

Арифметические операции в NASM

Лабораторная работа №6

Приходько Иван Иванович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	17
4	Выводы	21

Список иллюстраций

2.1	Создание файла lab6-1.asm	6
2.2	Запись кода в файл lab6-1.asm	6
2.3	Рабочее пространство на данный момент	7
2.4	Сборка lab6-1.asm	7
2.5	Запуск lab6-1.asm	7
2.6	Редактирование файла lab6-1.asm	8
2.7	Повторный запуск lab6-1.asm	8
2.8	Создание файла lab6-2.asm	8
2.9	Редактирование файла lab6-2.asm	9
2.10	Сборка и запуск lab6-2.asm	9
2.11	Повторное редактирование файла lab6-2.asm	9
2.12	Повторный запуск lab6-2.asm	10
2.13	Создание lab6-3.asm	10
2.14	Редактирование lab6-3.asm	11
2.15	Сборка и запуск lab6-3.asm	11
2.16	Повторное редактирование lab6-3.asm	12
2.17	Повторная сборка и запуск lab6-3.asm	13
2.18	Создание variant.asm	13
2.19	Редактирование variant.asm	14
2.20	Запуск variant.asm	15
3.1	Создание task6.asm	17
3.2	Редактирование task6.asm	18
3.3	Сборка и запуск task6.asm	19
3.4	Повторные запуски task6.asm	20

Список таблиц

1 Цель работы

Познакомиться с базовыми инструкциями языка Ассемблер, отвечающими за основные арифметические операции.

2 Выполнение лабораторной работы

Для начала выполнения лабораторной работы необходимо создать файл lab6-1.asm (рис. 2.1).

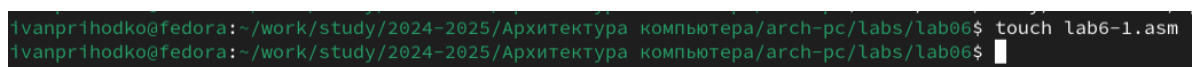
A terminal window showing the command 'touch lab6-1.asm' being executed in a directory path: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06. The prompt is 'ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06\$'.

Рис. 2.1: Создание файла lab6-1.asm

Вставим в наш созданный файл код из листинга 6.1 (рис. 2.2).

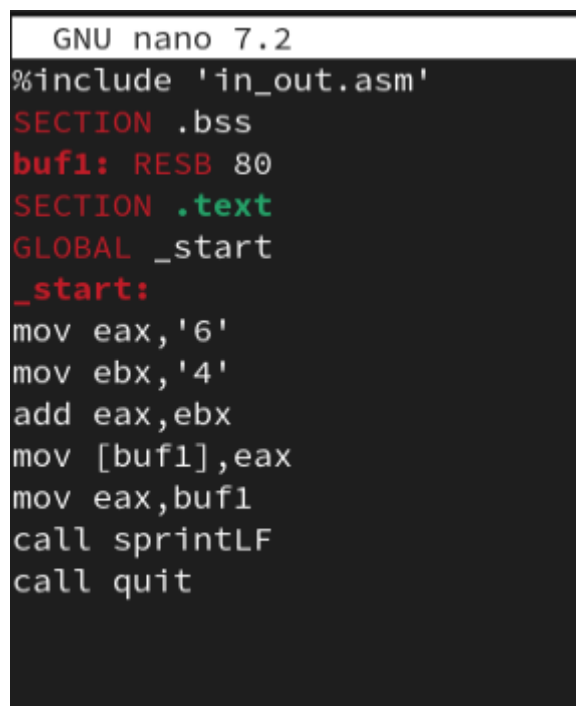
A screenshot of the GNU nano 7.2 text editor. The code being edited is assembly code. It starts with '%include \'in_out.asm\'', followed by a '.bss' section with a buffer 'buf1' of size 80. Then it moves to a '.text' section, declares '_start' as global, and defines the '_start' function. The function contains instructions to move the character '6' into 'eax', '4' into 'ebx', add them, store the result in 'buf1', and then call 'sprintf' and 'quit'.

Рис. 2.2: Запись кода в файл lab6-1.asm

Перед началом работы скопируем файл in_out.asm для корректной работы (рис. 2.3).

```
.M
/..
/presentation
/report
in_out.asm
lab6-1.asm
```

Рис. 2.3: Рабочее пространство на данный момент

Соберем и запустим программу (рис. 2.4 и 2.5).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf lab6-1.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
```

Рис. 2.4: Сборка lab6-1.asm

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./lab6-1
j
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$
```

Рис. 2.5: Запуск lab6-1.asm

Нам выводит символ j, однако это неправильный вывод. Наша цель - сложить 6 и 4, и получить в выводе число 10. Попробуем изменить наш файл (рис. 2.6).

```
GNU nano 7.2
#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
```

Рис. 2.6: Редактирование файла lab6-1.asm

Теперь соберем и запустим файл заново (рис. 2.7).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf lab6-1.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./lab6-1
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$
```

Рис. 2.7: Повторный запуск lab6-1.asm

Когда мы вызываем команду `sprintLF`, она выводит не число 10, а символ с номером 10. Символ под номером 10 это символ перевода строки. Теперь создадим второй файл под названием `lab6-2.asm` (рис. 2.8).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ touch lab6-2.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$
```

Рис. 2.8: Создание файла lab6-2.asm

Теперь вставим в него код из листинга 6.2 (рис. 2.9).


```

GNU nano 7.2
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit

```

Рис. 2.9: Редактирование файла lab6-2.asm

Соберем и запустим файл (рис. 2.10).

```

ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf lab6-2.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./lab6-2
106

```

Рис. 2.10: Сборка и запуск lab6-2.asm

Мы видим число 106. Так как цифры в коде указаны в кавычках, мы складываем их коды (54 и 52 в сумме дают 106). Изменим файл (рис. 2.11).

```

GNU nano 7.2
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit

```

Рис. 2.11: Повторное редактирование файла lab6-2.asm

Теперь запустим файл (рис. 2.12).

```
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf lab6-2.asm
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./lab6-2
10
```

Рис. 2.12: Повторный запуск lab6-2.asm

Создадим третий файл, вставим в него код из листинга 6.3, соберем и запустим его (рис. 2.13-2.15).

```
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ touch lab6-3.asm
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$
```

Рис. 2.13: Создание lab6-3.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,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. 2.14: Редактирование lab6-3.asm

```

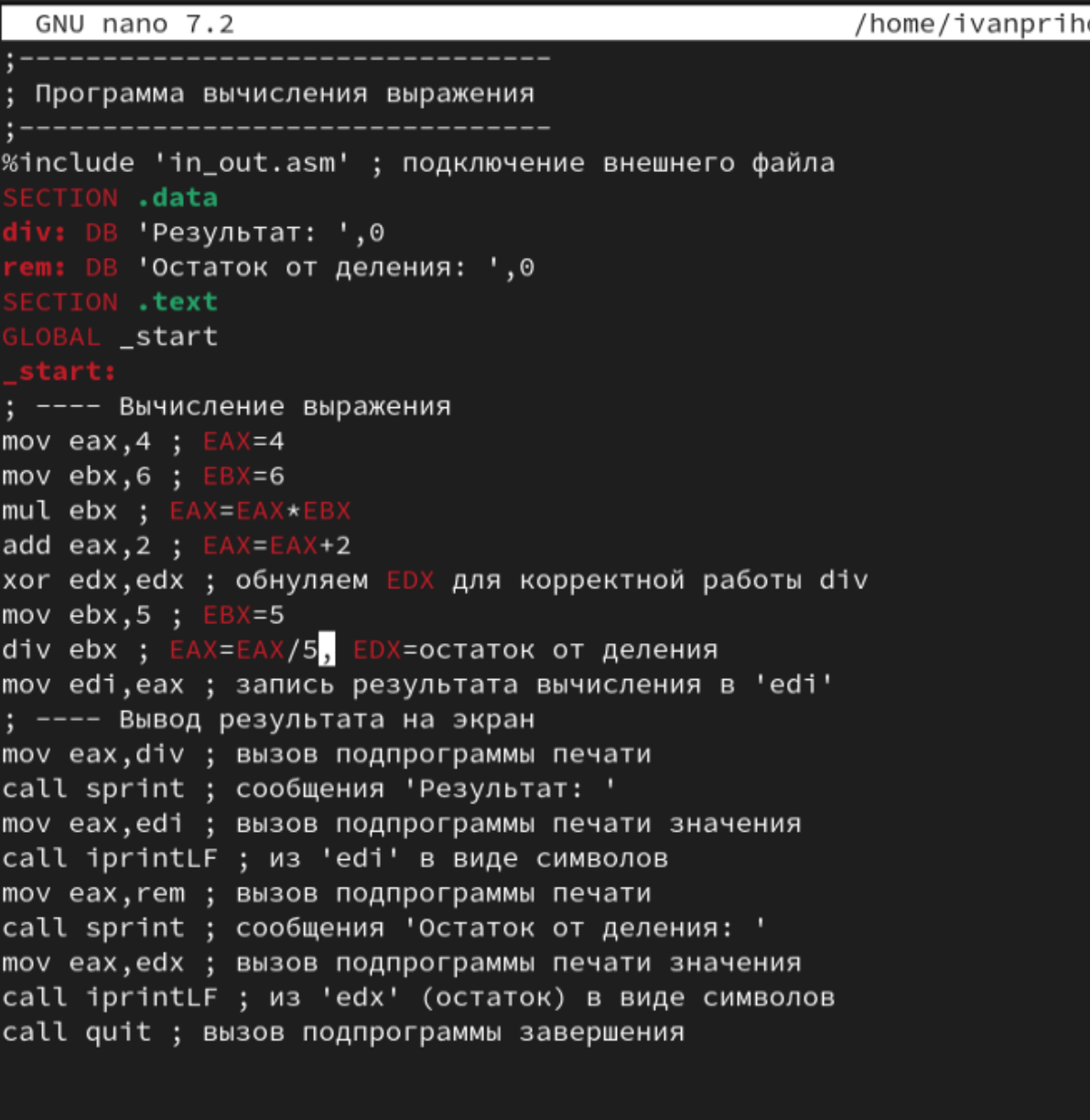
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf lab6-3.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$

```

Рис. 2.15: Сборка и запуск lab6-3.asm

Полученный результат совпадает с результатом, указанным в лабораторной

работе. Теперь изменим файл так, чтобы он вычислял значение выражения $(4*6+2)/5$. (рис. 2.16).



```
GNU nano 7.2 /home/ivanprih
;-----
; 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
```

Рис. 2.16: Повторное редактирование lab6-3.asm

Соберем и запустим его (рис. 2.17).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf lab6-3.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$
```

Рис. 2.17: Повторная сборка и запуск lab6-3.asm

Теперь создадим файл variant.asm для вычисления варианта самостоятельной работы и вставим в него код из листинга 6.4 (рис. 2.18 и 2.19).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ touch variant.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$
```

Рис. 2.18: Создание variant.asm

```
GNU nano 7.2 /home/iva
;-----
; Программа вычисления варианта
;-----
#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 iprintLF
call quit
```

Рис. 2.19: Редактирование variant.asm

Запишем номер своего студенческого и узнаем номер варианта (рис. 2.20).

```
ivanprikhodko@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf variant.asm
ivanprikhodko@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o variant variant.o
ivanprikhodko@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./variant
Введите № студенческого билета:
1132246285
Ваш вариант: 6
ivanprikhodko@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$
```

Рис. 2.20: Запуск variant.asm

Теперь отвечу на предложенные в лабораторной работе вопросы:

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

За это отвечает строчка `call sprint` в связке с `mov eax,rem`

2. Для чего используются следующие инструкции?

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

Эти строки используются для того, чтобы записать данные в переменную `x`

3. Для чего используется инструкция "call atoi"?

Для преобразования ASCII кода в число

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

```
div ebx - для деления
inc edx - для прибавки единицы
```

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

В регистр `edx`

6. Для чего используется инструкция "inc edx"?

Для увеличения значения регистра `edx` на единицу

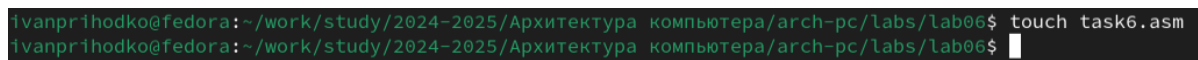
7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

`mov eax,edx` - для переноса значения регистра `edx` в `eax`

`call iprintLF` - выводит значение из регистра `eax`

3 Задание для самостоятельной работы

Теперь в качестве самостоятельной работы напомним код программы для вычисления выражения в варианте 6: $(x^3)/2+1$ (рис. 3.1).



```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ touch task6.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$
```

Рис. 3.1: Создание task6.asm

```
GNU nano 7.2 /
#include 'in_out.asm'
SECTION .data
msg: DB 'Выражение для вычисления (x^3)/2+1: ',0
msg2: DB 'Введите x: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov eax, msg2
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
push eax
mov ebx, eax
mul ebx
mov ebx, eax
pop eax
mul ebx
mov ebx, 2
div ebx
add eax, 1
call iprintLF
call quit
```

Рис. 3.2: Редактирование task6.asm

```
mov eax, msg
call sprintLF
mov eax, msg2
call sprintLF
```

Эти строчки отвечают за вывод сообщений пользователю

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

Эти строчки отвечают за считывание x у пользователя

```
mov eax, x
call atoi
```

Эти строчки конвертируют все строки в десятичные числа

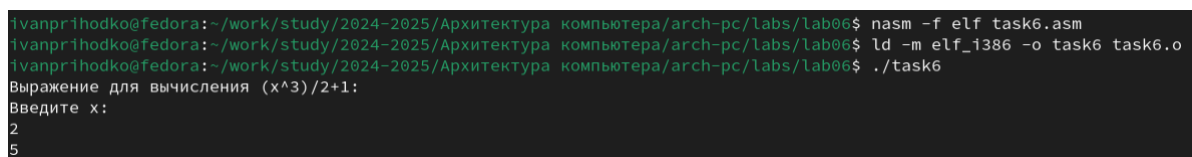
```
push eax
mov ebx, eax
mul ebx
mov ebx, eax
pop eax
mul ebx
```

Эти строчки отвечают за счет x^3 , сначала мы запоминаем оригинальное значение x , потом считаем x^2 , потом умножаем x на x^2

```
mov ebx, 2
div ebx
add eax, 1
```

Эти строчки отвечают за деление на 2 и прибавку единицы

Теперь соберем и запустим код (рис. 3.3).



```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf task6.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o task6 task6.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./task6
Выражение для вычисления (x^3)/2+1:
Введите x:
2
5
```

Рис. 3.3: Сборка и запуск task6.asm

Все правильно, проведем еще несколько тестов (рис. 3.4).

```
ivanpr1hodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./task6
Выражение для вычисления (x^3)/2+1:
Введите x:
4
33
ivanpr1hodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab06$ ./task6
Выражение для вычисления (x^3)/2+1:
Введите x:
6
109
```

Рис. 3.4: Повторные запуски task6.asm

4 Выводы

В результате выполнения лабораторной работы было получено представление о том, какие арифметические операции есть в языке Ассемблера, и как они работают. Были написаны программы, использующие в себе операции сложения, вычитания, умножения и деления.