

Лабораторная работа №6

Работа с NASM

Ромицына Анасасия Романовна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задания для самостоятельной работы	14
4	Выводы	16

Список иллюстраций

2.1	Создаем каталог и переходим в него	6
2.2	Программа из листнинга	6
2.3	Запуск программы	7
2.4	Программа из листнинга	7
2.5	Запуск программы	7
2.6	Создание файла	8
2.7	Программа из листнинга	8
2.8	Запуск программы	8
2.9	Изменение текста программы	9
2.10	Запуск программы	9
2.11	Создание нового файла	9
2.12	Изменение текста программы	10
2.13	Запуск программы	10
2.14	Редактируем файл	11
2.15	Запуск программы	11
2.16	Создание файла	11
2.17	Заполнение файла	12
2.18	Запуск программы	12
3.1	Создание нового файла	14
3.2	Открытие файла	14
3.3	Содание новой программы	15
3.4	Запуск и проверка	15

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM # Задание
Написать собственную программу.

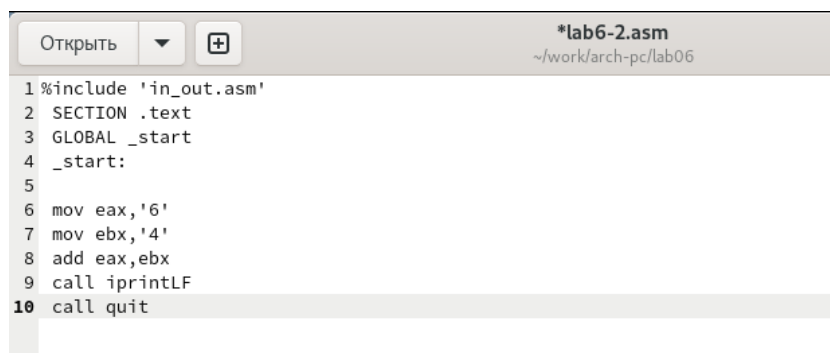
2 Выполнение лабораторной работы

Создаем каталог для программ лабораторной работы No 6, переходим в него и создаем файл lab6-1.asm: (рис. 2.1).

```
[romitsinaar@fedora ~]$ mkdir ~/work/arch-pc/lab06  
[romitsinaar@fedora ~]$ cd ~/work/arch-pc/lab06  
[romitsinaar@fedora lab06]$ touch lab6-1.asm  
[romitsinaar@fedora lab06]$ gedit lab6-1.asm
```

Рис. 2.1: Создаем каталог и переходим в него

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax. Вводим в файл lab6-1.asm текст программы из листинга 6.1.(рис. 2.2).



```
*lab6-2.asm  
~/work/arch-pc/lab06  
Открыть ▼ +  
1 %include 'in_out.asm'  
2 SECTION .text  
3 GLOBAL _start  
4 _start:  
5  
6 mov eax, '6'  
7 mov ebx, '4'  
8 add eax, ebx  
9 call iprintLF  
10 call quit
```

Рис. 2.2: Программа из листинга

Создайте исполняемый файл и запустите его.(рис. 2.3).

```

[romitsinaar@fedora lab06]$ nasm -f elf lab6-1.asm
[romitsinaar@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[romitsinaar@fedora lab06]$ ./lab6-1
j
[romitsinaar@fedora lab06]$

```

Рис. 2.3: Запуск программы

Далее изменим текст программы и вместо символов, запишем в регистры числа. Ис- правим текст программы (Листинг 6.1) следующим образом: (рис. 2.4).

```

1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov eax, 6
11 mov ebx, 4
12 add eax,ebx
13 mov [buf1],eax
14 mov eax,buf1
15 call sprintLF
16
17 call quit

```

Рис. 2.4: Программа из листнинга

Создайте исполняемый файл и запустите его (рис. 2.5).

```

[romitsinaar@fedora lab06]$ gedit lab6-1.asm
[romitsinaar@fedora lab06]$ nasm -f elf lab6-1.asm
[romitsinaar@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[romitsinaar@fedora lab06]$ ./lab6-1

[romitsinaar@fedora lab06]$

```

Рис. 2.5: Запуск программы

Согласно таблице код 10 соответствует переносу на след строку, это программа

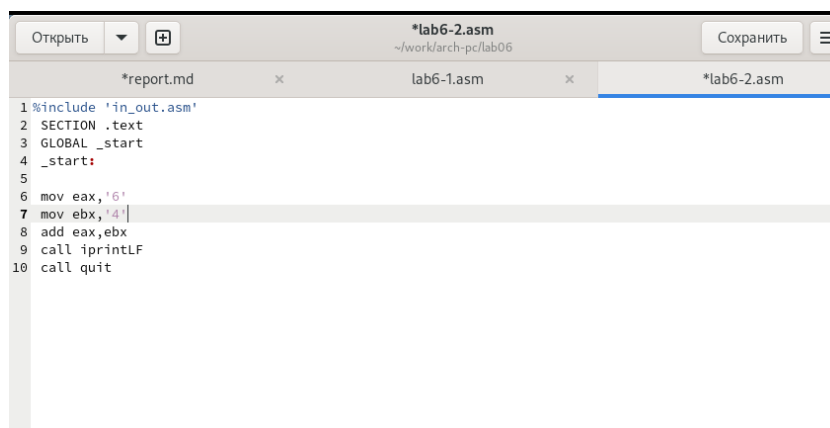
и выводит.

Создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. 2.6).

```
[romitsinaar@fedora lab06]$ touch lab6-2.asm
[romitsinaar@fedora lab06]$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
[romitsinaar@fedora lab06]$
```

Рис. 2.6: Создание файла

Введем в него текст программы из листинга 6.2 (рис. 2.7).



```
*lab6-2.asm
~/work/arch-pc/lab06

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5
6 mov eax, '6'
7 mov ebx, '4'
8 add eax, ebx
9 call iprintLF
10 call quit
```

Рис. 2.7: Программа из листинга

Создадим исполняемый файл и запустим его. (рис. 2.8).

```
[romitsinaar@fedora lab06]$ gedit lab6-2.asm
[romitsinaar@fedora lab06]$ nasm -f elf lab6-2.asm
[romitsinaar@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[romitsinaar@fedora lab06]$ ./lab6-2
106
[romitsinaar@fedora lab06]$
```

Рис. 2.8: Запуск программы

Аналогично предыдущему примеру изменим символы на числа. (рис. 2.9).

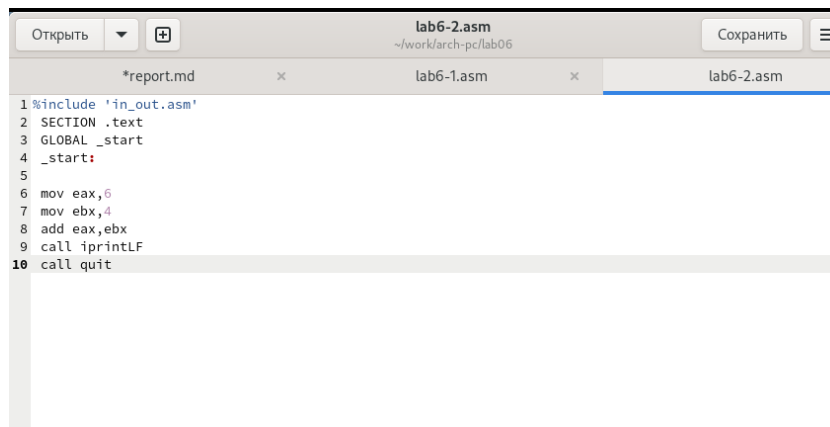


Рис. 2.9: Изменение текста программы

Создадим исполняемый файл и запустим его.(рис. 2.10).



Рис. 2.10: Запуск программы

Создадим файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. 2.11).

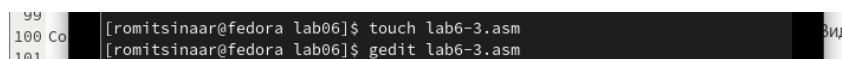
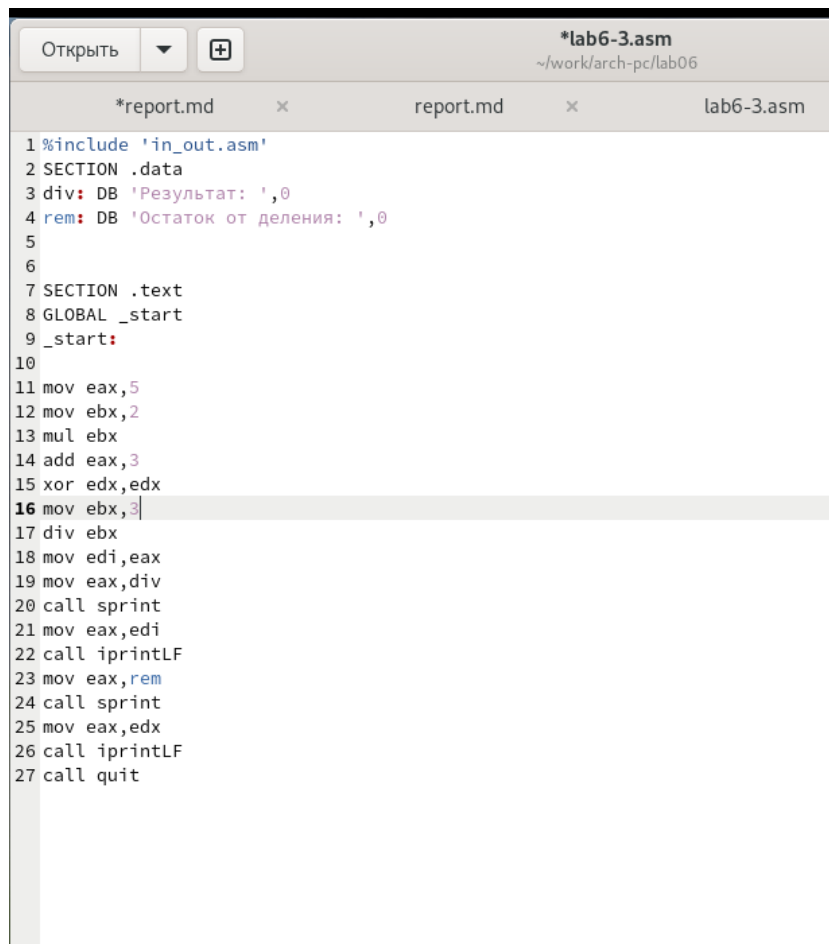


Рис. 2.11: Создание нового файла

Внимательно изучаем текст программы из листинга 6.3 и вводим в lab6-3.asm.(рис. 2.12).



```
*lab6-3.asm
~/work/arch-pc/lab06

*report.md x report.md x lab6-3.asm

1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5
6
7 SECTION .text
8 GLOBAL _start
9 _start:
10
11 mov eax,5
12 mov ebx,2
13 mul ebx
14 add eax,3
15 xor edx,edx
16 mov ebx,3
17 div ebx
18 mov edi,eax
19 mov eax,div
20 call sprint
21 mov eax,edi
22 call iprintLF
23 mov eax,rem
24 call sprint
25 mov eax,edx
26 call iprintLF
27 call quit
```

Рис. 2.12: Изменение текста программы

Создаем исполняемый файл и запускаем его. Результат работы программы должен быть следующим:(рис. 2.13).

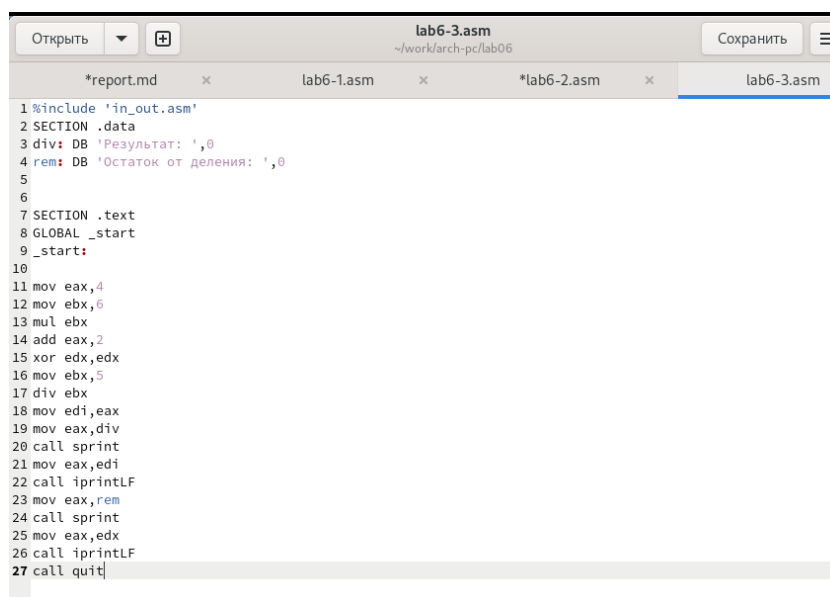


```
[romitsinaar@fedora lab06]$ touch lab6-3.asm
[romitsinaar@fedora lab06]$ gedit lab6-3.asm
[romitsinaar@fedora lab06]$ nasm -f elf lab6-3.asm
[romitsinaar@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[romitsinaar@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
[romitsinaar@fedora lab06]$
```

Рис. 2.13: Запуск программы

Изменяем текст программы для вычисления выражения $\boxed{\boxed{x}} = (4 \times 6 + 2)/5$.


(рис. 2.14).



```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5
6
7 SECTION .text
8 GLOBAL _start
9 _start:
10
11 mov eax,4
12 mov ebx,6
13 mul ebx
14 add eax,2
15 xor edx,edx
16 mov ebx,5
17 div ebx
18 mov edi,eax
19 mov eax,div
20 call sprint
21 mov eax,edi
22 call iprintLF
23 mov eax,rem
24 call sprint
25 mov eax,edx
26 call iprintLF
27 call quit
```

Рис. 2.14: Редактируем файл

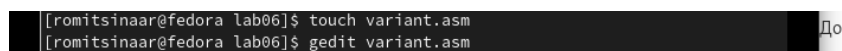
Создаем исполняемый файл и проверяем его работу. (рис. 2.15).



```
[romitsinaar@fedora lab06]$ gedit lab6-3.asm
[romitsinaar@fedora lab06]$
[romitsinaar@fedora lab06]$ nasm -f elf lab6-3.asm
[romitsinaar@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[romitsinaar@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[romitsinaar@fedora lab06]$
```

Рис. 2.15: Запуск программы

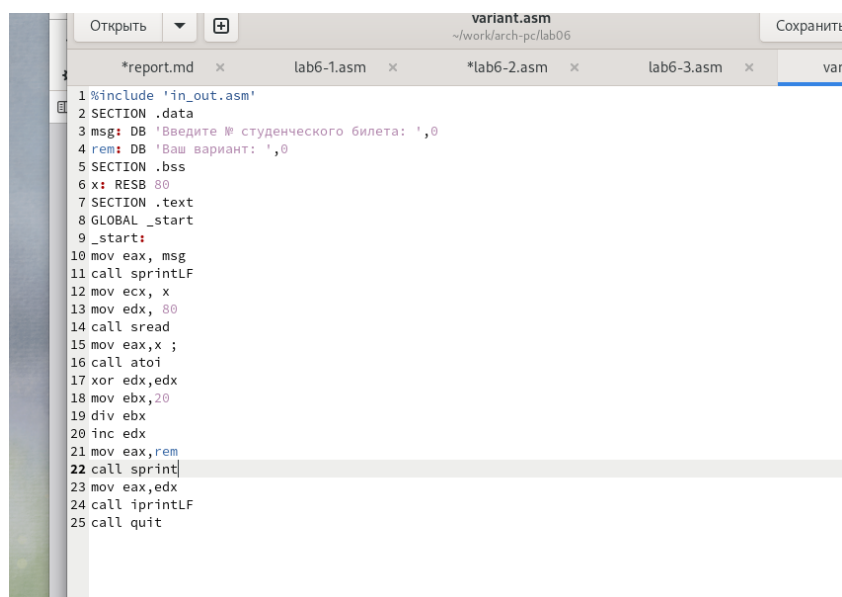
Создаем файл variant.asm в каталоге ~/work/arch-pc/lab06: рис. 2.16).



```
[romitsinaar@fedora lab06]$ touch variant.asm
[romitsinaar@fedora lab06]$ gedit variant.asm
```

Рис. 2.16: Создание файла

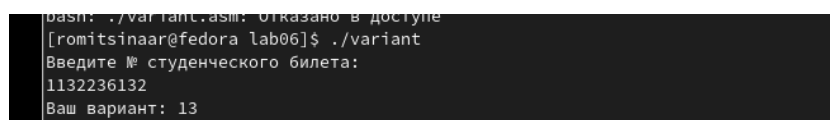
Внимательно изучаем текст программы из листинга 6.4 и вводим в файл variant.asm.(рис. 2.17).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого билета: ',0
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x ;
16 call atoi
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax, rem
22 call sprintf
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рис. 2.17: Заполнение файла

Создаем исполняемый файл и запускаем его. (рис. 2.18).



```
bash: ./variant.asm: Отказано в доступе
[romitsinaar@fedora lab06]$ ./variant
Введите № студенческого билета:
1132236132
Ваш вариант: 13
```

Рис. 2.18: Запуск программы

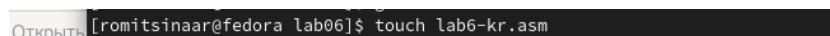
Ответы на вопросы

1 Строка “mov eax,rem” и строка “call sprint” отвечают за вывод на экран сообщения ‘Ваш вариант:’. 2 Эти инструкции используются для чтения строки с вводом данных от пользователя. Начальный адрес строки сохраняется в регистре ecx, а количество символов в строке (максимальное количество символов, которое может быть считано) сохраняется в регистре edx. Затем вызывается процедура sread, которая выполняет чтение строки. 3 Инструкция “call atoi” используется для преобразования строки в целое число. Она принимает адрес строки в регистре eax и возвращает полученное число в регистре eax. 4 Строка “xor edx,edx” обнуляет регистр edx перед выполнением деления. Строка “mov ebx,20” загру-

жает значение 20 в регистр ebx. Строка “div ebx” выполняет деление регистра eax на значение регистра ebx с сохранением частного в регистре eax и остатка в регистре edx. 5 Остаток от деления записывается в регистр edx. 6 Инструкция “inc edx” используется для увеличения значения в регистре edx на 1. В данном случае, она увеличивает остаток от деления на 1. 7 Строка “mov eax,edx” передает значение остатка от деления в регистр eax. Строка “call iprintLF” вызывает процедуру iprintLF для вывода значения на экран вместе с переводом строки.

3 Задания для самостоятельной работы

Создадим файл для выполнения самостоятельной работы . (рис. 3.1).



```
Открыть [romitsinaar@fedora lab06]$ touch lab6-kr.asm
```

Рис. 3.1: Создание нового файла

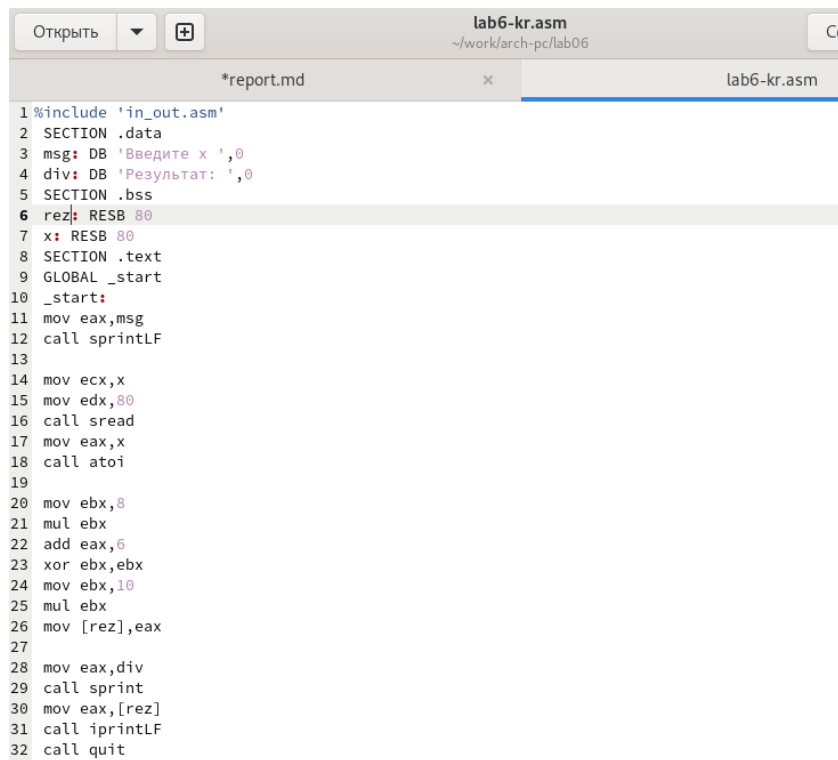
Откроем новый файл в редакторе. (рис. 3.2).



```
save to lab6-kr.asm - create a new file combination of opcode and operands  
[romitsinaar@fedora lab06]$ gedit lab6-kr.asm  
^[[2~  
ret  
1 23:26 28 mov [res
```

Рис. 3.2: Открытие файла

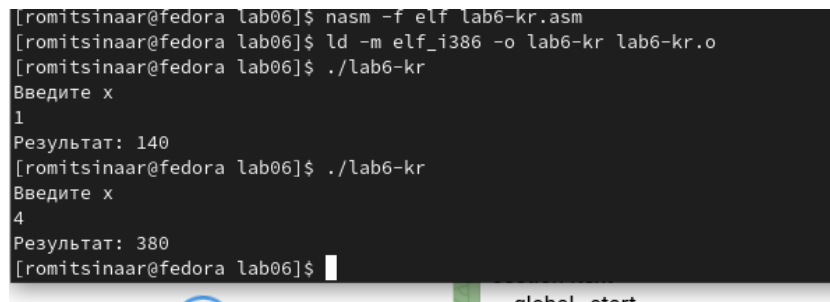
Напишем программу для выполнения задания. (рис. 3.3).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x ',0
4 div: DB 'Результат: ',0
5 SECTION .bss
6 rez: RESB 80
7 x: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 mov eax,msg
12 call sprintf
13
14 mov ecx,x
15 mov edx,80
16 call sread
17 mov eax,x
18 call atoi
19
20 mov ebx,8
21 mul ebx
22 add eax,6
23 xor ebx,ebx
24 mov ebx,10
25 mul ebx
26 mov [rez],eax
27
28 mov eax,div
29 call sprintf
30 mov eax,[rez]
31 call iprintLF
32 call quit
```

Рис. 3.3: Содание новой программы

Запустим программу и проверим правильность её работы. Все работает верно.
(рис. 3.4).



```
[romitsinaar@fedora lab06]$ nasm -f elf lab6-kr.asm
[romitsinaar@fedora lab06]$ ld -m elf_i386 -o lab6-kr lab6-kr.o
[romitsinaar@fedora lab06]$ ./lab6-kr
Введите x
1
Результат: 140
[romitsinaar@fedora lab06]$ ./lab6-kr
Введите x
4
Результат: 380
[romitsinaar@fedora lab06]$
```

Рис. 3.4: Запуск и проверка

4 Выводы

Мы освоили арифметические инструкции языка ассемблера NASM.