

# **Отчёт по лабораторной работе №7.Арифметические операции в NASM.**

Королёв Иван Андреевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Символьные и численные данные в NASM</b>	<b>8</b>
<b>5</b>	<b>Выполнение арифметических операций в NASM</b>	<b>12</b>
<b>6</b>	<b>Выводы</b>	<b>16</b>

# Список иллюстраций

4.1	lab7-1	8
4.2	lab7-1	9
4.3	lab7-1	9
4.4	lab7-1	9
4.5	lab7-2	10
4.6	lab7-2	10
4.7	lab7-2	11
4.8	lab7-2	11
5.1	lab7-3	12
5.2	lab7-3	13
5.3	lab7-3	13
5.4	variant	14
5.5	variant	14
5.6	z3	15
5.7	z3	15

## Список таблиц

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Теоретическое введение

### 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:

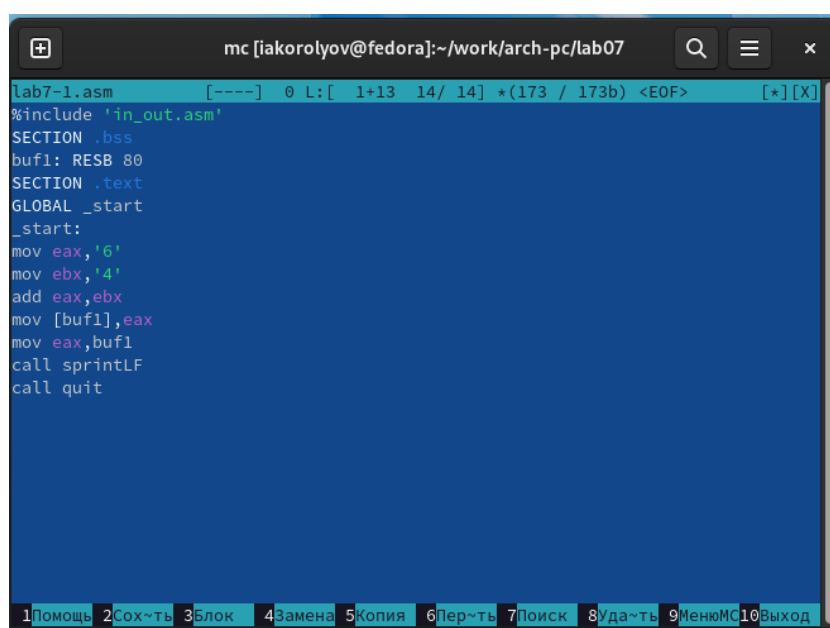
- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию

Например, определим переменную `intg DD 3` – это означает, что задается область памяти размером 4 байта, адрес которой обозначен меткой `intg`. В таком случае, команда `mov eax,[intg]` копирует из памяти по адресу `intg` данные в регистр `eax`. В свою очередь команда `mov [intg],eax` запишет в память по адресу `intg` данные из регистра `eax`. Также рассмотрим команду `mov eax,intg`. В этом случае в регистр `eax` запишется адрес `intg`. Допустим, для `intg` выделена память начиная с ячейки с адресом `0x600144`, тогда команда `mov eax,intg` аналогична команде `mov eax,0x600144` – т.е. эта команда запишет в регистр `eax` число `0x600144`.

### **3 Выполнение лабораторной работы**

## 4 Символьные и численные данные в NASM

1. Создаю каталог для программ лабораторной работы № 7, перейду в него и создам файл lab7-1.asm:
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax. Листинг 7.14.1, 4.2



```
lab7-1.asm  [----]  0  L: [ 1+13 14/ 14] *(173 / 173b) <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 sprintf
call quit
```

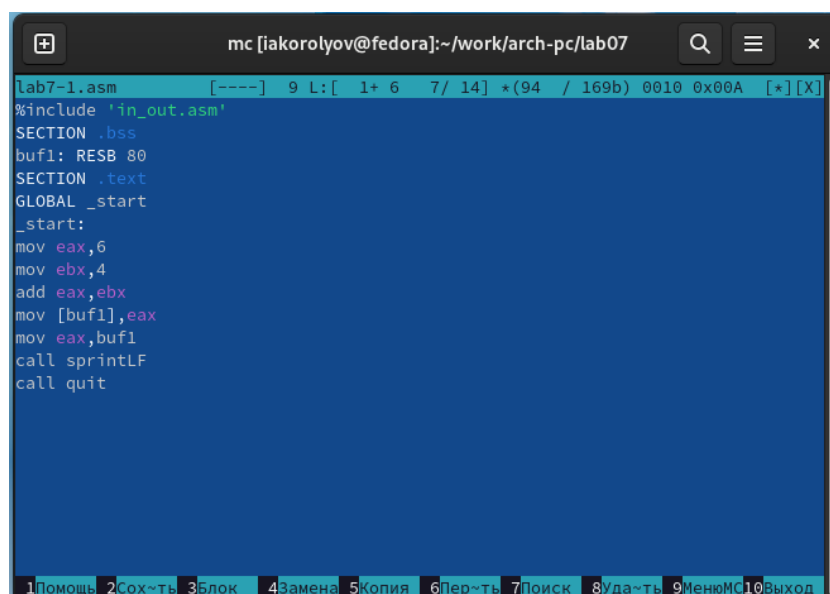
Рис. 4.1: lab7-1



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

Рис. 4.2: lab7-1

3. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 1) следующим образом. Не отображается 4.3, 4.4



```
lab7-1.asm [----] 9 L: [ 1+ 6 7/ 14] *(94 / 169b) 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
```

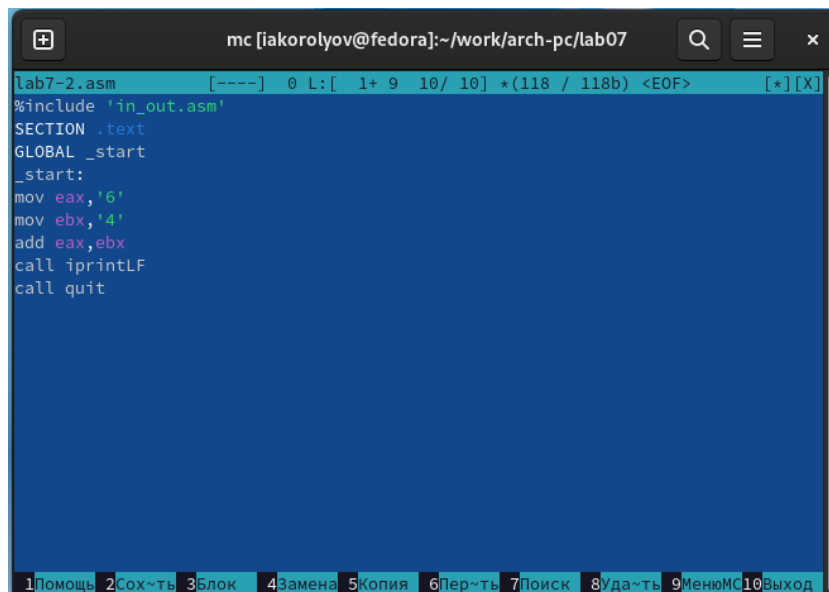
Рис. 4.3: lab7-1

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

[iakorolyov@fedora lab07]$
```

Рис. 4.4: lab7-1

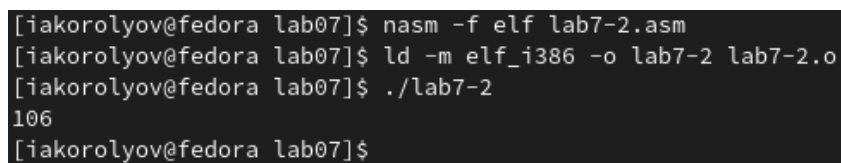
4. Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и введу в него текст программы из листинга 7.2.4.5, 4.6



```
lab7-2.asm  [----]  0 L: [ 1+ 9 10/ 10] *(118 / 118b) <EOF>  [*] [X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10Выход

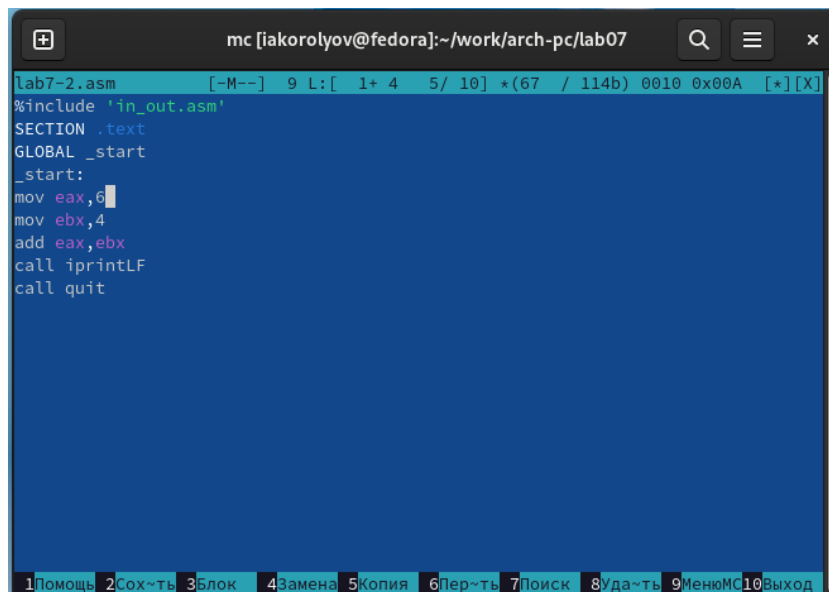
Рис. 4.5: lab7-2



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

Рис. 4.6: lab7-2

5. Аналогично предыдущему примеру изменим символы на числа. Замените строки. Не переходит на новую строку. 4.7, 4.8

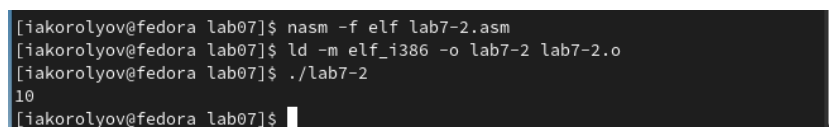


The screenshot shows a nano text editor window titled 'mc [iakorolyov@fedora]:~/work/arch-pc/lab07'. The editor is displaying the file 'lab7-2.asm'. The code content is as follows:

```
lab7-2.asm      [-M--]  9 L: [ 1+ 4 5/ 10] *(67 / 114b) 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
```

At the bottom of the window, there is a menu bar with the following items: 1Помощь, 2Сох~ть, 3Блок, 4Замена, 5Копия, 6Пер~ть, 7Поиск, 8Уда~ть, 9МенюМС, 10Выход.

Рис. 4.7: lab7-2



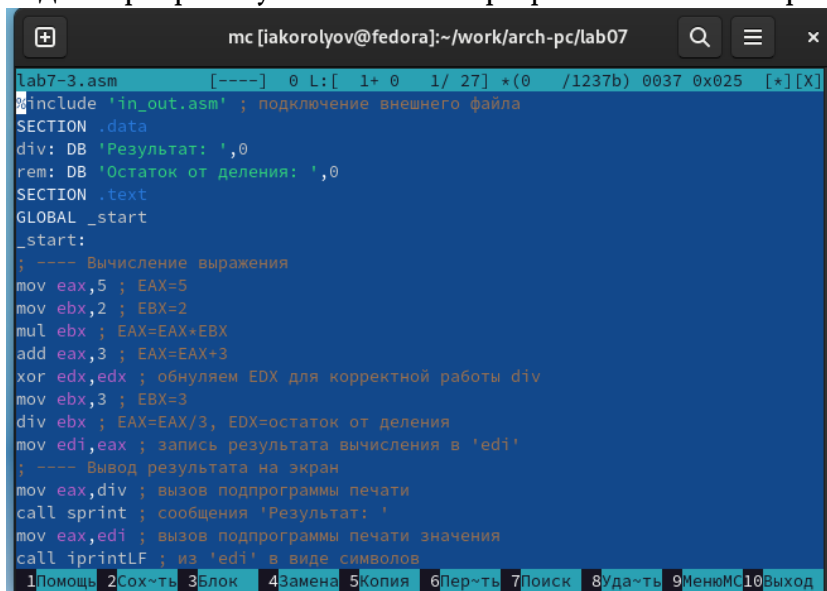
The screenshot shows a terminal window with the following commands and output:

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

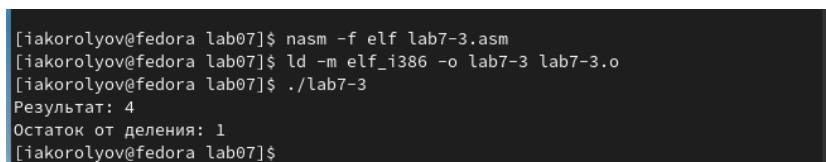
Рис. 4.8: lab7-2

## 5 Выполнение арифметических операций в NASM

6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения. ??,5.1,5.2,5.3



```
lab7-3.asm  [----]  0 L: [ 1+ 0 1/ 27] *(0 /1237b) 0037 0x025 [*][X]
#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
1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```



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

Рис. 5.1: lab7-3

```
lab7-3.asm  [----]  0  L: [ 7+20  27/ 27]  *(1237/1237b) <EOF>  [*] [X]
_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
```

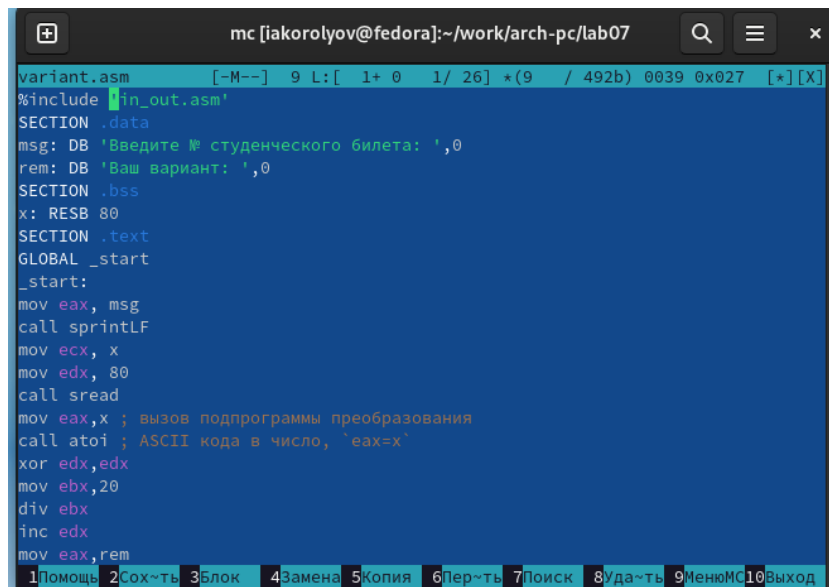
1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход

Рис. 5.2: lab7-3

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

Рис. 5.3: lab7-3

7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму5.4,5.5



```
variant.asm [-M--] 9 L: [ 1+ 0 1/ 26] *(9 / 492b) 0039 0x027 [*][X]
#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 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
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10Выход
```

Рис. 5.4: variant

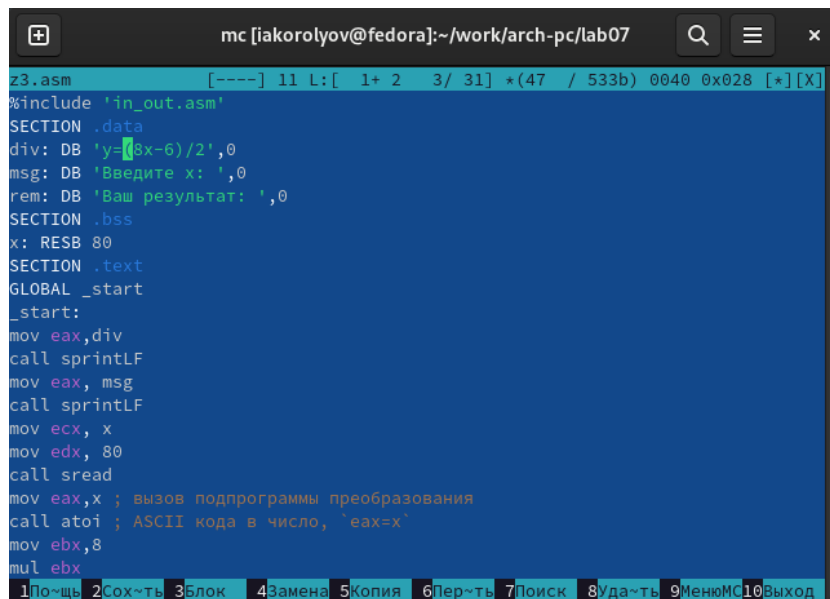


```
[iakorolyov@fedora lab07]$ nasm -f elf variant.asm
[iakorolyov@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[iakorolyov@fedora lab07]$ ./variant
Введите № студенческого билета:
1032225751
Ваш вариант: 12
[iakorolyov@fedora lab07]$
```

Рис. 5.5: variant

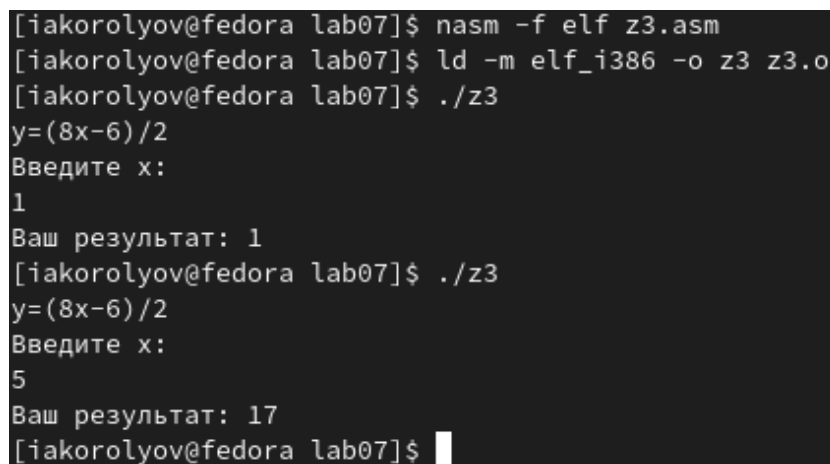
Ответы на вопросы:

1. mov eax, rem call sprint
2. Ввод переменной x
3. Преобразует строку string в целое значение типа int.
4. xor edx, edx mov ebx, 20 div ebx inc edx
5. В mov edi, eax
6. inc увеличивает на 1 свой операнд
7. mov eax, edx call iprintLF
8. Самостоятельная работа: 5.6, 5.7



```
z3.asm [----] 11 L: [ 1+ 2 3/ 31] *(47 / 533b) 0040 0x028 [*][X]
%include 'in_out.asm'
SECTION .data
div: DB 'y=(8x-6)/2',0
msg: DB 'Введите x: ',0
rem: DB 'Ваш результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,div
call sprintLF
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'
mov ebx,8
mul ebx
1По~щ~ 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

Рис. 5.6: z3



```
[iakorolyov@fedora lab07]$ nasm -f elf z3.asm
[iakorolyov@fedora lab07]$ ld -m elf_i386 -o z3 z3.o
[iakorolyov@fedora lab07]$ ./z3
y=(8x-6)/2
Введите x:
1
Ваш результат: 1
[iakorolyov@fedora lab07]$ ./z3
y=(8x-6)/2
Введите x:
5
Ваш результат: 17
[iakorolyov@fedora lab07]$
```

Рис. 5.7: z3

## 6 Выводы

Я освоил арифметические инструкции языка ассемблера NASM.