

# **Отчет по лабораторной работе №7**

**Арифметические операции в NASM**

*Акопян Сатеник*

# Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение работы	7
4	Задание для самостоятельной работы	14
5	Выводы	16
	Список литературы	17

## Список иллюстраций

3.1	рисунок 1 . . . . .	7
3.2	рисунок 2 . . . . .	7
3.3	рисунок 3 . . . . .	8
3.4	рисунок 4 . . . . .	8
3.5	рисунок 5 . . . . .	8
3.6	рисунок 6 . . . . .	9
3.7	рисунок 7 . . . . .	9
3.8	рисунок 8 . . . . .	9
3.9	рисунок 9 . . . . .	10
3.10	рисунок 10 . . . . .	10
3.11	рисунок 11 . . . . .	10
3.12	рисунок 12 . . . . .	11
3.13	рисунок 13 . . . . .	11
3.14	рисунок 14 . . . . .	12
3.15	рисунок 15 . . . . .	12
3.16	рисунок 16 . . . . .	12
3.17	рисунок 17 . . . . .	13
3.18	рисунок 18 . . . . .	13
4.1	рисунок 19 . . . . .	14
4.2	рисунок 20 . . . . .	15
4.3	рисунок 21 . . . . .	15

## Список таблиц

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

## 2 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. # Выполнение лабораторной работы

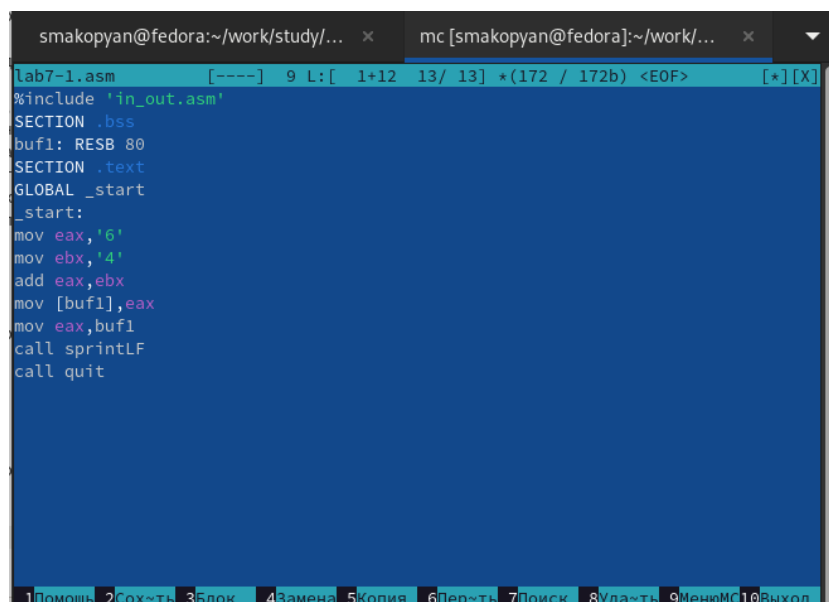
### 3 Выполнение работы

1. Создаем каталог для программ лабораторной работы № 7, переходим в него и создаем файл lab7-1.asm (рис. 3.1)

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

Рис. 3.1: рисунок 1

2. Вводим в файл lab7-1.asm текст программы из листинга, представленного в лабораторной работе (рис. 3.2)



```
lab7-1.asm  [----]  9  L: [ 1+12 13/ 13] *(172 / 172b) <EOF>  [*] [X]  
%include 'in_out.asm'  
SECTION .bss  
buf1: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, '6'  
mov ebx, '4'  
add eax, ebx  
mov [buf1], eax  
mov eax, buf1  
call printf  
call quit
```

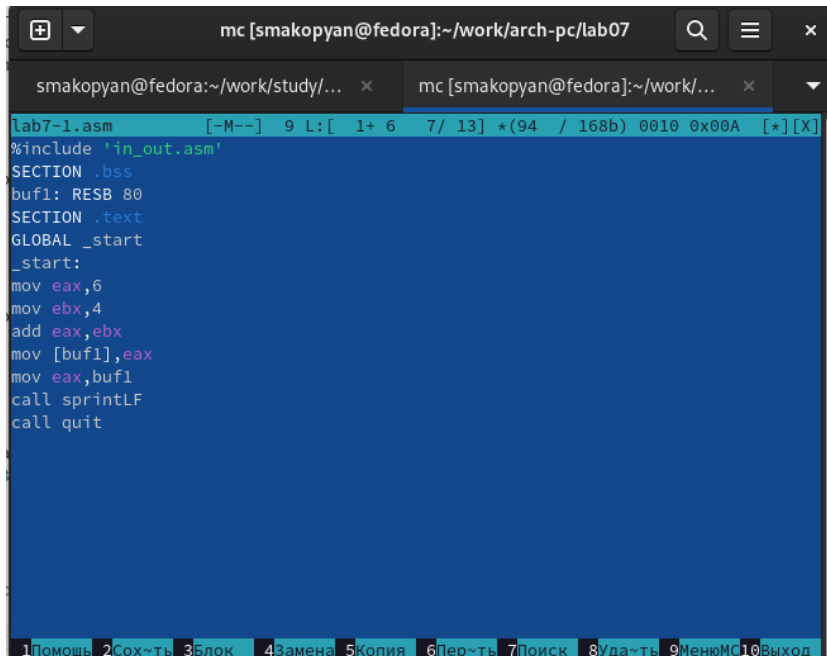
Рис. 3.2: рисунок 2

3. Создаем исполняемый файл и запускаем его, предварительно скопировав подключаемый файл `in_out.asm` в каталог с текстом программы (рис. 3.3)

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

Рис. 3.3: рисунок 3

4. Изменим текст программы и вместо символов запишем в регистры числа (рис. 3.4)



```
lab7-1.asm [-M--] 9 L: [ 1+ 6 7/ 13] *(94 / 168b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf
call quit
```

Рис. 3.4: рисунок 4

5. Создадим исполняемый файл и запустим его (рис. 3.5).

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

Рис. 3.5: рисунок 5



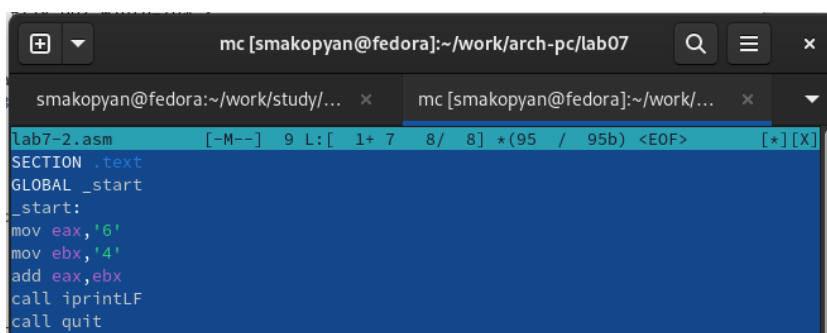
Пользуясь таблицей ASCII, определяем что код 10 соответствует символу перевода на новую строку, что является причиной того, почему данный символ вывел на экран только 2 пустые строки.

6. Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. 3.6)

```
[smakopyan@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm  
[smakopyan@fedora lab07]$
```

Рис. 3.6: рисунок 6

7. Вводим в созданный файл текст программы из листинга 7.2, представленного в лабораторной работе (рис. 3.7)



```
lab7-2.asm [-M--] 9 L: [ 1+ 7 8/ 8] *(95 / 95b) <EOF> [*][X]  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, '6'  
mov ebx, '4'  
add eax, ebx  
call iprintLF  
call quit
```

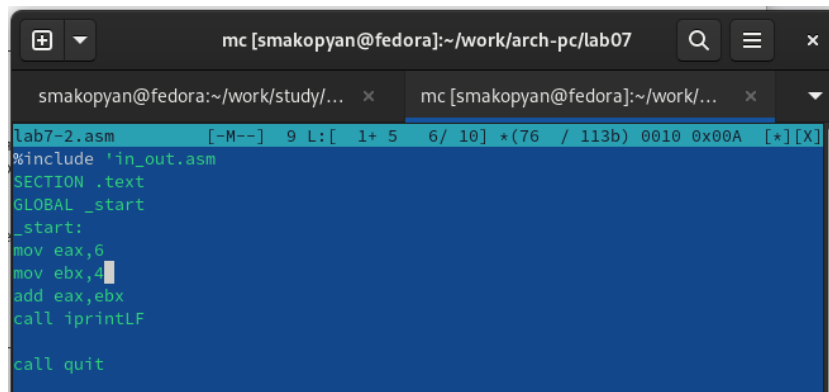
Рис. 3.7: рисунок 7

8. Создаем исполняемый файл и запускаем его (рис. 3.8)

```
[smakopyan@fedora lab07]$ nasm -f elf lab7-2.asm  
lab7-2.asm:1: warning: unterminated string [-w+other]  
[smakopyan@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o  
[smakopyan@fedora lab07]$ ./lab7-2  
106  
[smakopyan@fedora lab07]$
```

Рис. 3.8: рисунок 8

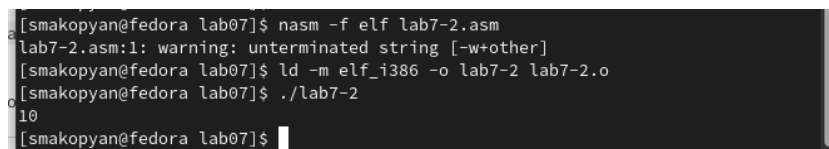
9. Аналогично предыдущему примеру изменим символы на числа. (рис. 3.9)



```
lab7-2.asm      [-M--]  9 L: [ 1+ 5 6/ 10] *(76 / 113b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 3.9: рисунок 9

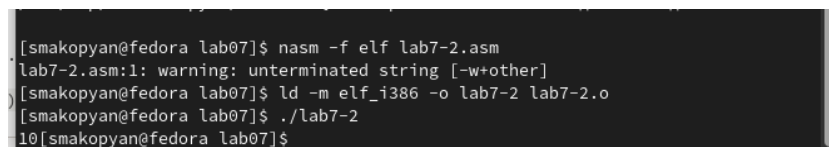
10. Создаем исполняемый файл и запускаем его (рис. 3.10)



```
[smakopyan@fedora lab07]$ nasm -f elf lab7-2.asm
lab7-2.asm:1: warning: unterminated string [-w+other]
[smakopyan@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[smakopyan@fedora lab07]$ ./lab7-2
10
[smakopyan@fedora lab07]$
```

Рис. 3.10: рисунок 10

11. Меняем функцию iprintLF на iprint. Создаем исполняемый файл и запускаем его. (рис. 3.11)

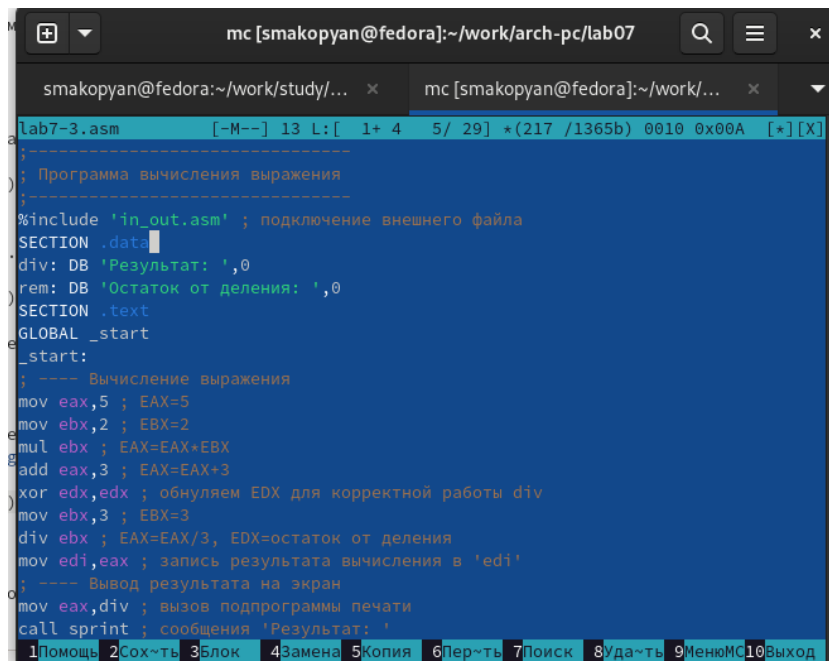


```
[smakopyan@fedora lab07]$ nasm -f elf lab7-2.asm
lab7-2.asm:1: warning: unterminated string [-w+other]
[smakopyan@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[smakopyan@fedora lab07]$ ./lab7-2
10[smakopyan@fedora lab07]$
```

Рис. 3.11: рисунок 11

Нетрудно заметить, что после проведенных действий изменилось только то, что следующая строка после вывода программы, начинается с текущей.

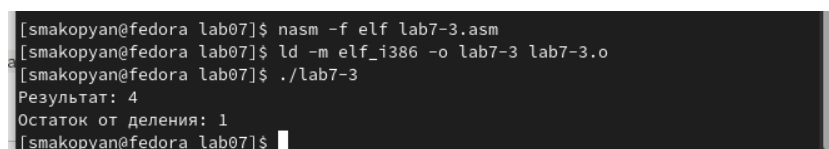
12. Создаем файл lab7-3.asm в каталоге ~/work/arch-pc/lab07 и вводим в него текст программы из листинга, данного в лабораторной работе (рис. 3.12)



```
lab7-3.asm [-M--] 13 L: [ 1+ 4 5/ 29] *(217 /1365b) 0010 0x00A [*][X]
; 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: '
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10Выход
```

Рис. 3.12: рисунок 12

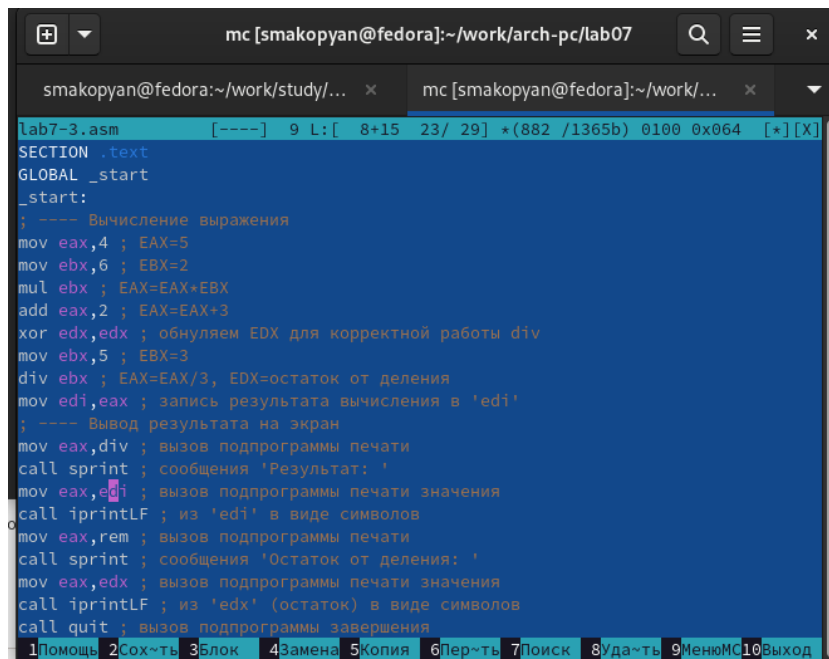
13. Создаем исполняемый файл и запускаем его (рис. 3.13)



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

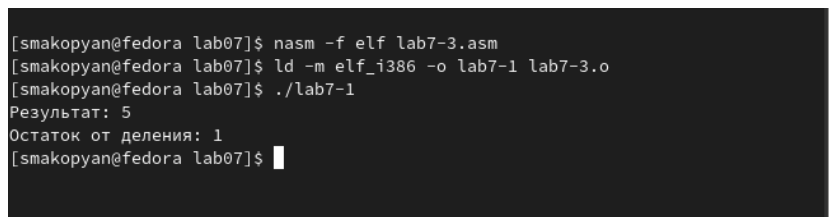
Рис. 3.13: рисунок 13

14. Изменим текст программы для вычисления выражения  $f(x) = (4 * 6 + 2) / 5$  (рис. 3.14), создадим исполняемый файл и проверим его работу (рис. 3.15)



```
lab7-3.asm  [----]  9  L:  8+15  23/ 29]  *(882 /1365b) 0100 0x064  [*][X]
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; 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
1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход
```

Рис. 3.14: рисунок 14



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

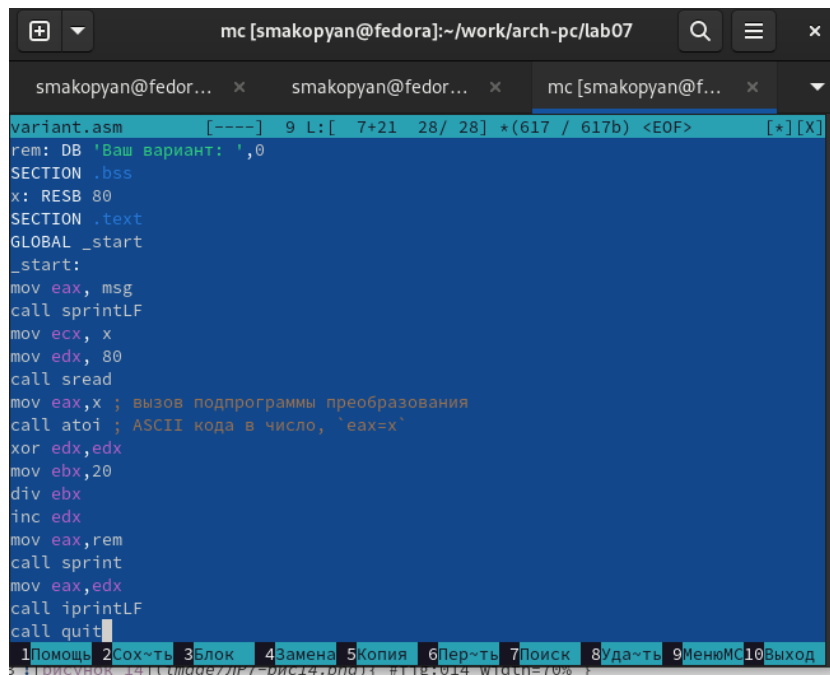
Рис. 3.15: рисунок 15

15. Создаем файл variant.asm в каталоге ~/work/arch-pc/lab07 (рис. 3.16), и вводим в него текст, представленный в лабораторной работе (рис. 3.17)



```
[smakopyan@fedora lab07]$ touch ~/work/arch-pc/lab07/variant.asm
[smakopyan@fedora lab07]$ mc
```

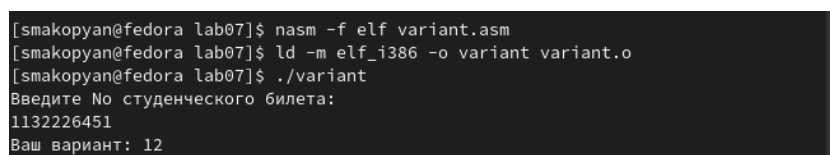
Рис. 3.16: рисунок 16



```
mc [smakopyan@fedora]:~/work/arch-pc/lab07
variant.asm  [----] 9 L: [ 7+21 28/ 28] *(617 / 617b) <EOF> [*] [X]
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
```

Рис. 3.17: рисунок 17

16. Создаем исполняемый файл и запускаем его (рис. 3.18)



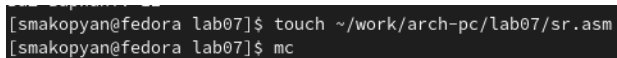
```
[smakopyan@fedora lab07]$ nasm -f elf variant.asm
[smakopyan@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[smakopyan@fedora lab07]$ ./variant
Введите No студенческого билета:
1132226451
Ваш вариант: 12
```

Рис. 3.18: рисунок 18

## 4 Задание для самостоятельной работы

Вариант 12.

1. Создаем файл, в котором запишем текст программы (рис. 4.1)

A terminal window with a dark background. The prompt is [smakopyan@fedora lab07]\$. The first command is touch ~/work/arch-pc/lab07/sr.asm. The second command is mc.

```
[smakopyan@fedora lab07]$ touch ~/work/arch-pc/lab07/sr.asm  
[smakopyan@fedora lab07]$ mc
```

Рис. 4.1: рисунок 19

2. Вводим текст программы в созданный файл (рис. 4.2)

```

smakopyan@fedora:~/work/study/2022-2023/Архитекту
sr.asm [----] 7 L:[ 10+15 25/ 31] *
_start: ; Чтение данных
mov eax, stm
call sprintLF
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi ; Вычисления
mov ebx, 8
mul ebx
sub eax, 6
xor edx, edx
mov ebx, 2
div ebx
mov edi, eax ; Вывод результата на экран
mov eax, res
call sprint
mov eax, edi
call iprintLF
call quit

```

Рис. 4.2: рисунок 20

3. Создаем исполняемый файл и проверяем его работу для значений 1 и 5 (рис. 4.3)

```

[smakopyan@fedora lab07]$ nasm -f elf sr.asm
[smakopyan@fedora lab07]$ ld -m elf_i386 -o sr sr.o
[smakopyan@fedora lab07]$ ./sr
y = (8*x - 6)/2
Введите значение x:
1
Результат: 1
[smakopyan@fedora lab07]$ ./sr
y = (8*x - 6)/2
Введите значение x:
5
Результат: 17
[smakopyan@fedora lab07]$

```

Рис. 4.3: рисунок 21

## 5 Выводы

В результате данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.



## **Список литературы**