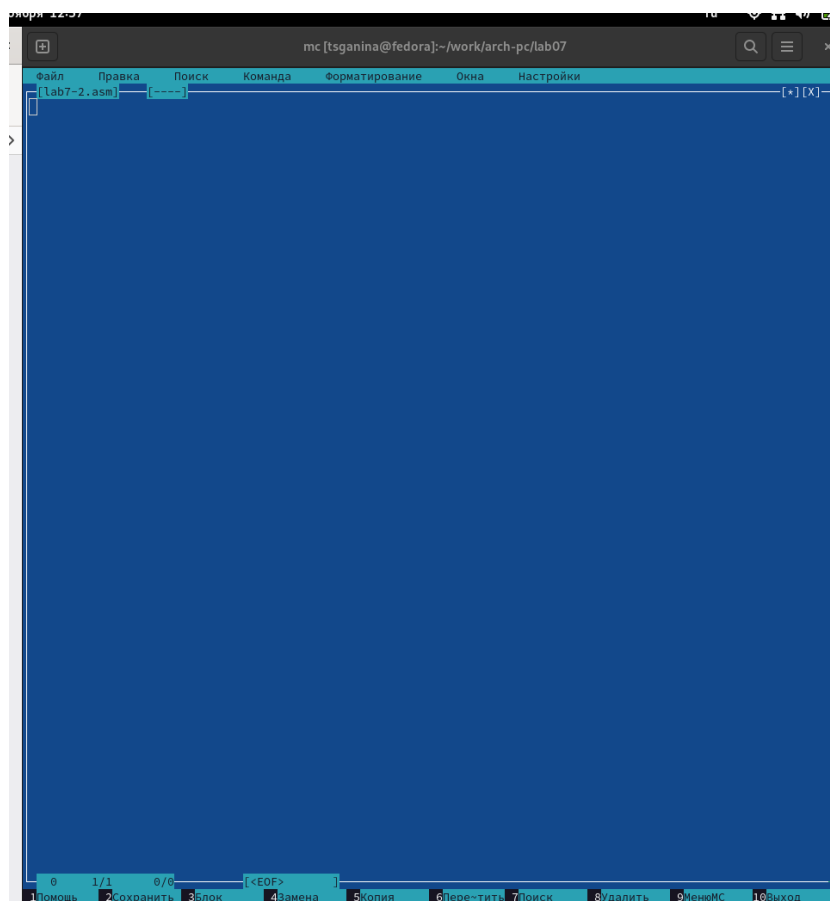
4.6: Рис. 6

Вывод: код 10 соответствует символу переноса строки, но на экране этот символ не отображается.

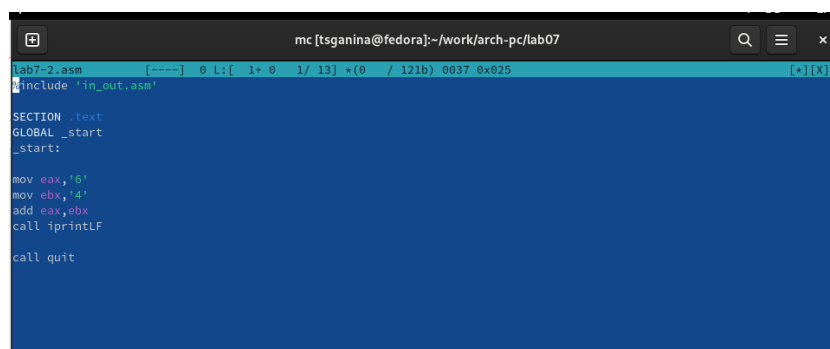
3. Создала файл lab7-2.asm, ввела в него программу из листинга 2, создала исполняемый файл и запустила его (рис. 4.7, 4.8, 4.9, 4.10).

```
[tsganina@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
[tsganina@fedora lab07]$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
[tsganina@fedora lab07]$
```

4.7: Рис. 7



4.8: Рис. 8



4.9: Рис. 9

```
[tsganina@fedora lab07]$ nasm -f elf lab7-2.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[tsganina@fedora lab07]$ ./lab7-2
106
[tsganina@fedora lab07]$
```

4.10: Рис. 10

#### 4. Исправила листинг 2, заменив строки

```
mov eax, '6'
```

```
mov ebx, '4'
```

на строки

```
mov eax, 6
```

```
mov ebx, 4
```

Создала исполняемый файл и запустила его (рис. 4.11, 4.12).

```
lab7-2.asm [-M--]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit
```

4.11: Рис. 11

```
[tsganina@fedora lab07]$ nasm -f elf lab7-2.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[tsganina@fedora lab07]$ ./lab7-2
106
[tsganina@fedora lab07]$ mc

[tsganina@fedora lab07]$ nasm -f elf lab7-2.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[tsganina@fedora lab07]$ ./lab7-2
10
[tsganina@fedora lab07]$
```

4.12: Рис. 12



5. Заменяла функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его. Выяснила чем отличается вывод функций `iprintLF` и `iprint` (рис. 4.13).

```
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[tsganina@fedora lab07]$ ./lab7-2
10
[tsganina@fedora lab07]$ mc

[tsganina@fedora lab07]$ nasm -f elf lab7-2.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[tsganina@fedora lab07]$ ./lab7-2
[tsganina@fedora lab07]$ ./lab7-2
10[tsganina@fedora lab07]$
```

4.13: Рис. 13

Вывод: отличие состоит в том, что `iprint` не совершает перенос строки.

6. Создала файл `lab7-3.asm`, заполнила его соответственно с листингом 3, создала исполняемый файл и запустила его (рис. 4.14, 4.15, 4.16).

```
[tsganina@fedora lab07]$ ./lab7-2
10[tsganina@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-3.asm
[tsganina@fedora lab07]$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.o lab7-3.asm
[tsganina@fedora lab07]$
```

4.14: Рис. 14

```

; 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
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

```

4.15: Рис. 15

```

[tsganina@fedora lab07]$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.o lab7-3.asm
[tsganina@fedora lab07]$ mc

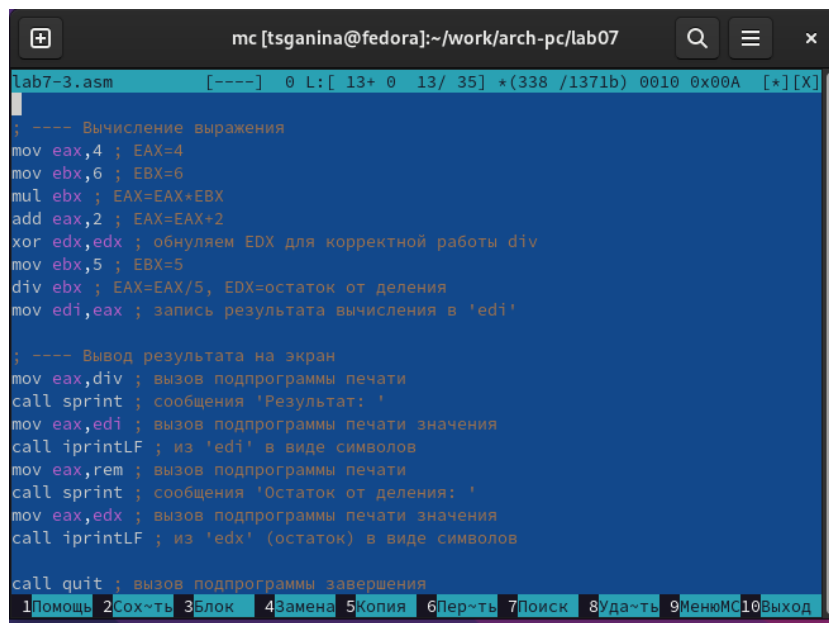
[tsganina@fedora lab07]$ nasm -f elf lab7-3.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[tsganina@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[tsganina@fedora lab07]$

```

4.16: Рис. 16

7. Изменила файл так, чтобы программа вычисляла выражение  $f(x) = (4 *$

$6 + 2)/5$  (рис. 4.17, 4.18).

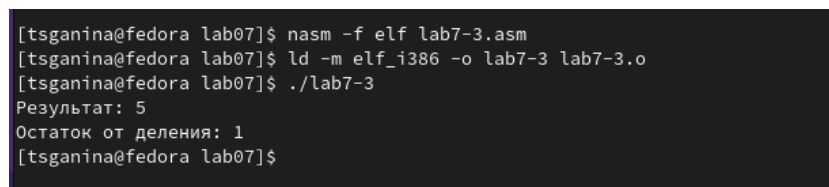


```
mc [tsganina@fedora]:~/work/arch-pc/lab07
lab7-3.asm  [----]  0 L: [ 13+ 0  13/ 35] *(338 /1371b) 0010 0x00A  [*][X]
; ---- Вычисление выражения
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран
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
1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

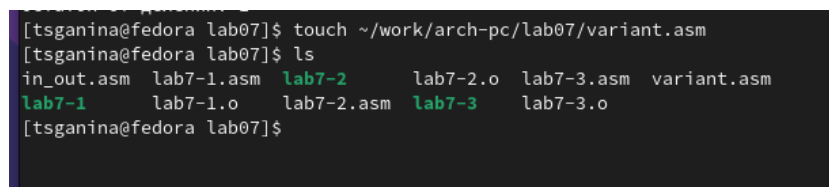
4.17: Рис. 17



```
[tsganina@fedora lab07]$ nasm -f elf lab7-3.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[tsganina@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[tsganina@fedora lab07]$
```

4.18: Рис. 18

8. Создала файл “вариант”, заполнила его соответственно с листингом 4, создала исполняемый файл и запустила его (рис. 4.19, 4.20, 4.21).



```
[tsganina@fedora lab07]$ touch ~/work/arch-pc/lab07/variant.asm
[tsganina@fedora lab07]$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.o  lab7-3.asm  variant.asm
lab7-1      lab7-1.o   lab7-2.asm  lab7-3    lab7-3.o
[tsganina@fedora lab07]$
```

4.19: Рис. 19

```

variant.asm      [----]  0 L:[  1+ 0  1/ 37] *(0
;-----
; Программа вычисления варианта
;-----
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите No студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintLF

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 sprint
mov eax,edx
call iprintLF

call quit

```

4.20: Рис. 20

```

[tsganina@fedora lab07]$ nasm -f elf variant.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[tsganina@fedora lab07]$ ./variant
Введите No студенческого билета:
1132226429
Ваш вариант: 10
[tsganina@fedora lab07]$

```

4.21: Рис. 21

Выполнив те же вычисления вручную, выяснила, что ответ, данный программой, верен.

#### 9. Отвечаю на вопросы по разделу:

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

```
`mov eax,rem`
```

```
`call sprint`
```

2. Для чего используются следующие инструкции?

```
`mov ecx, x` - адрес вводимой строки `x` записывается в регистр `ecx`.
```

```
`mov edx, 80` - 80 - длина вводимой строки, записана в `edx`.
```

```
`call sread` - считывание ввода с клавиатуры.
```

3. Для чего используется инструкция "call atoi"?

Эта инструкция вызывает программу из файла "in\_out.asm" и преобразует ascii-код символа в целое число и записывает результат в регистр eax.

4. Какие строки листинга 7.4 отвечают за вычисления варианта?

```
xor edx,edx
```

```
mov ebx,20
```

```
div ebx
```

```
inc edx
```

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

В регистр `edx`

6. Для чего используется инструкция “inc edx”?

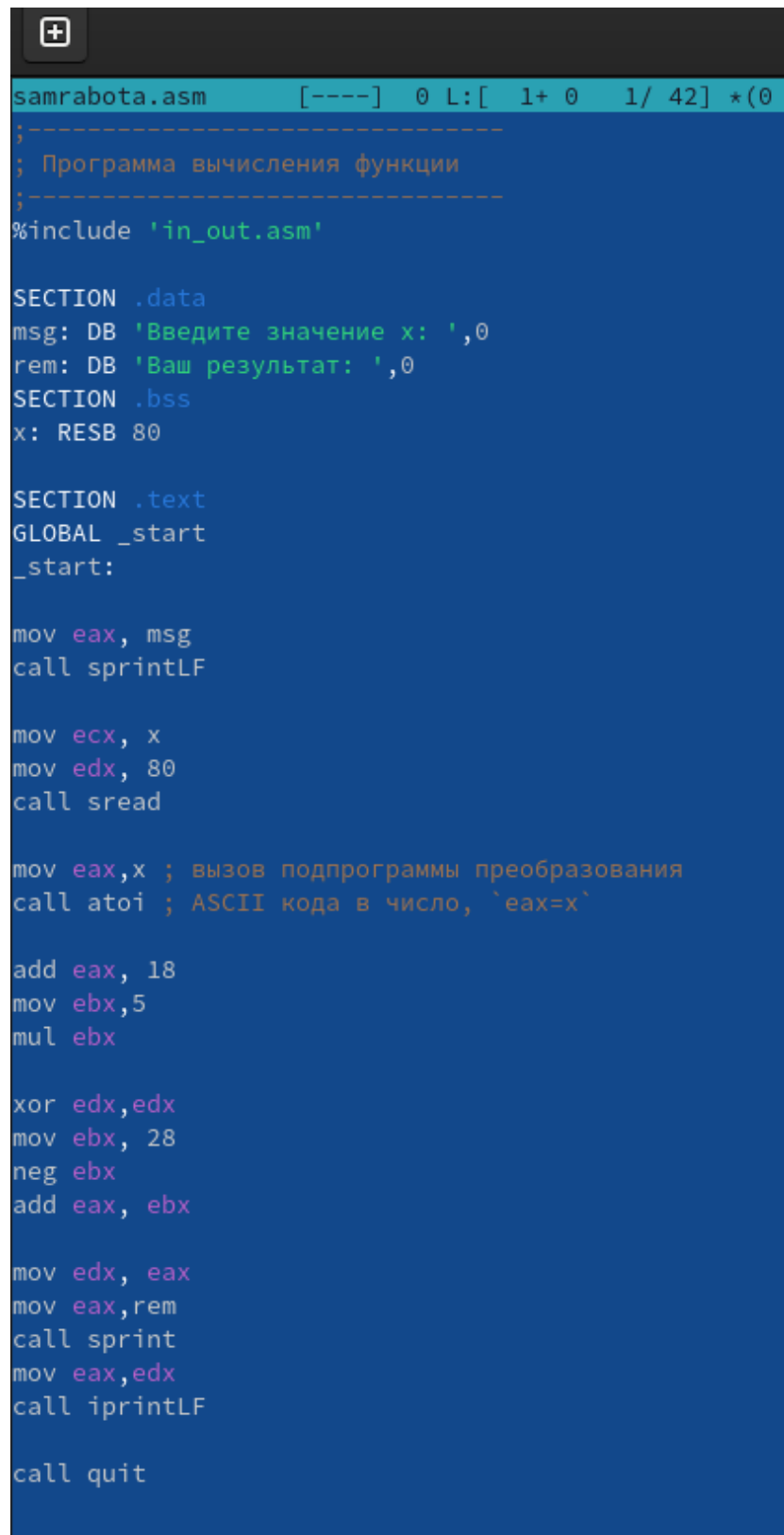
Для того, чтобы прибавить к значению `edx` единицу

7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?

```
mov eax,edx
```

```
call iprintLF
```

10. Написать программу для вычисления выражения  $5 * (x + 18) - 28$  и проверить его при  $x=2$  и при  $x=3$  (рис. 4.22, 4.23).



```
samrabota.asm [----] 0 L: [ 1+ 0 1/ 42] *(0
;
; Программа вычисления функции
;
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите значение x: ',0
rem: DB 'Ваш результат: ',0
SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintLF

mov ecx, x
mov edx, 80
call sread

mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'

add eax, 18
mov ebx, 5
mul ebx

xor edx, edx
mov ebx, 28
neg ebx
add eax, ebx

mov edx, eax
mov eax, rem
call sprint
mov eax, edx
call iprintLF

call quit
```

4.22: Рис. 22

```
[tsganina@fedora lab07]$ nasm -f elf samrabota.asm
[tsganina@fedora lab07]$ ld -m elf_i386 -o samrabota samrabota.o
[tsganina@fedora lab07]$ ./samrabota
Введите значение x:
2
Ваш результат: 72
[tsganina@fedora lab07]$ ./samrabota
Введите значение x:
3
Ваш результат: 77
[tsganina@fedora lab07]$
```

4.23: Рис. 23

Проверила себя, выполнив вычисления вручную - ответ получен верный.



## 5 Листинги программ:

### 1. lab7-1.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 sprintLF
```

```
call quit
```

### 2. lab7-2.asm

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

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax,6
```

```
mov ebx,4
```

```
add eax,ebx
```

```
call iprint
```

```
call quit
```

### 3. lab7-3.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
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

```

#### 4. variant.asm

```

;-----
; Программа вычисления варианта
;-----
%include 'in_out.asm'

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

```

```

x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintLF

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 sprint
mov eax,edx
call iprintLF

call quit

```

## 5. samrabota.asm - самостоятельная работа

```

;-----
; Программа вычисления функции
;-----
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите значение x: ',0
rem: DB 'Ваш результат: ',0
SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintLF

mov ecx, x
mov edx, 80
call sread

mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`

add eax, 18
mov ebx,5
mul ebx

```

```
xor edx,edx
mov ebx, 28
neg ebx
add eax, ebx

mov edx, eax
mov eax,rem
call sprint
mov eax,edx
call iprintLF

call quit
```

## 6 Выводы

В ходе этой лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

# Список литературы

1. Текстовый документ “Лабораторная работа №7. Арифметические операции в NASM.”