

Лабораторная работа №9.

Программирование цикла. Обработка аргументов командной строки.

Боровиков Даниил Александрович

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Самостоятельная работа	13
4	Выводы	17

Список иллюстраций

2.1	Создание файла lab9-1.asm в соответствующем каталоге	5
2.2	Текст программы из листинга 9.1.	5
2.3	Запуск исполняемого файла lab9-1.asm	6
2.4	Текст измененной программы	6
2.5	Запуск исправленного исполняемого файла lab8-1.asm	7
2.6	Листинг программы с сохранением значения счетсика в цикле . .	8
2.7	Запуск измененного исполняемого файла lab9-1.asm	8
2.8	Листинга 9.2	9
2.9	Запуск исполняемого файла lab9-2.asm	9
2.10	Листинг 9.3	10
2.11	Запуск исполняемого файла lab9-3.asm	10
2.12	Текст программы для вычисления произведения командной строки	11
2.13	Запуск исполняемого файла lab9-4.asm	11
3.1	Текст программы	14
3.2	Запуск исполняемого файла sam.asm	14

1 Цель работы

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

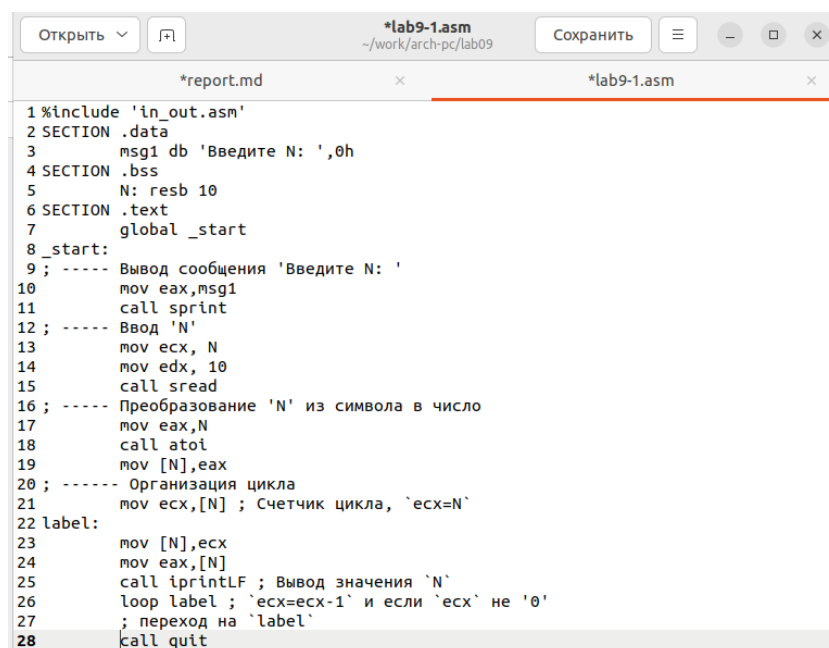
2 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы № 9, перейдем в него и создадим файл lab9-1.asm(рис. 2.1)

```
user@daborovikov:~$ mkdir ~/work/arch-pc/lab09
user@daborovikov:~$ cd ~/work/arch-pc/lab09
user@daborovikov:~/work/arch-pc/lab09$ touch lab9-1.asm
user@daborovikov:~/work/arch-pc/lab09$
```

Рис. 2.1: Создание файла lab9-1.asm в соответствующем каталоге

Введем в файл lab9-1.asm текст программы из листинга 9.1.(рис. 2.2)



```
*lab9-1.asm
~/work/arch-pc/lab09
Сохранить

*report.md x
*lab9-1.asm x

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 iprintlnLF ; Вывод значения `N`
26     loop label ; `ecx=ecx-1` и если `ecx` не `0`
27     ; переход на `label`
28     call quit
```

Рис. 2.2: Текст программы из листинга 9.1.

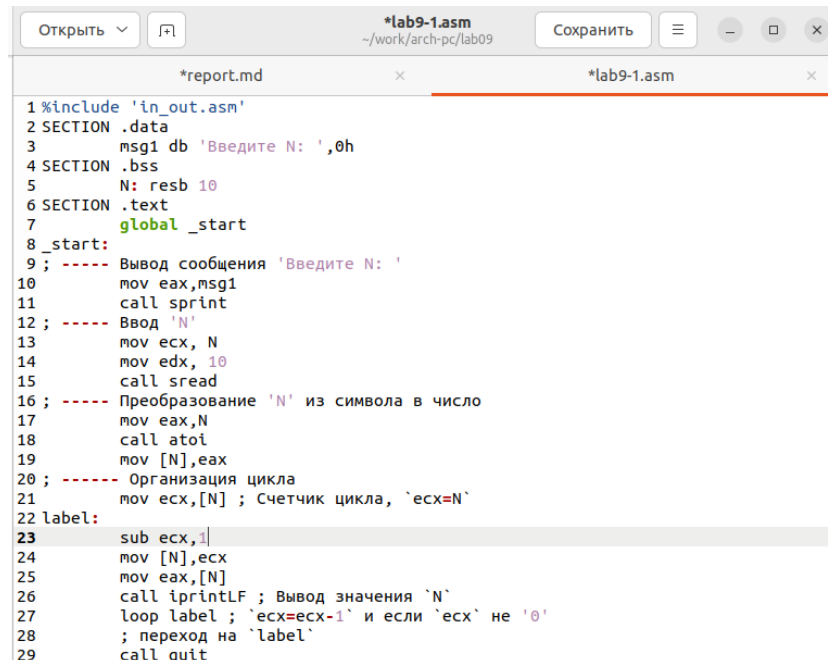
Создадим исполняемый файл и запустим его.(рис. 2.3)

```
user@daborovikov:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
user@daborovikov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
user@daborovikov:~/work/arch-pc/lab09$ ./lab9-1
Введите N: 9
9
8
7
6
5
4
3
2
1
user@daborovikov:~/work/arch-pc/lab09$
```

Рис. 2.3: Запуск исполняемого файла lab9-1.asm

Число проходов цикла соответствует введенному значению N.

Далее изменим текст программы добавив изменение значение регистра ecx в цикле(рис. 2.4)



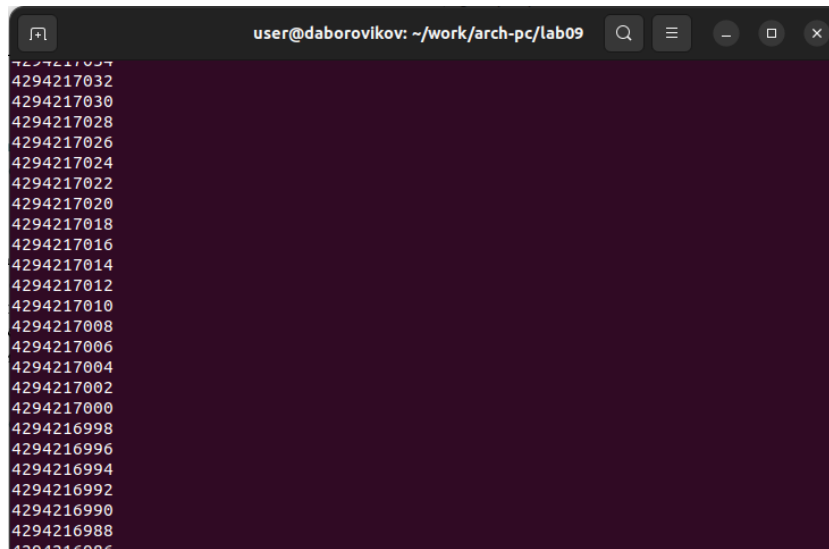
```
*lab9-1.asm
~/work/arch-pc/lab09
Сохранить

*report.md
*lab9-1.asm

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    sub ecx,1
24    mov [N],ecx
25    mov eax,[N]
26    call iprintlnLF ; Вывод значения `N`
27    loop label ; `ecx=ecx-1` и если `ecx` не `0`
28    ; переход на `label`
29    call quit
```

Рис. 2.4: Текст измененной программы

Создадим исполняемый файл исправленного текста программы lab9-1.asm и запустите его.(рис. 2.5)

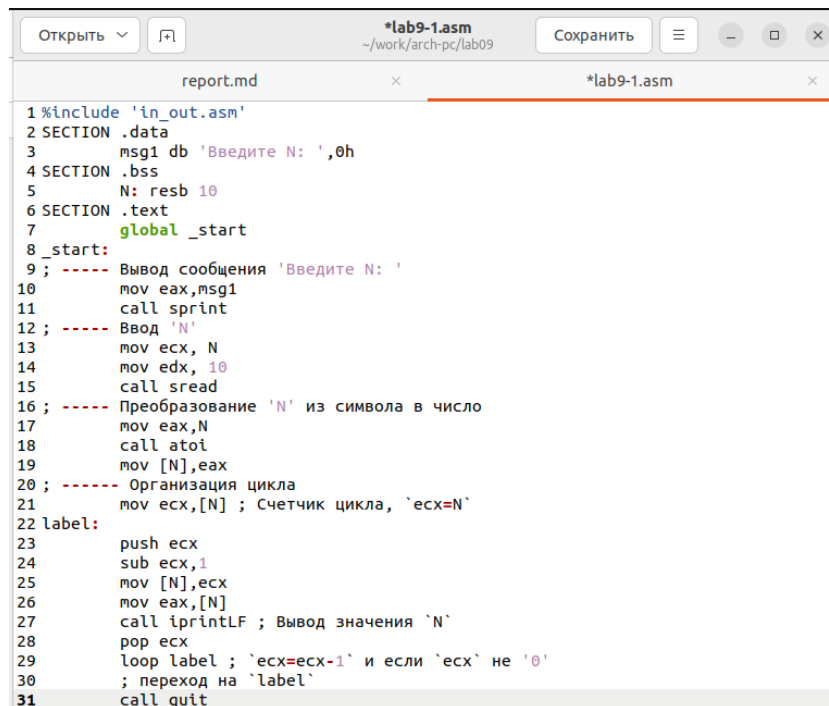


```
user@daborovikov: ~/work/arch-pc/lab09
4294217034
4294217032
4294217030
4294217028
4294217026
4294217024
4294217022
4294217020
4294217018
4294217016
4294217014
4294217012
4294217010
4294217008
4294217006
4294217004
4294217002
4294217000
4294216998
4294216996
4294216994
4294216992
4294216990
4294216988
4294216986
```

Рис. 2.5: Запуск исправленного исполняемого файла lab8-1.asm

Программа выводит некорректные значения из-за использования регистра еsx в теле цикла loop. Значения еsx перезаписываются в еax и выводятся на экран. Число проходов не соответствует значению N, как это было в первоначальной версии программы.

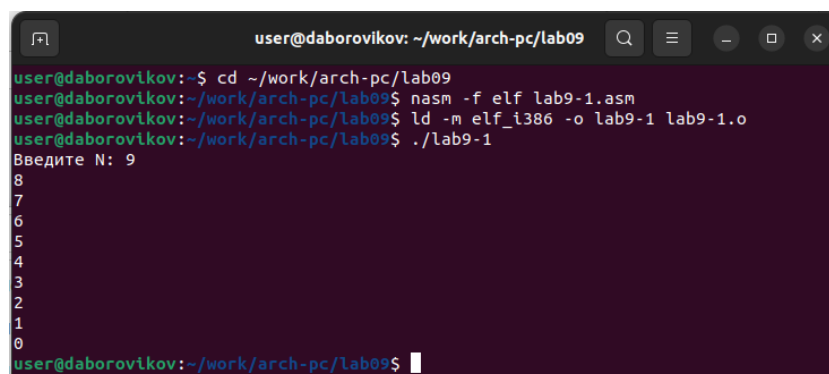
Внесем изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:(рис. 2.6)



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

Рис. 2.6: Листинг программы с сохранением значения счетсика в цикле

Создадим исполняемый файл и запустим его(рис. 2.7)



```
user@daborovikov: ~/work/arch-pc/lab09
user@daborovikov:~/work/arch-pc/lab09$ cd ~/work/arch-pc/lab09
user@daborovikov:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
user@daborovikov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
user@daborovikov:~/work/arch-pc/lab09$ ./lab9-1
Введите N: 9
8
7
6
5
4
3
2
1
0
user@daborovikov:~/work/arch-pc/lab09$
```

Рис. 2.7: Запуск измененного исполняемого файла lab9-1.asm

В данном случае число проходов цикла соответствует значению, введенному с клавиатуры

Создадим файл lab9-2.asm в каталоге ~/work/arch-pc/lab09. Внимательно изучим текст программы из листинга 9.2 и введем в lab9-2.asm.(рис. 2.8)

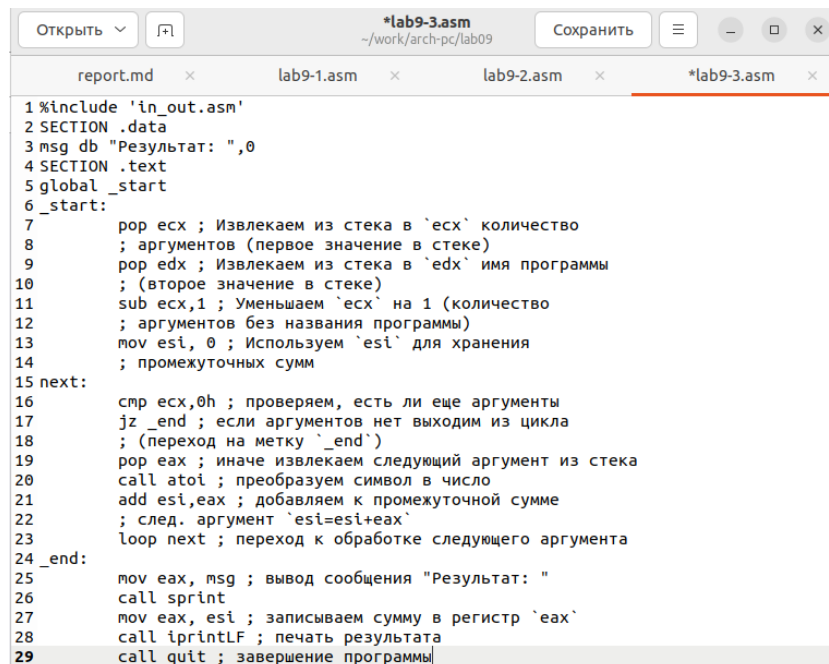
Рис. 2.8: Листинга 9.2

Создадим измененный исполняемый файл и запустим его(рис. 2.9)

Рис. 2.9: Запуск исполняемого файла lab9-2.asm

Программой было обработано 4 аргумента.

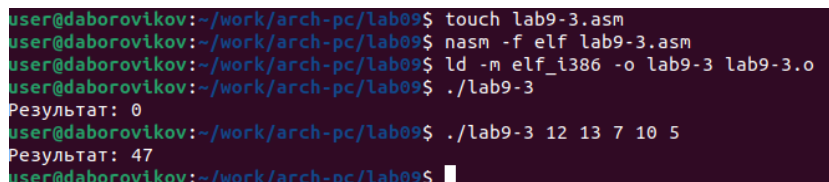
Создадим файл lab9-3.asm в каталоге ~/.work/arch-pc/lab09. Внимательно изучим текст программы из листинга 9.3 и введем в lab9-3.asm.(рис. 2.10)



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7     pop ecx ; 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. 2.10: Листинг 9.3

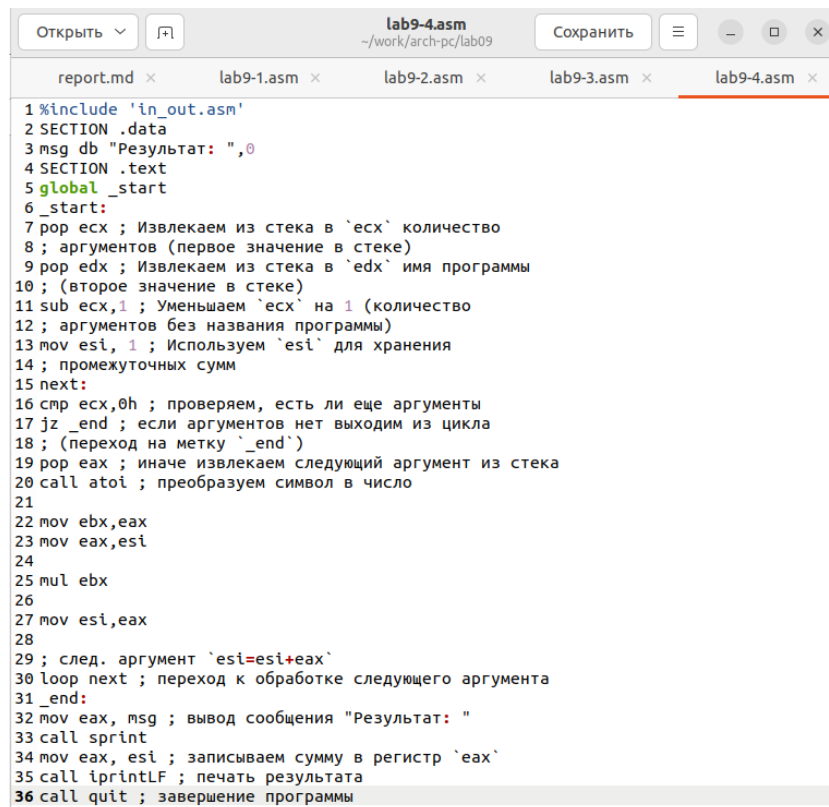
Создадим исполняемый файл и запустим его, указав аргументы(рис. 2.11)



```
user@daborovikov:~/work/arch-pc/lab09$ touch lab9-3.asm
user@daborovikov:~/work/arch-pc/lab09$ nasm -f elf lab9-3.asm
user@daborovikov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
user@daborovikov:~/work/arch-pc/lab09$ ./lab9-3
Результат: 0
user@daborovikov:~/work/arch-pc/lab09$ ./lab9-3 12 13 7 10 5
Результат: 47
user@daborovikov:~/work/arch-pc/lab09$
```

Рис. 2.11: Запуск исполняемого файла lab9-3.asm

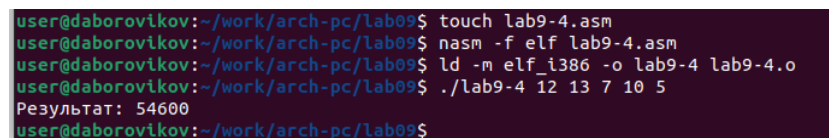
Изменим файл lab9-3.asm для вычисления произведения аргументов командной строки (рис. 2.12)



```
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, 1 ; 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
22 mov ebx,eax
23 mov eax,esi
24
25 mul ebx
26
27 mov esi,eax
28
29 ; след. аргумент `esi=esi+eax`
30 loop next ; переход к обработке следующего аргумента
31 _end:
32 mov eax, msg ; вывод сообщения "Результат: "
33 call sprint
34 mov eax, esi ; записываем сумму в регистр `eax`
35 call iprintf ; печать результата
36 call quit ; завершение программы
```

Рис. 2.12: Текст программы для вычисления произведения командной строки

Создадим исполняем файл и запустим его, указав аргументы(рис. 2.13)



```
user@daborovikov:~/work/arch-pc/lab09$ touch lab9-4.asm
user@daborovikov:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm
user@daborovikov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-4 lab9-4.o
user@daborovikov:~/work/arch-pc/lab09$ ./lab9-4 12 13 7 10 5
Результат: 54600
user@daborovikov:~/work/arch-pc/lab09$
```

Рис. 2.13: Запуск исполняемого файла lab9-4.asm

Листинг программы:

```
%include 'in_out.asm'
SECTION .data
msg db "Результат:",0
SECTION .text
global _start
_start:
```

```

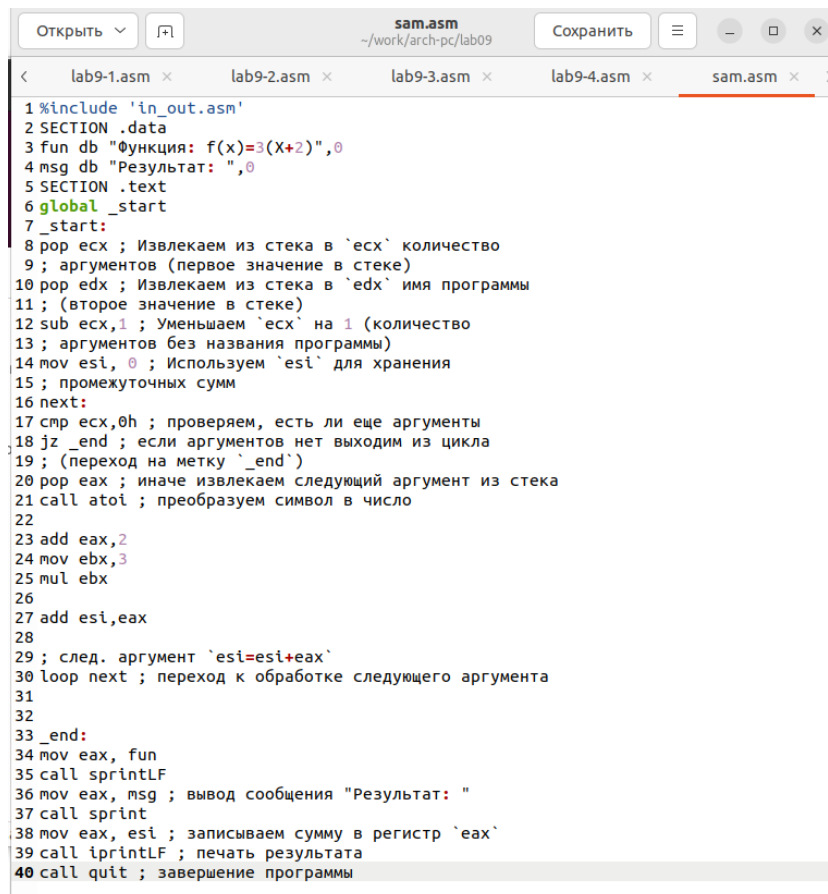
pop ecx ; Извлекаем из стека в ecx количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в edx имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем ecx на 1 (количество
; аргументов без названия программы)
mov esi, 1 ; Используем esi для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку _end)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mov ebx,eax
mov eax,esi
mul ebx
mov esi,eax
; след. аргумент esi=esi+eax
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат:"
call sprint
mov eax, esi ; записываем сумму в регистр eax
call iprintLF ; печать результата
call quit ; завершение программы

```

3 Самостоятельная работа

Мой вариант номер 7

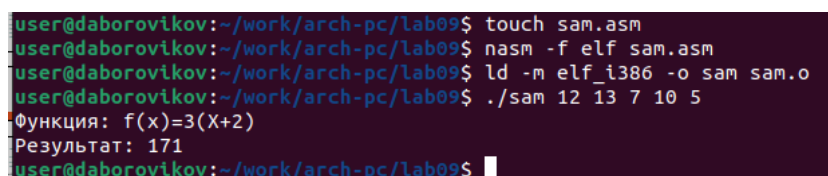
Напишем программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выберем из таблицы 9.1 вариантов заданий в соответствии с вариантом номер 7, полученным при выполнении лабораторной работы № 7. Создадим исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$. (рис. 3.1)



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

Рис. 3.1: Текст программы

Создадим исполняем файл и запустим его, указав аргументы(рис. 3.2)



```
user@daborovikov:~/work/arch-pc/lab09$ touch sam.asm
user@daborovikov:~/work/arch-pc/lab09$ nasm -f elf sam.asm
user@daborovikov:~/work/arch-pc/lab09$ ld -m elf_i386 -o sam sam.o
user@daborovikov:~/work/arch-pc/lab09$ ./sam 12 13 7 10 5
Функция: f(x)=3(X+2)
Результат: 171
user@daborovikov:~/work/arch-pc/lab09$
```

Рис. 3.2: Запуск исполняемого файла sam.asm

Листинг программы:

```
%include 'in_out.asm'
SECTION .data
fun db "Функция: f(x)=3(X+2)",0
msg db "Результат:",0
```

```

SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в ecx количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в edx имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем ecx на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем esi для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку _end)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add eax,2
mov ebx,3
mul ebx
add esi,eax
; след. аргумент esi=esi+eax
loop next ; переход к обработке следующего аргумента
_end:
mov eax, fun
call sprintLF
mov eax, msg ; вывод сообщения "Результат:"
call sprint
mov eax, esi ; записываем сумму в регистр eax

```

call iprintLF ; печать результата

call quit ; завершение программы

4 Выводы

В ходе лабораторной работы мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.

https://github.com/daBorovikov/study_2022-2023_arh-pc-