

Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Луцкая Алиса Витальевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	12
4.3	Задание для самостоятельной работы	15
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Создание каталога и файла, перемещение между деррикториями .	8
4.2	Ввод программы из листинга	9
4.3	Запуск программы	9
4.4	Изменение программы	10
4.5	Запуск измененной программы	11
4.6	Добавление push и pop в цикл программы	11
4.7	Запуск измененной программы	12
4.8	Ввод программы из листинга	12
4.9	Запуск второй программы	13
4.10	Ввод программы из третьего листинга	13
4.11	Запуск третьей программы	14
4.12	Изменение третьей программы	14
4.13	Запуск измененной третьей программы	15
4.14	Написание программы для самостоятельной работы	16
4.15	Запуск измененной третьей программы	17

Список таблиц

1 Цель работы

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

2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Задания для самостоятельной работы

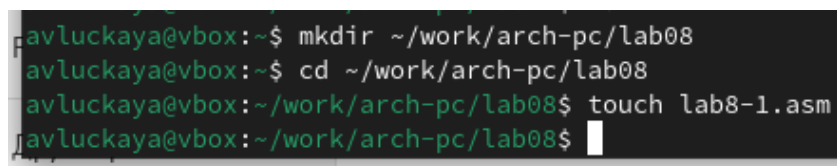
3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

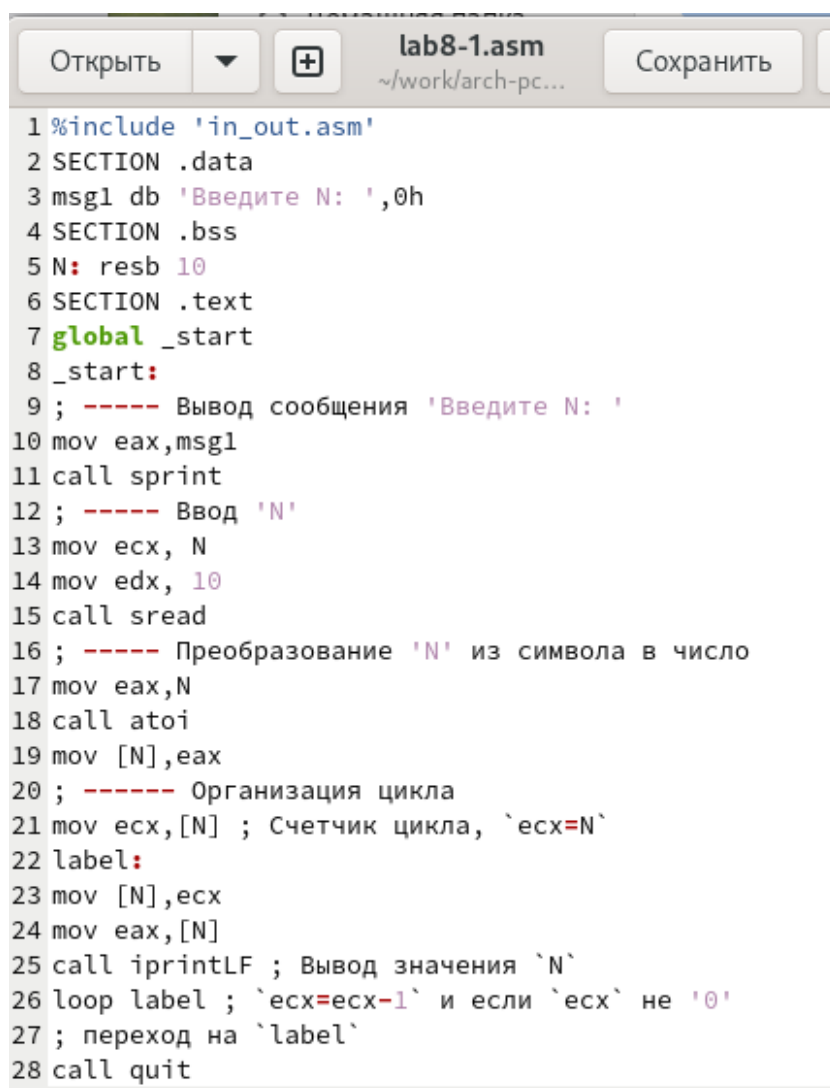
Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm (рис. -fig. 4.1).



```
avluckaya@vbox:~$ mkdir ~/work/arch-pc/lab08
avluckaya@vbox:~$ cd ~/work/arch-pc/lab08
avluckaya@vbox:~/work/arch-pc/lab08$ touch lab8-1.asm
avluckaya@vbox:~/work/arch-pc/lab08$
```

Рис. 4.1: Создание каталога и файла, перемещение между деррикториями

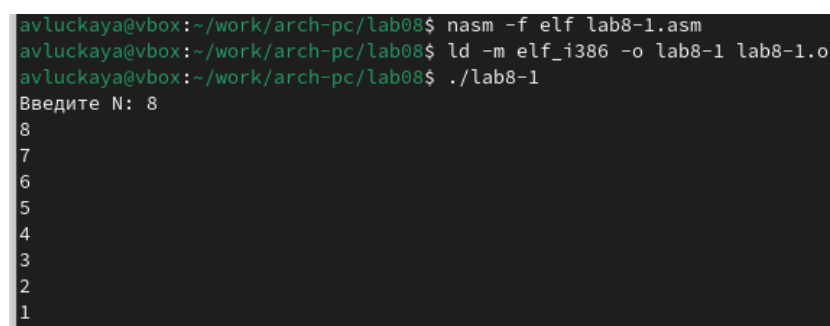
Ввожу в созданный файл программу из листинга. (рис. -fig. 4.2).



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

Рис. 4.2: Ввод программы из листинга

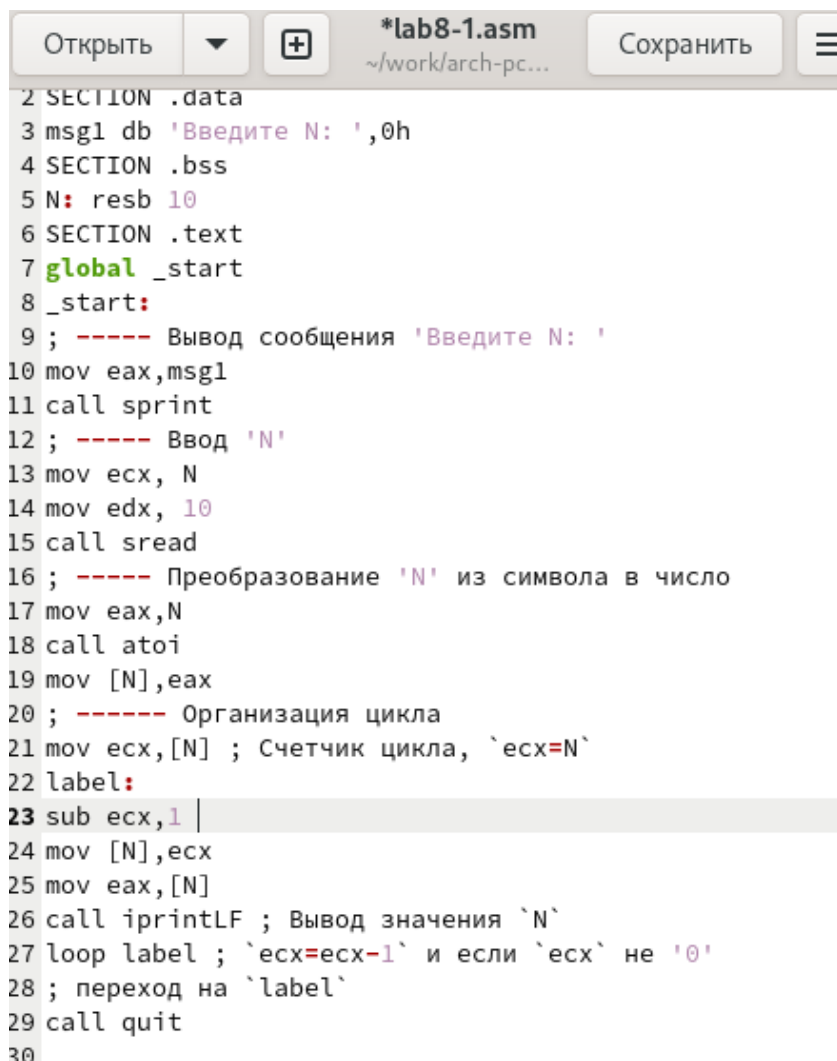
Создаю исполняемый файл и проверяю его работу (рис. -fig. 4.3).



```
avluckaya@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
avluckaya@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
avluckaya@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
8
7
6
5
4
3
2
1
```

Рис. 4.3: Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра `ecx` (рис. -fig. 4.4).



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

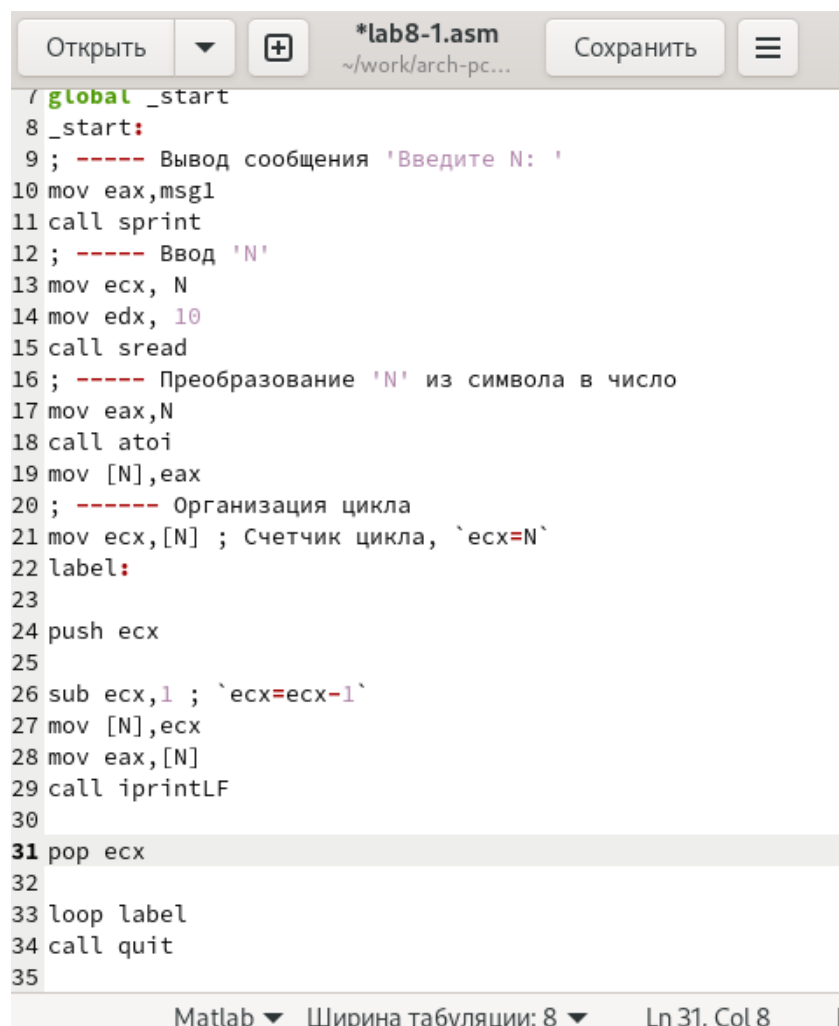
Рис. 4.4: Изменение программы

Из-за того, что теперь регистр `ecx` на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. -fig. 4.5).

```
avluckaya@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
avluckaya@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
avluckaya@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
5
3
1
avluckaya@vbox:~/work/arch-pc/lab08$
```

Рис. 4.5: Запуск измененной программы

Вношу изменения в текст программы добавив команды push и pop для сохранения значения счетчика цикла loop (рис. -fig. 4.6).



```
Открыть *lab8-1.asm Сохранить
~/work/arch-pc...
1 global _start
2 _start:
3 ; ----- Вывод сообщения 'Введите N: '
4 mov eax,msg1
5 call sprint
6 ; ----- Ввод 'N'
7 mov ecx, N
8 mov edx, 10
9 call sread
10 ; ----- Преобразование 'N' из символа в число
11 mov eax,N
12 call atoi
13 mov [N],eax
14 ; ----- Организация цикла
15 mov ecx,[N] ; Счетчик цикла, `ecx=N`
16 label:
17
18 push ecx
19
20 sub ecx,1 ; `ecx=ecx-1`
21 mov [N],ecx
22 mov eax,[N]
23 call iprintLF
24
25 pop ecx
26
27 loop label
28 call quit
29
```

Рис. 4.6: Добавление push и pop в цикл программы

Количество итераций совпадает с введенным N, но произошло смещение вы-

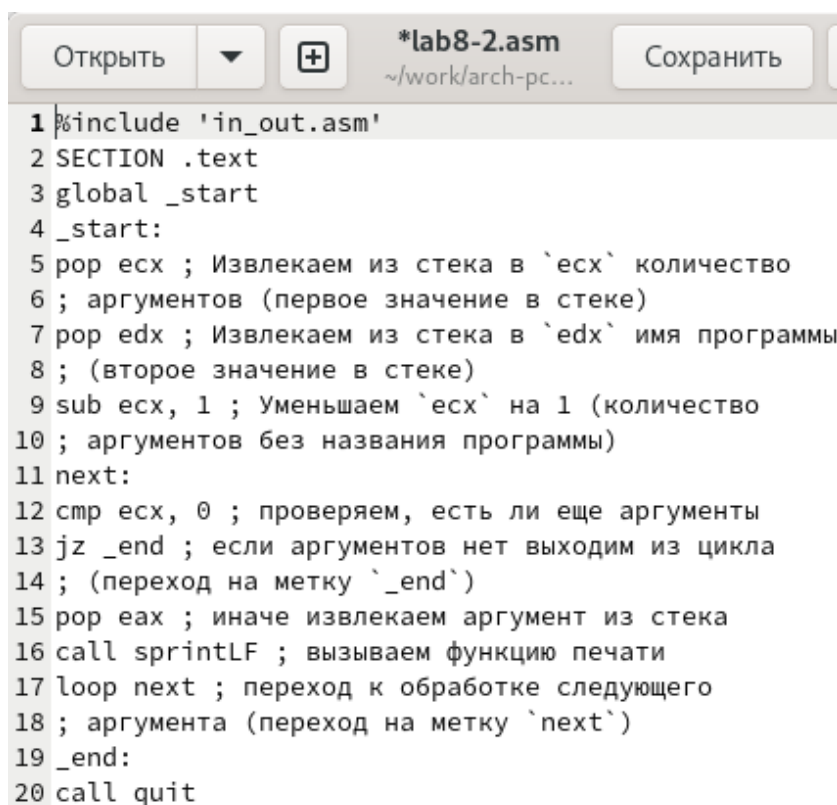
водимых чисел на -1 (рис. -fig. 4.7).

```
avluckaya@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
avluckaya@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
avluckaya@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
6
5
4
3
2
1
0
```

Рис. 4.7: Запуск измененной программы

4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы из листинга 8.2 (рис. -fig. 4.8).



```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов без названия программы)
11 next:
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет выходим из цикла
14 ; (переход на метку `_end`)
15 pop eax ; иначе извлекаем аргумент из стека
16 call sprintf ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit
```

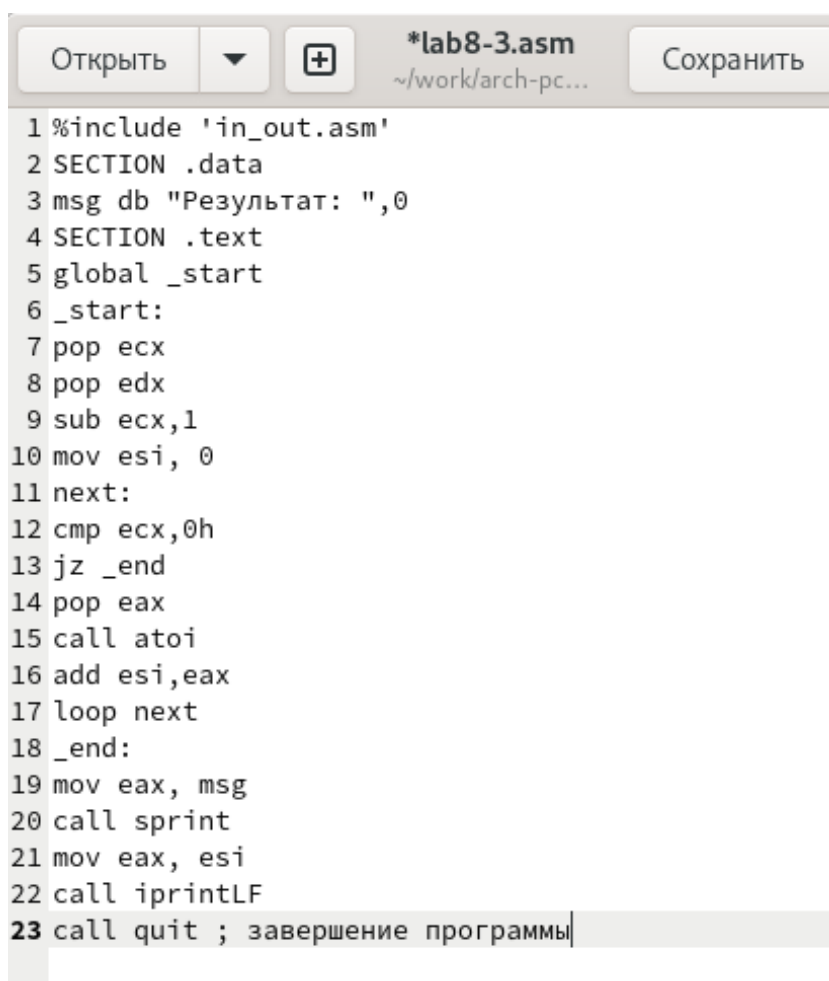
Рис. 4.8: Ввод программы из листинга

Создаю исполняемый файл и запускаю его, указав аргументы: аргумент1 аргумент 2 'аргумент 3' (рис. -fig. 4.9).

```
avluckaya@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
avluckaya@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
avluckaya@vbox:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рис. 4.9: Запуск второй программы

Создаю файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы из листинга 8.3. (рис. -fig. 4.10).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx
8 pop edx
9 sub ecx,1
10 mov esi, 0
11 next:
12 cmp ecx,0h
13 jz _end
14 pop eax
15 call atoi
16 add esi,eax
17 loop next
18 _end:
19 mov eax, msg
20 call sprint
21 mov eax, esi
22 call iprintLF
23 call quit ; завершение программы
```

Рис. 4.10: Ввод программы из третьего листинга

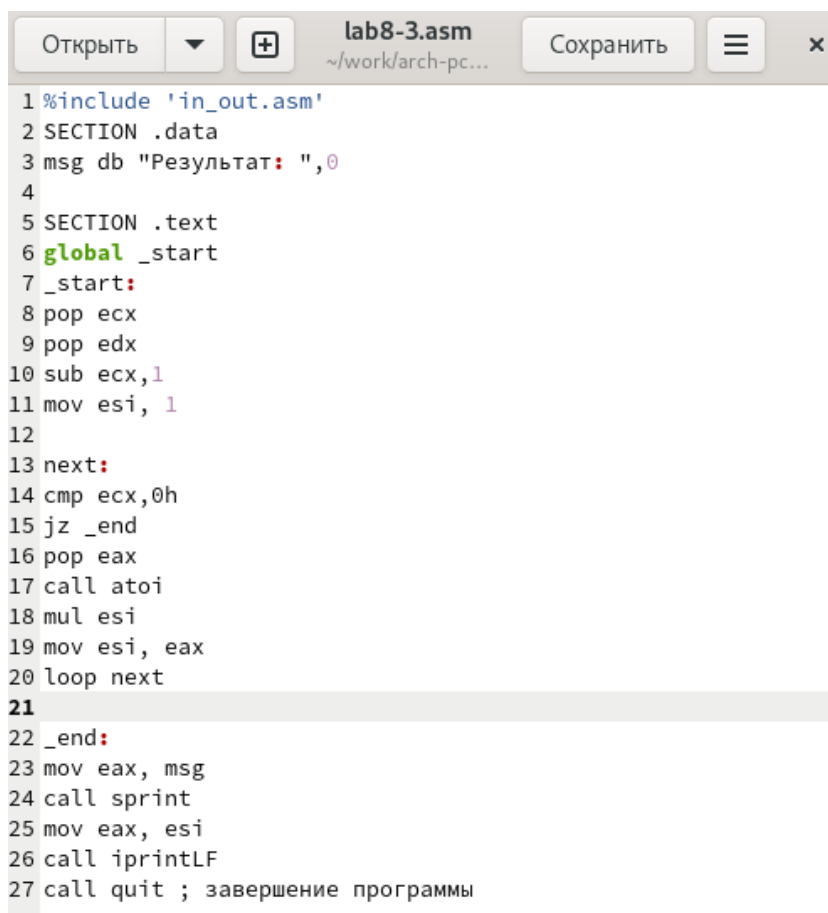
Создаю исполняемый файл и запускаю его, указав аргументы 12 13 7 10 5.(рис.

-fig. 4.11).

```
аргумент 3
avluckaya@vbox: ~/work/arch-pc/lab08$ touch lab8-3.asm
avluckaya@vbox: ~/work/arch-pc/lab08$ gedit lab8-3.asm
avluckaya@vbox: ~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
avluckaya@vbox: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
avluckaya@vbox: ~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
avluckaya@vbox: ~/work/arch-pc/lab08$
```

Рис. 4.11: Запуск третьей программы

Изменяю программу так, чтобы указанные аргументы она умножала, а не складывала (рис. -fig. 4.12).



```
Открыть  lab8-3.asm  Сохранить
~/work/arch-pc...
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4
5 SECTION .text
6 global _start
7 _start:
8 pop ecx
9 pop edx
10 sub ecx,1
11 mov esi, 1
12
13 next:
14 cmp ecx,0h
15 jz _end
16 pop eax
17 call atoi
18 mul esi
19 mov esi, eax
20 loop next
21
22 _end:
23 mov eax, msg
24 call sprint
25 mov eax, esi
26 call iprintLF
27 call quit ; завершение программы
```

Рис. 4.12: Изменение третьей программы

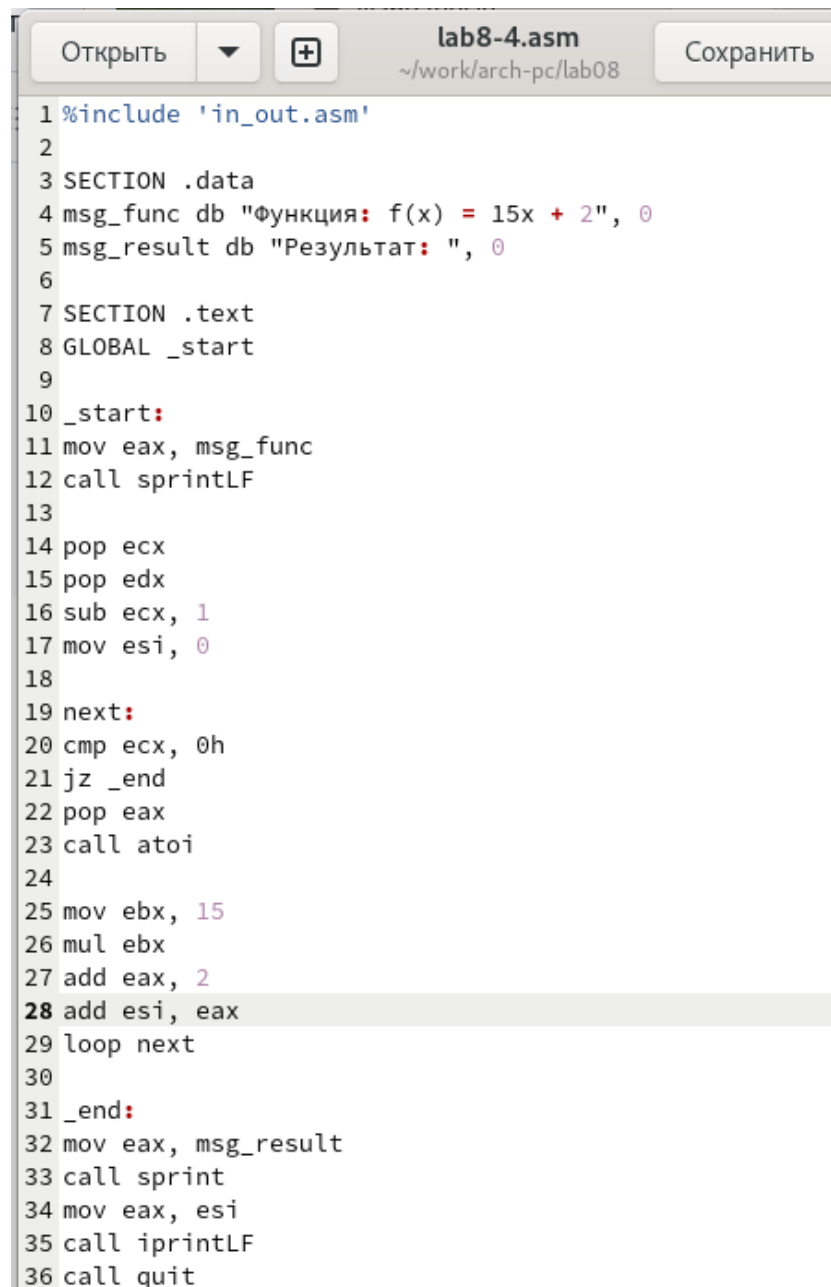
Создаю исполняемый файл и запускаю его. (рис. -fig. 4.13). Программа действительно перемножает вводимые аргументы.

```
Результат: 47
avluckaya@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
avluckaya@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
avluckaya@vbox:~/work/arch-pc/lab08$ ./lab8-3 12 13 7
Результат: 1092
```

Рис. 4.13: Запуск измененной третьей программы

4.3 Задание для самостоятельной работы

Пишу программу, которая будет находить сумма значений для функции $f(x) = 15x+2$, которая совпадает с моим 11 вариантом (рис. -fig. 4.14).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg_func db "Функция: f(x) = 15x + 2", 0
5 msg_result db "Результат: ", 0
6
7 SECTION .text
8 GLOBAL _start
9
10 _start:
11 mov eax, msg_func
12 call sprintLF
13
14 pop ecx
15 pop edx
16 sub ecx, 1
17 mov esi, 0
18
19 next:
20 cmp ecx, 0h
21 jz _end
22 pop eax
23 call atoi
24
25 mov ebx, 15
26 mul ebx
27 add eax, 2
28 add esi, eax
29 loop next
30
31 _end:
32 mov eax, msg_result
33 call sprint
34 mov eax, esi
35 call iprintLF
36 call quit
```

Рис. 4.14: Написание программы для самостоятельной работы

Создаю исполняемый файл и запускаю его. Проверяю работу программы с разными аргументами (рис. -fig. 4.15). Программа работает корректно.


```

avluckaya@vbox:~/work/arch-pc/lab08$ touch lab8-4.asm
avluckaya@vbox:~/work/arch-pc/lab08$ gedit lab8-4.asm
avluckaya@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
avluckaya@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
avluckaya@vbox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Функция: f(x) = 15x + 2
Результат: 96
avluckaya@vbox:~/work/arch-pc/lab08$ ./lab8-4 4 3 2
Функция: f(x) = 15x + 2
Результат: 141

```

Рис. 4.15: Запуск измененной третьей программы

Код программы:

```

#include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 15x + 2", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintf

pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end

```

```
pop eax
call atoi

mov ebx, 15
mul ebx
add eax, 2
add esi, eax
loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

Список литературы

1. <https://esystem.rudn.ru/course/view.php?id=112>