

Отчёт по лабораторной работе 8

дисциплина: Архитектура компьютера

Амина Усманова

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация циклов в NASM	6
2.2	Самостоятельное задание	17
3	Выводы	20

Список иллюстраций

2.1	Создание каталога	6
2.2	Программа lab8-1.asm	7
2.3	Запуск программы lab8-1.asm	8
2.4	Измененная программа lab8-1.asm	9
2.5	Запуск измененной программы lab8-1.asm	10
2.6	Исправленная программа lab8-1.asm	11
2.7	Запуск исправленной программы lab8-1.asm	12
2.8	Программа lab8-2.asm	13
2.9	Запуск программы lab8-2.asm	14
2.10	Программа lab8-3.asm	15
2.11	Запуск программы lab8-3.asm	15
2.12	Программа lab8-3.asm	16
2.13	Запуск программы lab8-3.asm	17
2.14	Программа task.asm	18
2.15	Запуск программы task.asm	19

Список таблиц

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки..

2 Выполнение лабораторной работы

2.1 Реализация циклов в NASM

Создала каталог для программ лабораторной работы № 8 и файл lab8-1.asm (рис. 2.1).

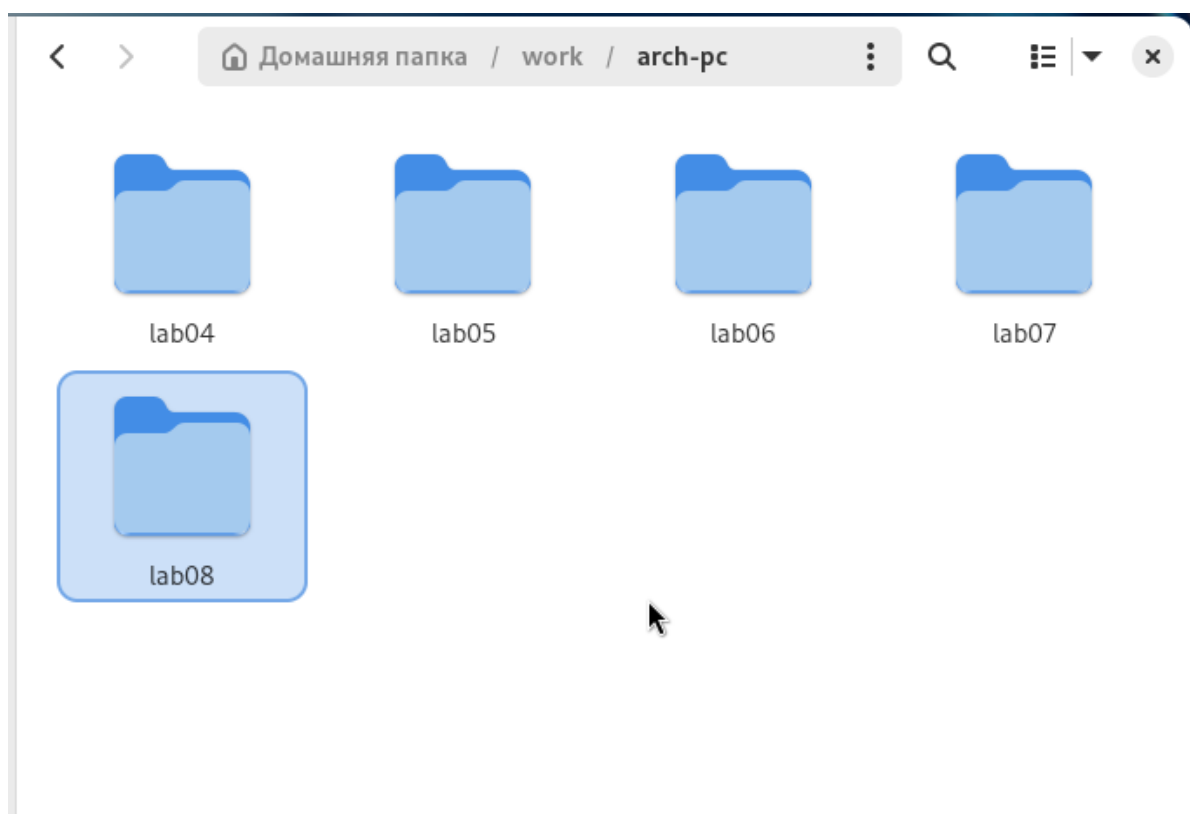
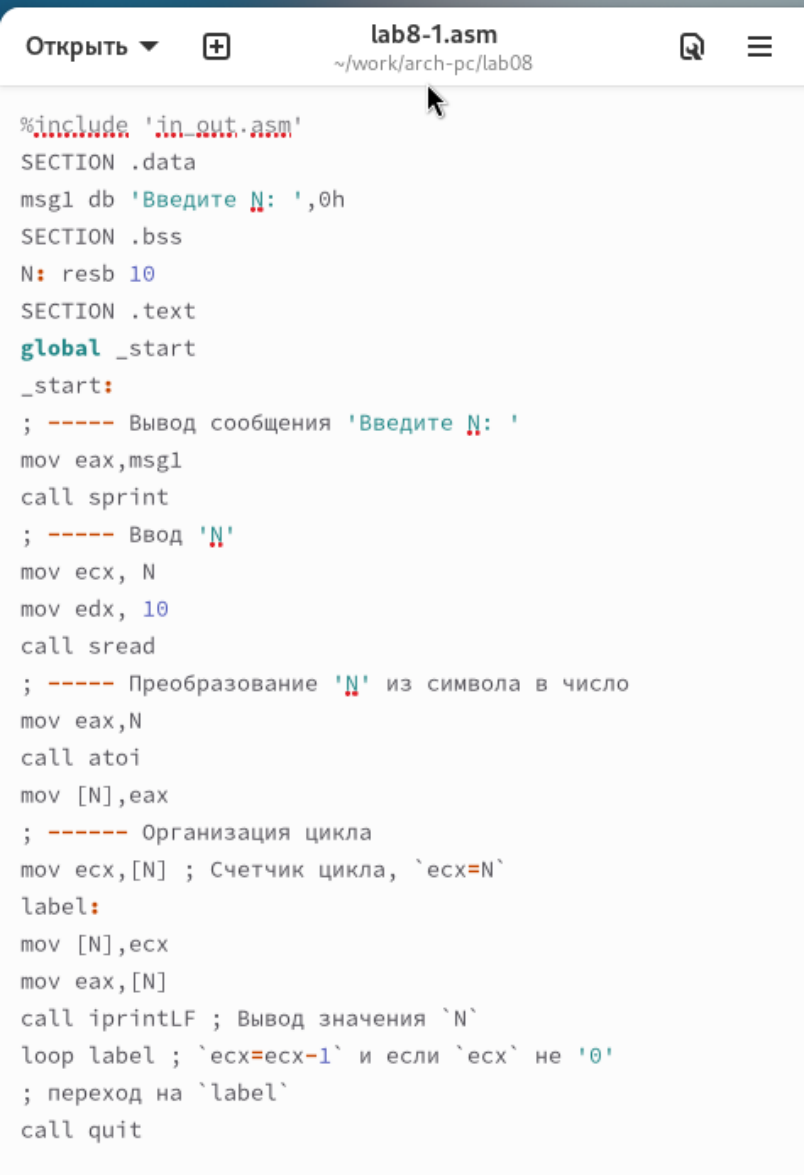


Рис. 2.1: Создание каталога

При реализации циклов в NASM с использованием инструкции `loop` важно помнить, что эта инструкция использует регистр `ecx` в качестве счетчика и на

каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра ехх.

Написала в файл lab8-1.asm текст программы из листинга 8.1. (рис. 2.2). Создала исполняемый файл и проверила его работу. (рис. 2.3).



```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Рис. 2.2: Программа lab8-1.asm

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

Рис. 2.3: Запуск программы lab8-1.asm

Этот пример демонстрирует, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Изменила текст программы, добавив изменение значения регистра `ecx` в цикле. (рис. 2.4). Программа теперь запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N`. (рис. 2.5).

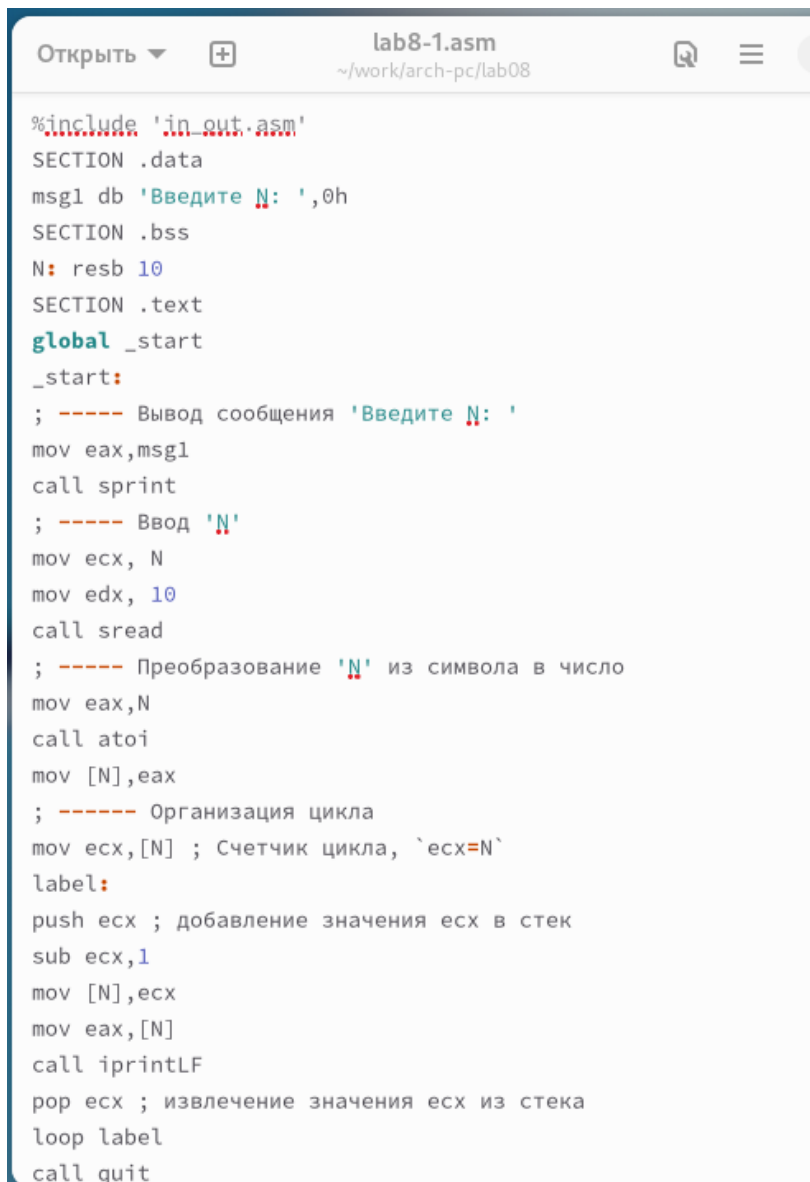

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
; переход на `label`
call quit
```

Рис. 2.4: Измененная программа lab8-1.asm

```
4294949798
4294949796
4294949794
4294949792
4294949790
4294949788
4294949786
4294949784
42949497^C
abusmanova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
abusmanova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.5: Запуск измененной программы lab8-1.asm

Для корректной работы программы с регистром `ecx` в цикле можно использовать стек. Внесла изменения в текст программы, добавив команды `push` и `pop` (для добавления в стек и извлечения из него значений), чтобы сохранить значение счетчика цикла `loop`. (рис. 2.6). Создала исполняемый файл и проверила его работу. (рис. 2.7). Программа теперь выводит числа от $N-1$ до 0, при этом число проходов цикла соответствует значению N .



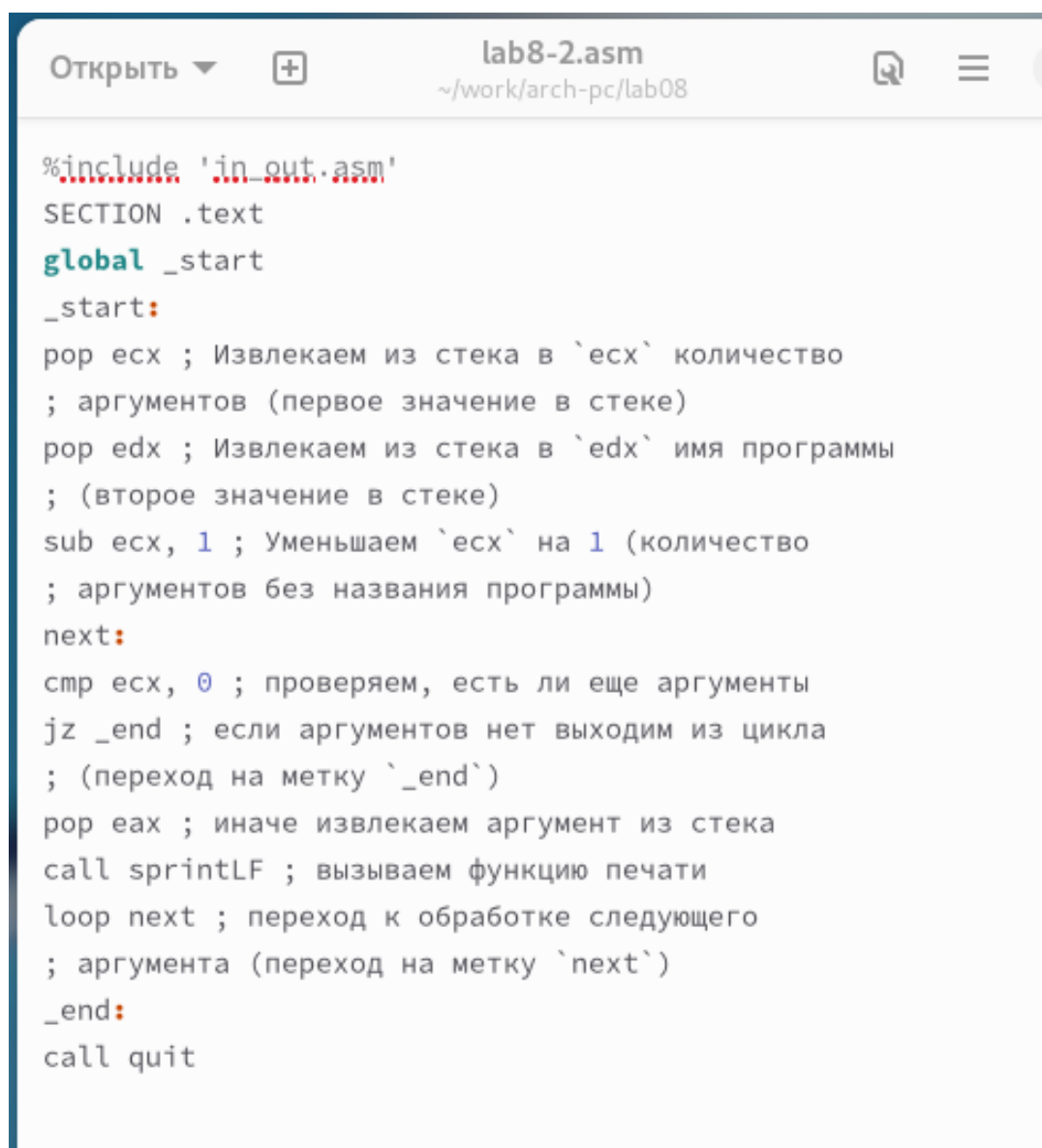
```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label
call quit
```

Рис. 2.6: Исправленная программа lab8-1.asm

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

Рис. 2.7: Запуск исправленной программы lab8-1.asm

Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и написала в него текст программы из листинга 8.2. (рис. 2.8). Компилирую исполняемый файл и запускаю его, указав аргументы. Программа обработала 4 аргумента. Аргументами считаются слова/числа, разделенные пробелом. (рис. 2.9).



```
Открыть ▾ [icon] lab8-2.asm
~/.work/arch-pc/lab08

%include 'in_out.asm'
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)
    next:
    cmp ecx, 0 ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
             ; (переход на метку `_end`)
    pop eax ; иначе извлекаем аргумент из стека
    call sprintLF ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку `next`)
    _end:
    call quit
```

Рис. 2.8: Программа lab8-2.asm

```
abusmanova@fedora:~/work/arch-pc/lab08$  
abusmanova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm  
abusmanova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2  
abusmanova@fedora:~/work/arch-pc/lab08$ ./lab8-2  
abusmanova@fedora:~/work/arch-pc/lab08$ ./lab8-2 1 2 3 4 5  
1  
2  
3  
4  
5  
abusmanova@fedora:~/work/arch-pc/lab08$  
abusmanova@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'  
аргумент1  
аргумент  
2  
аргумент 3  
abusmanova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.9: Запуск программы lab8-2.asm

Рассмотрим еще один пример программы, которая выводит сумму чисел, передаваемых в программу как аргументы. (рис. 2.10) (рис. 2.11).

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

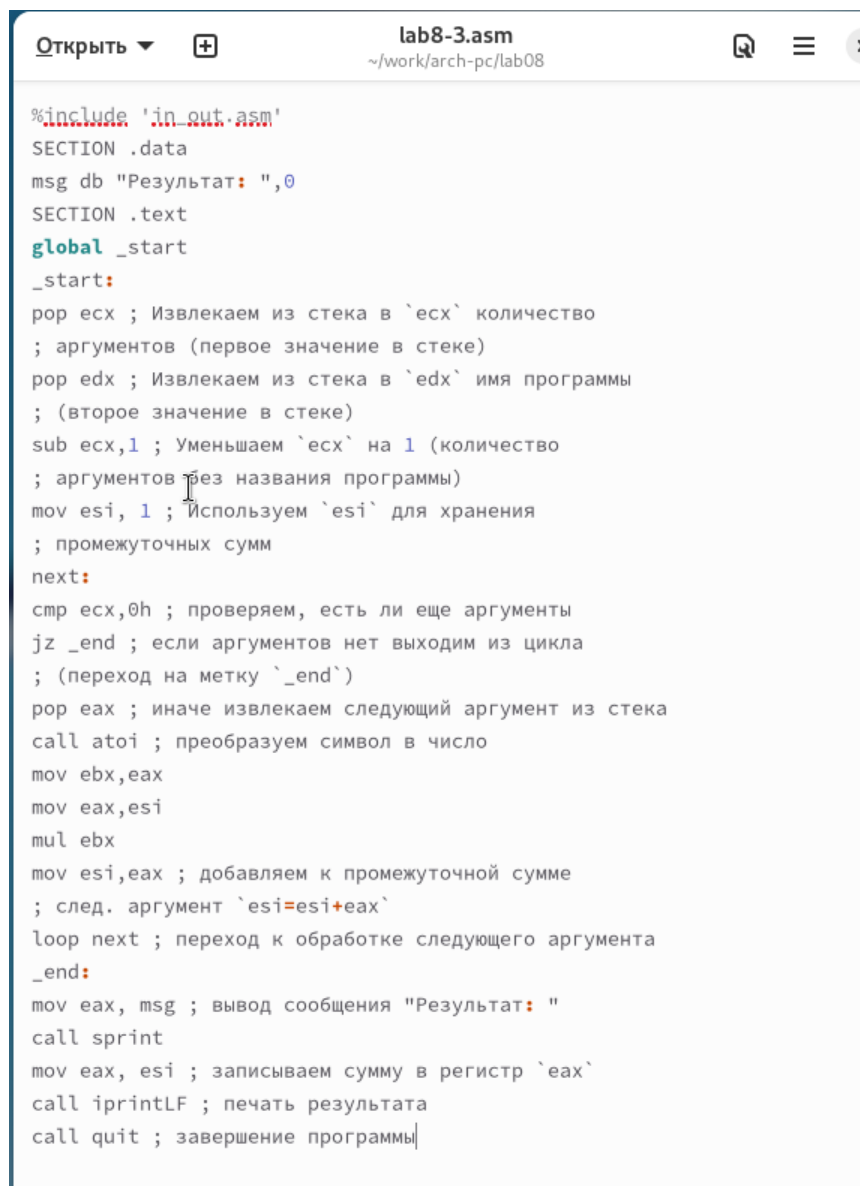
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
            ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
            ; (второе значение в стеке)
    sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
            ; аргументов без названия программы)
    mov esi, 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 esi,eax ; добавляем к промежуточной сумме
            ; 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
```

Рис. 2.10: Программа lab8-3.asm

```
abusmanova@fedora: ~/.work/arch-pc/lab08$
abusmanova@fedora: ~/.work/arch-pc/lab08$ nasm -f elf lab8-3.asm
abusmanova@fedora: ~/.work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
abusmanova@fedora: ~/.work/arch-pc/lab08$ ./lab8-3 3 4 5 6 7
Результат: 25
abusmanova@fedora: ~/.work/arch-pc/lab08$ ./lab8-3 1 2 3
Результат: 6
abusmanova@fedora: ~/.work/arch-pc/lab08$
```

Рис. 2.11: Запуск программы lab8-3.asm

Изменила текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. 2.12) (рис. 2.13).



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

%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 ; добавляем к промежуточной сумме
; 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
```

Рис. 2.12: Программа lab8-3.asm


```
abusmanova@fedora:~/work/arch-pc/lab08$  
abusmanova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm  
abusmanova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3  
abusmanova@fedora:~/work/arch-pc/lab08$ ./lab8-3 3 4 5 6 7  
Результат: 2520  
abusmanova@fedora:~/work/arch-pc/lab08$ ./lab8-3 1 2 3  
Результат: 6  
abusmanova@fedora:~/work/arch-pc/lab08$
```

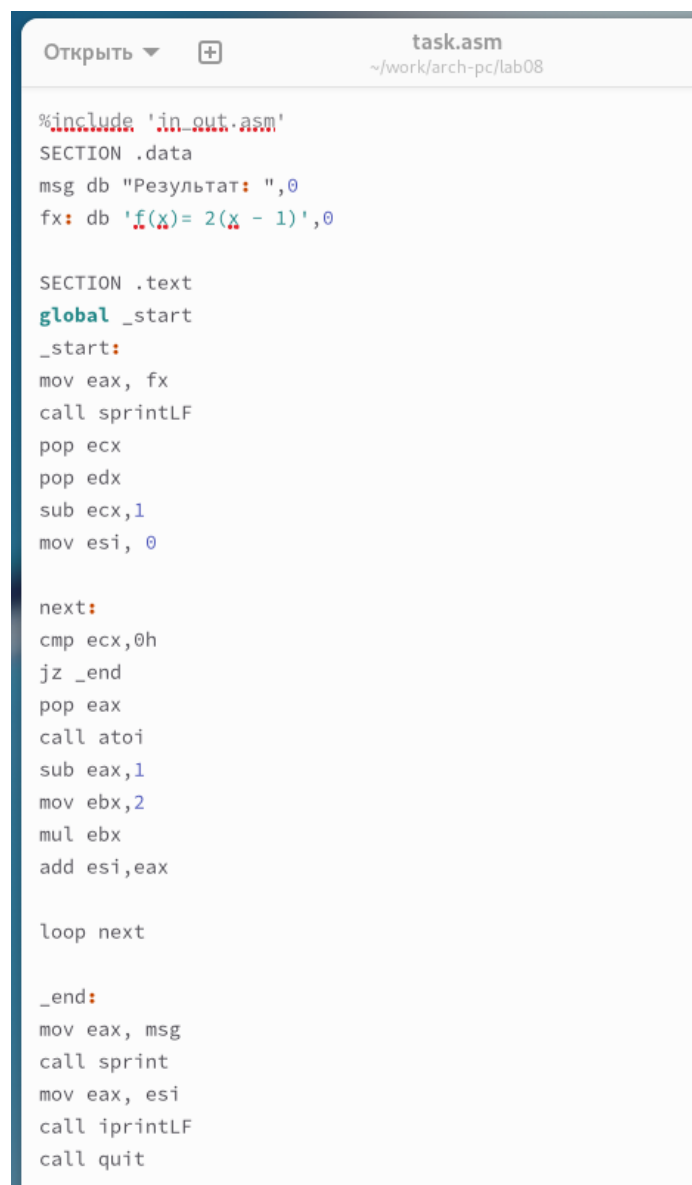
Рис. 2.13: Запуск программы lab8-3.asm


2.2 Самостоятельное задание

Напишите 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 передаются как аргументы. Вид функции $f(x)$ следует выбрать согласно таблице 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создала исполняемый файл и проверила его работу на нескольких наборах x . (рис. 2.14) (рис. 2.15).

Для варианта 4

$$f(x) = 2(x - 1)$$



```
Открыть ▾  task.asm
~\work\arch-pc\lab08

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 2(x - 1)',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
sub eax,1
mov ebx,2
mul ebx
add esi,eax

loop next

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

Рис. 2.14: Программа task.asm

```
abusmanova@fedora:~/work/arch-pc/lab08$  
abusmanova@fedora:~/work/arch-pc/lab08$ nasm -f elf task.asm  
abusmanova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 task.o -o task  
abusmanova@fedora:~/work/arch-pc/lab08$ ./task  
f(x)= 2(x - 1)  
Результат: 0  
abusmanova@fedora:~/work/arch-pc/lab08$ ./task 1  
f(x)= 2(x - 1)  
Результат: 0  
abusmanova@fedora:~/work/arch-pc/lab08$ ./task 5  
f(x)= 2(x - 1)  
Результат: 8  
abusmanova@fedora:~/work/arch-pc/lab08$ ./task 5 4 3 2 1 5  
f(x)= 2(x - 1)  
Результат: 28  
abusmanova@fedora:~/work/arch-pc/lab08$
```

Рис. 2.15: Запуск программы task.asm

Убедилась, что программа правильно вычисляет $f(1) = 0$, $f(5) = 8$.

3 Выводы

Освоила работу со стеком, циклами и аргументами на ассемблере NASM.