

Лабораторная работа №9

Программирование цикла. Обработка аргументов командной строки

Медникова Екатерина Михайловна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Самостоятельная работа	14
5	Выводы	17

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Ввод текста программы из листинга 9.1	8
3.3	Создание файла и проверка его работы	8
3.4	Изменение текста программы	9
3.5	Бесконечный цикл	9
3.6	Внесение изменений в программу	9
3.7	Создание файла и проверка его работы	10
3.8	Создание файла	10
3.9	Ввод текста из листинга 9.2	10
3.10	Запуск программы	11
3.11	Создание файла	11
3.12	Ввод программы из листинга 9.3	12
3.13	Запуск программы	12
3.14	Изменение текста программы	13
3.15	Создание файла и проверка его работы	13
4.1	Создание файла для самостоятельной работы	14
4.2	Текст программы	15
4.3	Текст программы	16
4.4	Создание файла и проверка его работы	16

Список таблиц

1 Цель работы

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

2 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.

Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается.

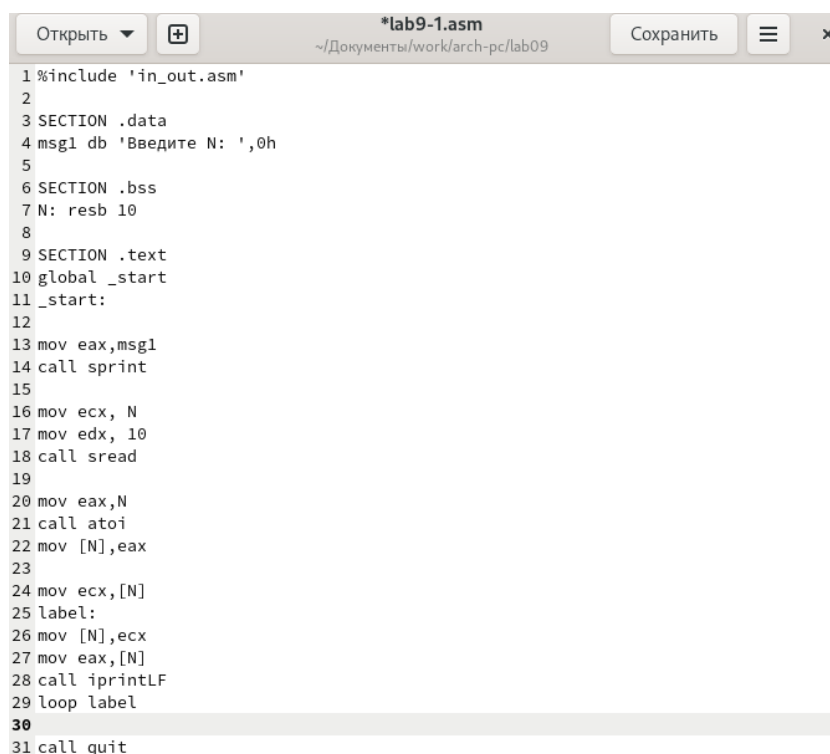
3 Выполнение лабораторной работы

1. Создала каталог для программ лабораторной работы No 9, перешла в него и создала файл lab9-1.asm.

```
[emmednikova@fedora ~]$ mc  
[emmednikova@fedora arch-pc]$ mkdir lab09  
[emmednikova@fedora arch-pc]$ cd lab09/  
[emmednikova@fedora lab09]$ touch lab9-1.asm  
[emmednikova@fedora lab09]$
```

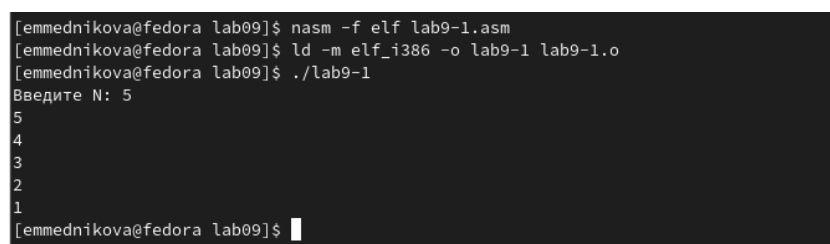
Рис. 3.1: Создание каталога и файла

2. Ввела в файл lab9-1.asm текст программы из листинга 9.1. Создала исполняемый файл и проверила его работу.



```
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
13 mov eax,msg1
14 call sprint
15
16 mov ecx, N
17 mov edx, 10
18 call sread
19
20 mov eax,N
21 call atoi
22 mov [N],eax
23
24 mov ecx,[N]
25 label:
26 mov [N],ecx
27 mov eax,[N]
28 call iprintLF
29 loop label
30
31 call quit
```

Рис. 3.2: Ввод текста программы из листинга 9.1



```
[emmednikova@fedora lab09]$ nasm -f elf lab9-1.asm
[emmednikova@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[emmednikova@fedora lab09]$ ./lab9-1
Введите N: 5
5
4
3
2
1
[emmednikova@fedora lab09]$
```

Рис. 3.3: Создание файла и проверка его работы

3. Изменила текст программы, добавив изменение значения регистра `ecx` в цикле. При попытке создать исполняемый файл и проверить его работу, цикл получился бесконечным.


```

24 mov ecx,[N]
25 label:
26 sub ecx,1
27 mov [N],ecx
28 mov eax,[N]
29 call iprintLF
30 loop label

```

Рис. 3.4: Изменение текста программы

```

4294835016
4294835014
4294835012
4294835010
4294835008
4294835006
4294835004
4294835002
4294835000
4294834998
4294834996
4294834994
4294834992
4294834990
4294834988
4294834986
4294834984
4294834982
4294834980
4294834978
4294834976
4294834974
4294834972
4294834970
4294834968
4294834966
4294834964
4294834962
4294834960
4294834958

```

Рис. 3.5: Бесконечный цикл

Внесла изменения в текст программы, добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`.

```

24 mov ecx,[N]
25 label:
26 push ecx
27 sub ecx,1
28 mov [N],ecx
29 mov eax,[N]
30 call iprintLF
31 pop ecx
32 loop label
33 call quit

```

Рис. 3.6: Внесение изменений в программу

Создала исполняемый файл и проверила его работу.

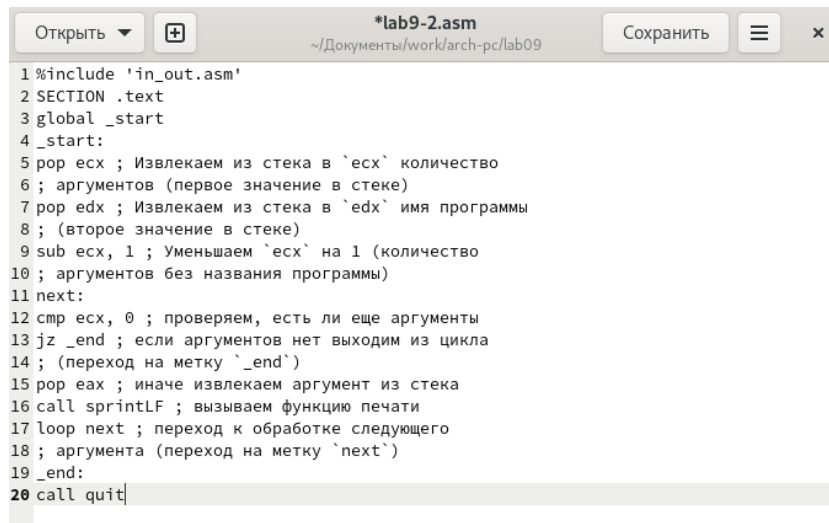
```
[emmednikova@fedora lab09]$ gedit lab9-1.asm
[emmednikova@fedora lab09]$ nasm -f elf lab9-1.asm
[emmednikova@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[emmednikova@fedora lab09]$ ./lab9-1
Введите N: 5
4
3
2
1
0
[emmednikova@fedora lab09]$
```

Рис. 3.7: Создание файла и проверка его работы

4. Создала файл lab9-2.asm в каталоге ~/work/arch-pc/lab09 и ввела в него текст программы из листинга 9.2.

```
[emmednikova@fedora lab09]$ touch lab9-2.asm
```

Рис. 3.8: Создание файла



```
*lab9-2.asm
~/.Документы/work/arch-pc/lab09
Сохранить
Открыть
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 printf ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit
```

Рис. 3.9: Ввод текста из листинга 9.2

Создала исполняемый файл и запустила его, указав аргументы. Программа вывела все три аргумента, но в разном порядке.

```

[emmednikova@fedora lab09]$ nasm -f elf lab9-2.asm
[emmednikova@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[emmednikova@fedora lab09]$ ./lab9-2
[emmednikova@fedora lab09]$ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[emmednikova@fedora lab09]$ ./lab9-2 аргумент1 аргумент2 'аргумент 3'
аргумент1
аргумент2
аргумент 3
[emmednikova@fedora lab09]$

```

Рис. 3.10: Запуск программы

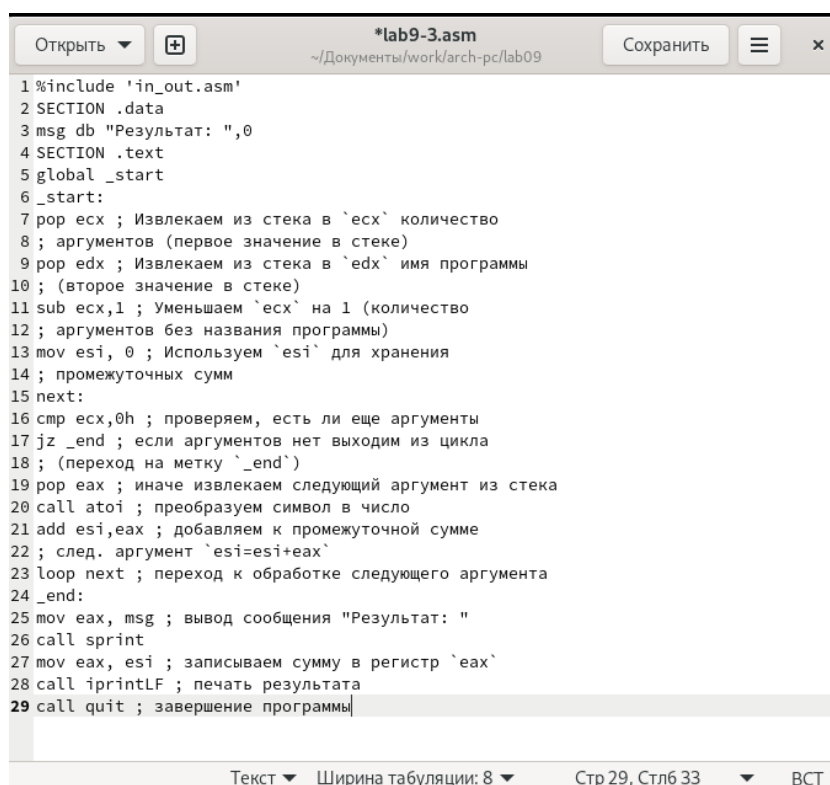
5. Создала файл lab9-3.asm в каталоге ~/work/arch-рс/lab09 и ввела в него текст программы из листинга 9.3.

```

[emmednikova@fedora lab09]$ touch lab9-3.asm
[emmednikova@fedora lab09]$ ls
in_out.asm lab9-1 lab9-1.asm lab9-1.o lab9-2 lab9-2.asm lab9-2.o lab9-3.asm
[emmednikova@fedora lab09]$

```

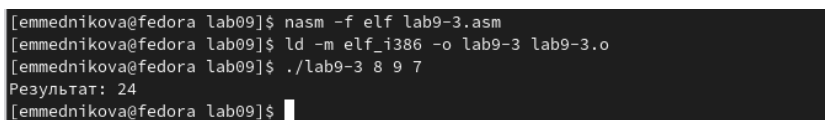
Рис. 3.11: Создание файла



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

Рис. 3.12: Ввод программы из листинга 9.3

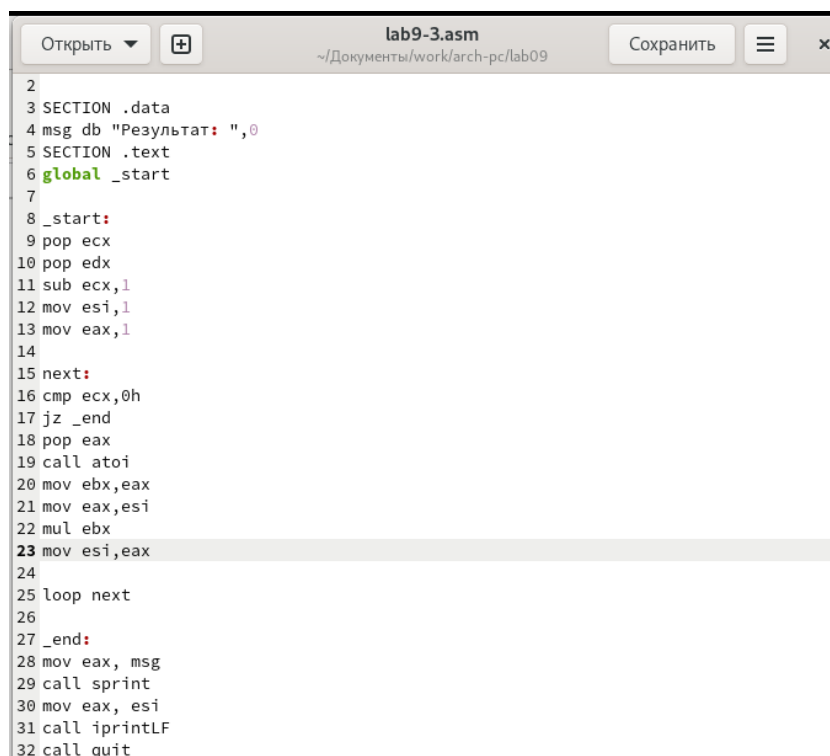
Создала исполняемый файл и запустила его, указав аргументы.



```
[emmednikova@fedora lab09]$ nasm -f elf lab9-3.asm
[emmednikova@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[emmednikova@fedora lab09]$ ./lab9-3 8 9 7
Результат: 24
[emmednikova@fedora lab09]$
```

Рис. 3.13: Запуск программы

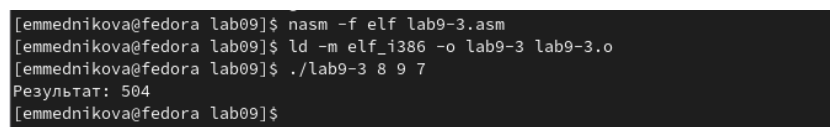
6. Изменила текст программы из листинга 9.3 для вычисления произведения аргументов командной строки.



```
2
3 SECTION .data
4 msg db "Результат: ",0
5 SECTION .text
6 global _start
7
8 _start:
9 pop ecx
10 pop edx
11 sub ecx,1
12 mov esi,1
13 mov eax,1
14
15 next:
16 cmp ecx,0h
17 jz _end
18 pop eax
19 call atoi
20 mov ebx,eax
21 mov eax,esi
22 mul ebx
23 mov esi,eax
24
25 loop next
26
27 _end:
28 mov eax, msg
29 call sprint
30 mov eax, esi
31 call iprintLF
32 call quit
```

Рис. 3.14: Изменение текста программы

Создала исполняемый файл и проверила его работу.



```
[emmednikova@fedora lab09]$ nasm -f elf lab9-3.asm
[emmednikova@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[emmednikova@fedora lab09]$ ./lab9-3 8 9 7
Результат: 504
[emmednikova@fedora lab09]$
```

Рис. 3.15: Создание файла и проверка его работы

4 Самостоятельная работа

1. Создала файл для самостоятельной работы.

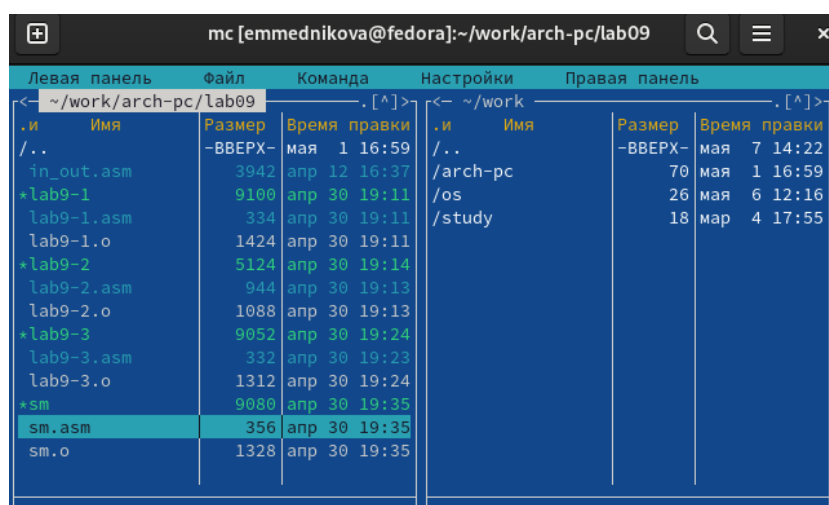



Рис. 4.1: Создание файла для самостоятельной работы

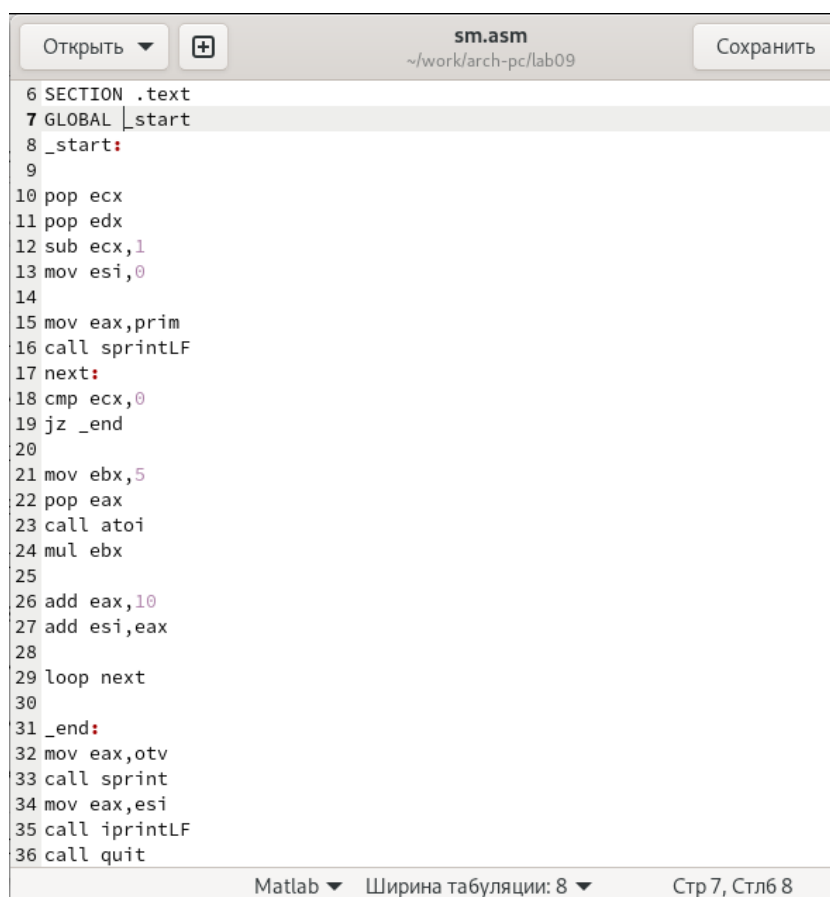
2. Написала в созданный файл текст программы.

Открыть  sm.asm ~/work/arch-pc/lab09 Сохранить

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

Matlab ▾ Ширина табуляции: 8 ▾ Стр 7, Стлб 8

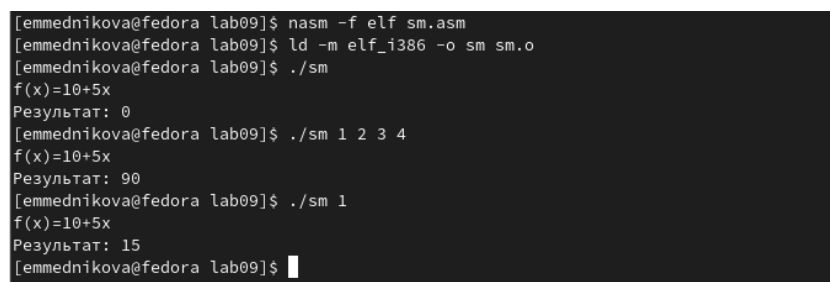
Рис. 4.2: Текст программы



```
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 pop ecx
11 pop edx
12 sub ecx,1
13 mov esi,0
14
15 mov eax,prim
16 call sprintLF
17 next:
18 cmp ecx,0
19 jz _end
20
21 mov ebx,5
22 pop eax
23 call atoi
24 mul ebx
25
26 add eax,10
27 add esi,eax
28
29 loop next
30
31 _end:
32 mov eax,otv
33 call sprint
34 mov eax,esi
35 call iprintLF
36 call quit
```

Рис. 4.3: Текст программы

3. Создала исполняемый файл и проверила его работу.



```
[emmednikova@fedora lab09]$ nasm -f elf sm.asm
[emmednikova@fedora lab09]$ ld -m elf_i386 -o sm sm.o
[emmednikova@fedora lab09]$ ./sm
f(x)=10+5x
Результат: 0
[emmednikova@fedora lab09]$ ./sm 1 2 3 4
f(x)=10+5x
Результат: 90
[emmednikova@fedora lab09]$ ./sm 1
f(x)=10+5x
Результат: 15
[emmednikova@fedora lab09]$
```

Рис. 4.4: Создание файла и проверка его работы

5 Выводы

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