

# **ОТЧЕТ по лабораторной работе №8**

Полякова Юлия Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Результаты выполнения лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Результаты выполнения заданий для самостоятельной работы</b>	<b>12</b>
<b>4</b>	<b>Вывод</b>	<b>15</b>

# Список иллюстраций

2.1	Создание каталога и lab8-1.asm . . . . .	6
2.2	Запуск lab8-1.asm из листинга 8.1. . . . .	7
2.3	Измененный lab8-1.asm . . . . .	7
2.4	Запуск измененного lab8-1.asm . . . . .	8
2.5	Измененный второй раз lab8-1.asm . . . . .	8
2.6	Запуск второй раз измененного lab8-1.asm . . . . .	8
2.7	lab8-2.asm из листинга 8.2. . . . .	9
2.8	Запуск lab8-2.asm из листинга 8.2. . . . .	9
2.9	lab8-3.asm из листинга 8.3. . . . .	10
2.10	Запуск lab8-3.asm из листинга 8.3. . . . .	10
2.11	lab8-3.asm, но для подсчета произведения . . . . .	11
2.12	Запуск измененного lab8-3.asm . . . . .	11
3.1	Файл samrab.asm с выполненной программой . . . . .	13
3.2	Запуск samrab.asm . . . . .	14

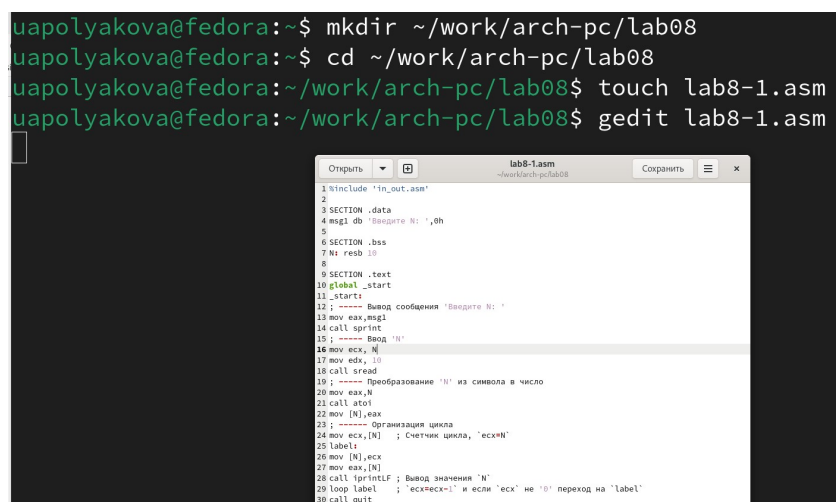
## **Список таблиц**

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

## 2 Результаты выполнения лабораторной работы

1. Создаем каталог для программ лабораторной №8, переходим в него и создаем файл lab8-1.asm. Записываем в файл программу из листинга 8.1. (Рис. 1).



```
uapolyakova@fedora:~$ mkdir ~/work/arch-pc/lab08
uapolyakova@fedora:~$ cd ~/work/arch-pc/lab08
uapolyakova@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
uapolyakova@fedora:~/work/arch-pc/lab08$ gedit lab8-1.asm
```

```
1 #include "in_out.asm"
2
3 SECTION .data
4 msg1 db "Введите N: ",0h
5
6 SECTION .bss
7 N: resb 10
8
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения "Введите N: "
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод "N"
16 mov ecx, N
17 mov edx, 10
18 call read
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, "ескN"
25 label:
26 mov [N],ecx
27 mov eax,[N]
28 call sprintf ; Вывод значения "N"
29 loop label ; "ескN" и если "еск" не '0' переход на "label"
30 call quit
```

Рис. 2.1: Создание каталога и lab8-1.asm

2. Создаем исполняемый файл и запускаем его (Рис. 2).

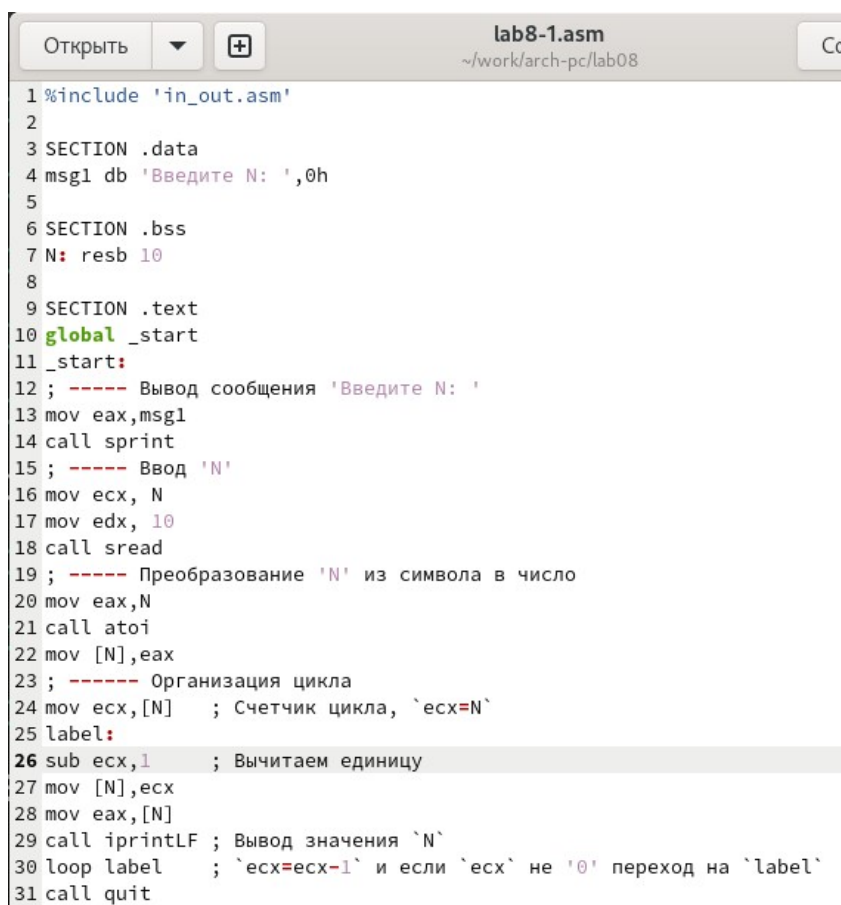
```

uapolyakova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
uapolyakova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
uapolyakova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
uapolyakova@fedora:~/work/arch-pc/lab08$

```

Рис. 2.2: Запуск lab8-1.asm из листинга 8.1.

3. Изменяем текст программы для примера некорректной работы (Рис. 3).



```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите N: ',0h
5
6 SECTION .bss
7 N: resb 10
8
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, `ecx=N`
25 label:
26 sub ecx,1 ; Вычитаем единицу
27 mov [N],ecx
28 mov eax,[N]
29 call iprintLF ; Вывод значения `N`
30 loop label ; `ecx=ecx-1` и если `ecx` не `0` переход на `label`
31 call quit

```

Рис. 2.3: Измененный lab8-1.asm

4. Создаем исполняемый файл и запускаем его (Рис. 4)

```
4294620652
4294620650
4294620648
4294620646
4294620644
4294620642
4294620640
4294620638
4294620636^Z
[1]+  Остановлен      ./lab8-1
uapolyakova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.4: Запуск измененного lab8-1.asm

Регистр принимает значение каждый раз на 2 меньше, так как loop вычитает 1, и мы принудительно вычитаем 1. Цикл стал бесконечным, число проходов не соответствует введенному числу.

5. Изменяем текст программы, используя стек, чтобы сохранить корректность работы (Рис. 5).

```
25 label:
26 push ecx
27 sub ecx,1      ; Вычитаем единицу
28 mov [N],ecx
29 mov eax,[N]
30 call iprintLF ; Вывод значения `N`
31 pop ecx
32 loop label     ; `ecx=ecx-1` и если `ecx` не `0` переход на `label`
33 call quit
```

Рис. 2.5: Измененный второй раз lab8-1.asm

6. Создаем исполняемый файл и запускаем его (Рис. 6).

```
uapolyakova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
uapolyakova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
uapolyakova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
uapolyakova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.6: Запуск второй раз измененного lab8-1.asm



Число проходов цикла соответствует N.

7. Создаем lab8-2.asm по листингу 8.2. (Рис. 7).

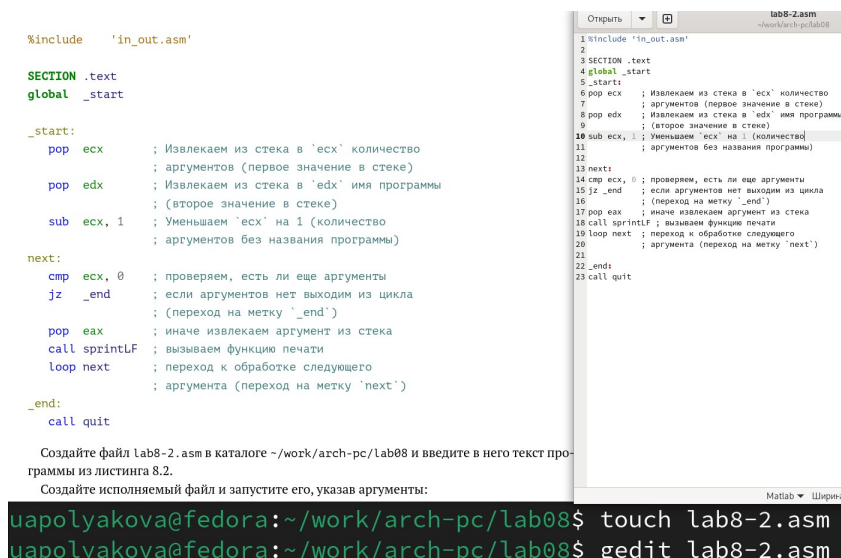


Рис. 2.7: lab8-2.asm из листинга 8.2.

8. Создаем исполняемый файл и запускаем его с указанными аргументами (Рис. 8).

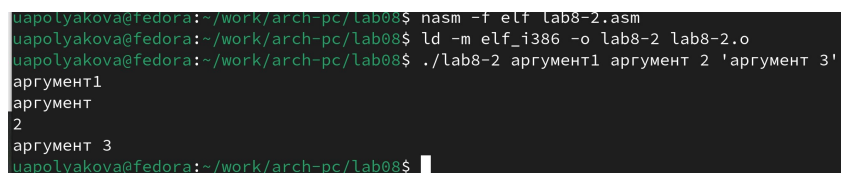


Рис. 2.8: Запуск lab8-2.asm из листинга 8.2.

Программой было обработано 4 аргумента, то есть все те, которые были указаны через пробел.

9. Создаем lab8-3.asm по листингу 8.3. (Рис. 9).

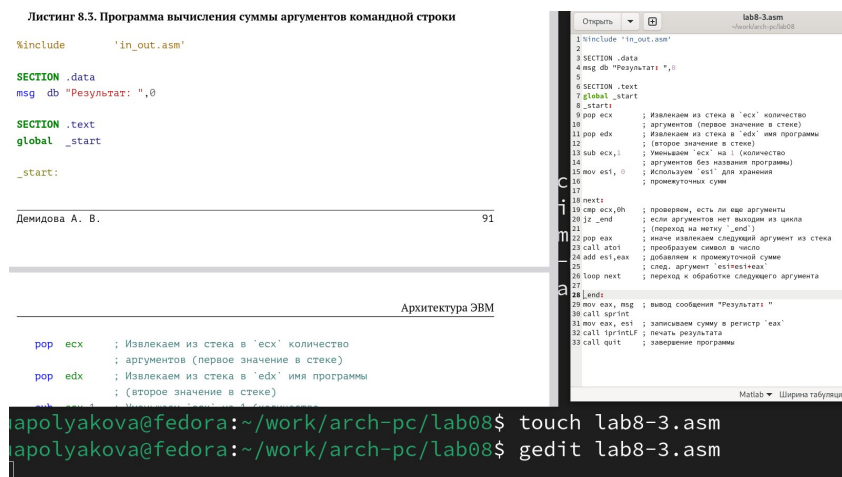


Рис. 2.9: lab8-3.asm из листинга 8.3.

10. Создаем исполняемый файл и запускаем его, указав аргументы (Рис. 10).

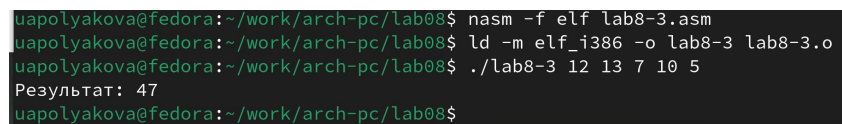
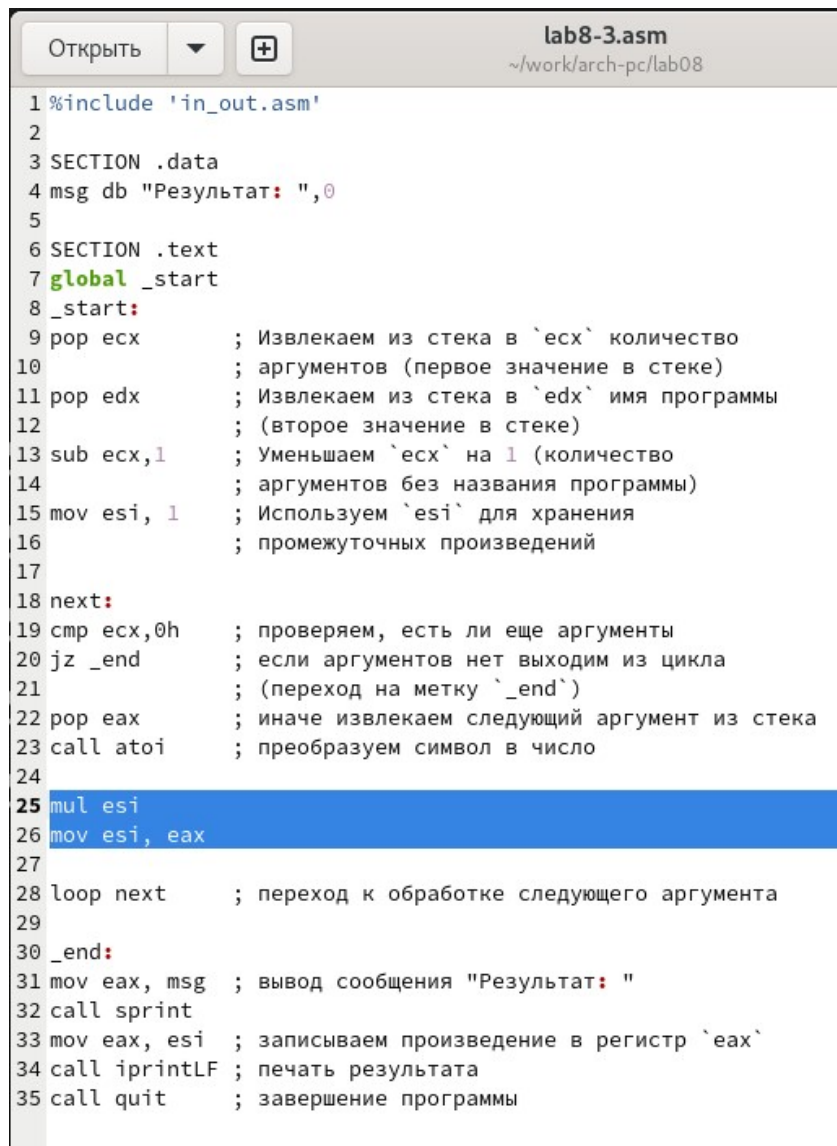


Рис. 2.10: Запуск lab8-3.asm из листинга 8.3.

11. Изменяем текст lab8-3.asm так, чтобы программа считала произведение аргументов (Рис. 11).

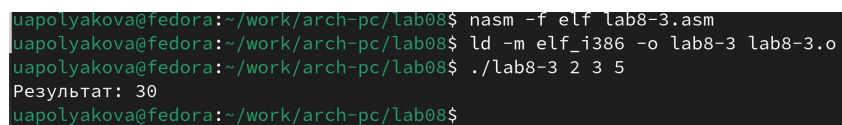


```
lab8-3.asm
~/work/arch-pc/lab08

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ",0
5
6 SECTION .text
7 global _start
8 _start:
9 pop ecx      ; Извлекаем из стека в `ecx` количество
10             ; аргументов (первое значение в стеке)
11 pop edx      ; Извлекаем из стека в `edx` имя программы
12             ; (второе значение в стеке)
13 sub ecx,1    ; Уменьшаем `ecx` на 1 (количество
14             ; аргументов без названия программы)
15 mov esi, 1   ; Используем `esi` для хранения
16             ; промежуточных произведений
17
18 next:
19 cmp ecx,0h   ; проверяем, есть ли еще аргументы
20 jz _end      ; если аргументов нет выходим из цикла
21             ; (переход на метку `_end`)
22 pop eax      ; иначе извлекаем следующий аргумент из стека
23 call atoi    ; преобразуем символ в число
24
25 mul esi
26 mov esi, eax
27
28 loop next    ; переход к обработке следующего аргумента
29
30 _end:
31 mov eax, msg ; вывод сообщения "Результат: "
32 call sprint
33 mov eax, esi ; записываем произведение в регистр `eax`
34 call iprintLF ; печать результата
35 call quit    ; завершение программы
```

Рис. 2.11: lab8-3.asm, но для подсчета произведения

12. Создаем исполняемый файл и запускаем его, указав аргументы, проверяем работу (Рис. 12).

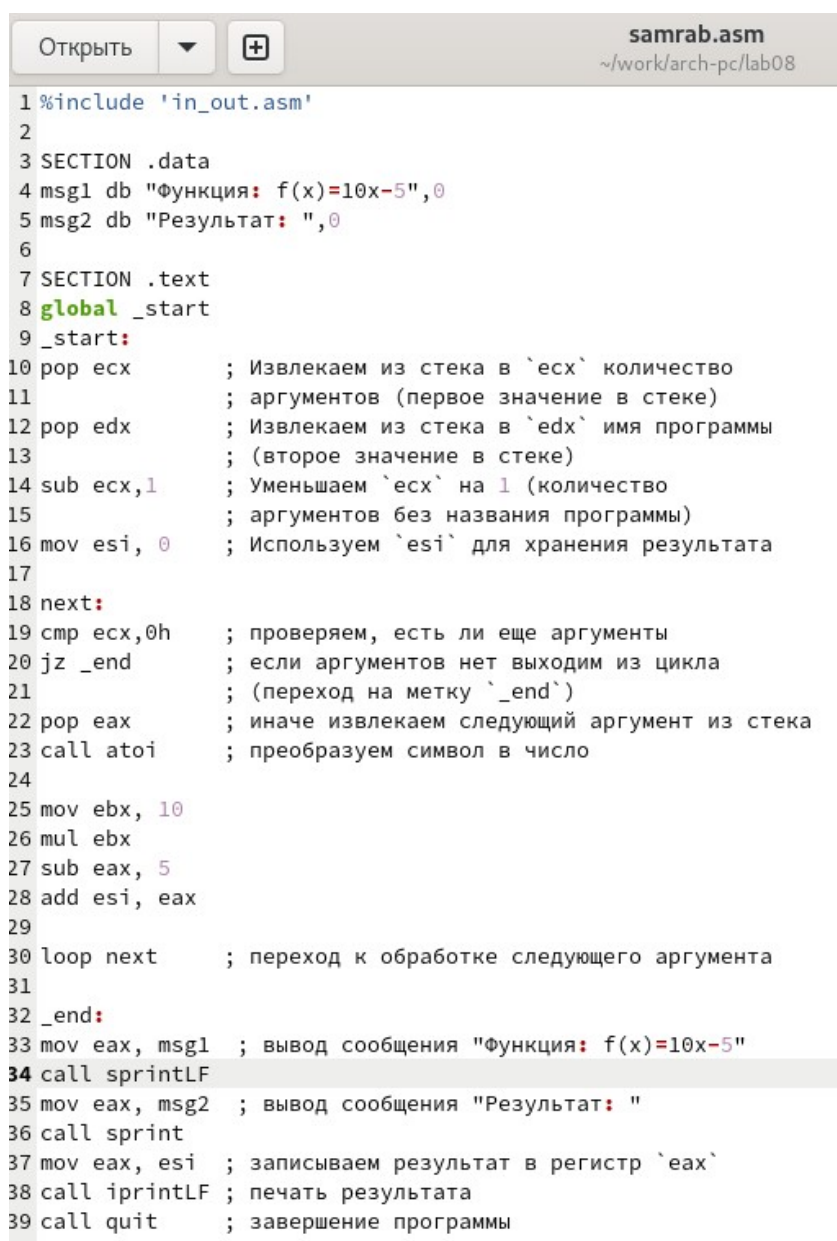


```
uapolyakova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
uapolyakova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
uapolyakova@fedora:~/work/arch-pc/lab08$ ./lab8-3 2 3 5
Результат: 30
uapolyakova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.12: Запуск измененного lab8-3.asm

## **3 Результаты выполнения заданий для самостоятельной работы**

1. Пишем программу, которая находит сумму значений функции  $f(x)$ , где в  $x$  подставляются значения из аргументов командной строки. В предыдущих лабораторных был Вариант 3, поэтому сейчас также был выбран Вариант 3  $f(x) = 10x - 5$  (Рис. 13).



```
Открыть ▼ + samrab.asm
~/work/arch-pc/lab08

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db "Функция: f(x)=10x-5",0
5 msg2 db "Результат: ",0
6
7 SECTION .text
8 global _start
9 _start:
10 pop ecx      ; Извлекаем из стека в `ecx` количество
11              ; аргументов (первое значение в стеке)
12 pop edx      ; Извлекаем из стека в `edx` имя программы
13              ; (второе значение в стеке)
14 sub ecx,1    ; Уменьшаем `ecx` на 1 (количество
15              ; аргументов без названия программы)
16 mov esi, 0   ; Используем `esi` для хранения результата
17
18 next:
19 cmp ecx,0h   ; проверяем, есть ли еще аргументы
20 jz _end      ; если аргументов нет выходим из цикла
21              ; (переход на метку `_end`)
22 pop eax      ; иначе извлекаем следующий аргумент из стека
23 call atoi    ; преобразуем символ в число
24
25 mov ebx, 10
26 mul ebx
27 sub eax, 5
28 add esi, eax
29
30 loop next    ; переход к обработке следующего аргумента
31
32 _end:
33 mov eax, msg1 ; вывод сообщения "Функция: f(x)=10x-5"
34 call sprintf
35 mov eax, msg2 ; вывод сообщения "Результат: "
36 call sprintf
37 mov eax, esi  ; записываем результат в регистр `eax`
38 call iprintf  ; печать результата
39 call quit     ; завершение программы
```

Рис. 3.1: Файл samrab.asm с выполненной программой

2. Создаем исполняемый файл и запускаем его, указав несколько наборов иксов (Рис. 14).

```
uapolyakova@fedora:~/work/arch-pc/lab08$ nasm -f elf samrab.asm
uapolyakova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o samrab samrab.o
uapolyakova@fedora:~/work/arch-pc/lab08$ ./samrab 1 2 3 4
Функция:  $f(x)=10x-5$ 
Результат: 80
uapolyakova@fedora:~/work/arch-pc/lab08$ ./samrab 1 1 1 1
Функция:  $f(x)=10x-5$ 
Результат: 20
uapolyakova@fedora:~/work/arch-pc/lab08$ ./samrab 5 10 15 20
Функция:  $f(x)=10x-5$ 
Результат: 480
uapolyakova@fedora:~/work/arch-pc/lab08$ ./samrab 5
Функция:  $f(x)=10x-5$ 
Результат: 45
uapolyakova@fedora:~/work/arch-pc/lab08$ ./samrab 20 10
Функция:  $f(x)=10x-5$ 
Результат: 290
uapolyakova@fedora:~/work/arch-pc/lab08$
```

Рис. 3.2: Запуск samrab.asm

## **4 Вывод**

Были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.