

Отчёт по лабораторной работе 8

Архитектура компьютера

Хаоладар Шаханеоядж НПИ-01-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Неализация циклов в NASM	6
2.2	Обработка аргументов командной строки	12
2.3	Задание для самостоятельной работы	16
3	Выводы	19

Список иллюстраций

2.1	Программа в файле lab8-1.asm	7
2.2	Запуск программы lab8-1.asm	8
2.3	Программа в файле lab8-1.asm	9
2.4	Запуск программы lab8-1.asm	10
2.5	Программа в файле lab8-1.asm	11
2.6	Запуск программы lab8-1.asm	12
2.7	Программа в файле lab8-2.asm	13
2.8	Запуск программы lab8-2.asm	13
2.9	Программа в файле lab8-3.asm	14
2.10	Запуск программы lab8-3.asm	14
2.11	Программа в файле lab8-3.asm	15
2.12	Запуск программы lab8-3.asm	15
2.13	Программа в файле task.asm	17
2.14	Запуск программы task.asm	18

Список таблиц

1 Цель работы

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

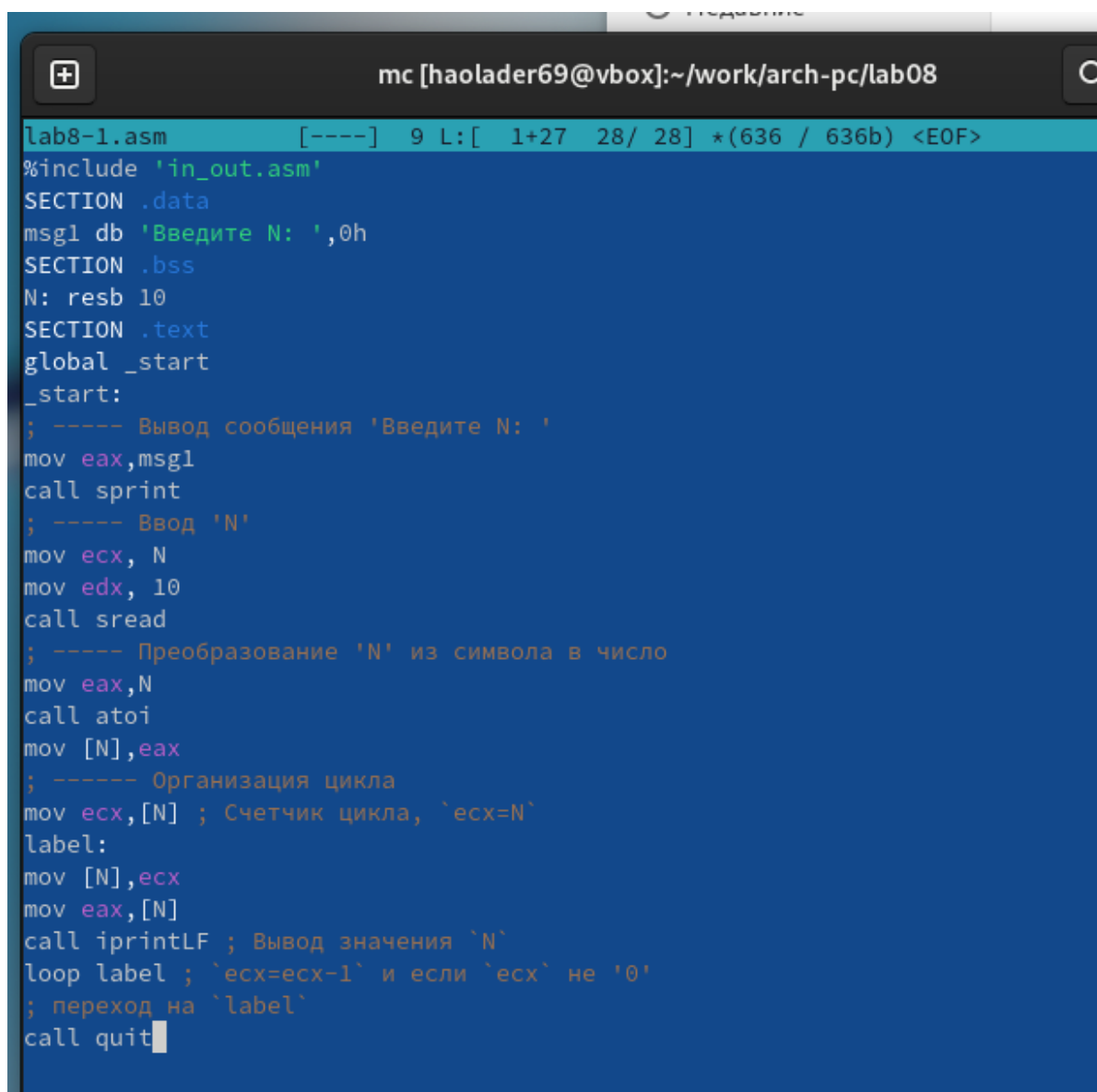
2 Выполнение лабораторной работы

2.1 Неализация циклов в NASM

Создал каталог для программам лабораторной работы № 8 и файл lab8-1.asm

При реализации циклов в NASM с использованием инструкции loop необходимо помнить о том, что эта инструкция использует регистр есх в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра есх.

Написал в файл lab8-1.asm текст программы из листинга 8.1. Создал исполняемый файл и проверил его работу.



```
lab8-1.asm [----] 9 L: [ 1+27 28/ 28] *(636 / 636b) <EOF>
#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
```

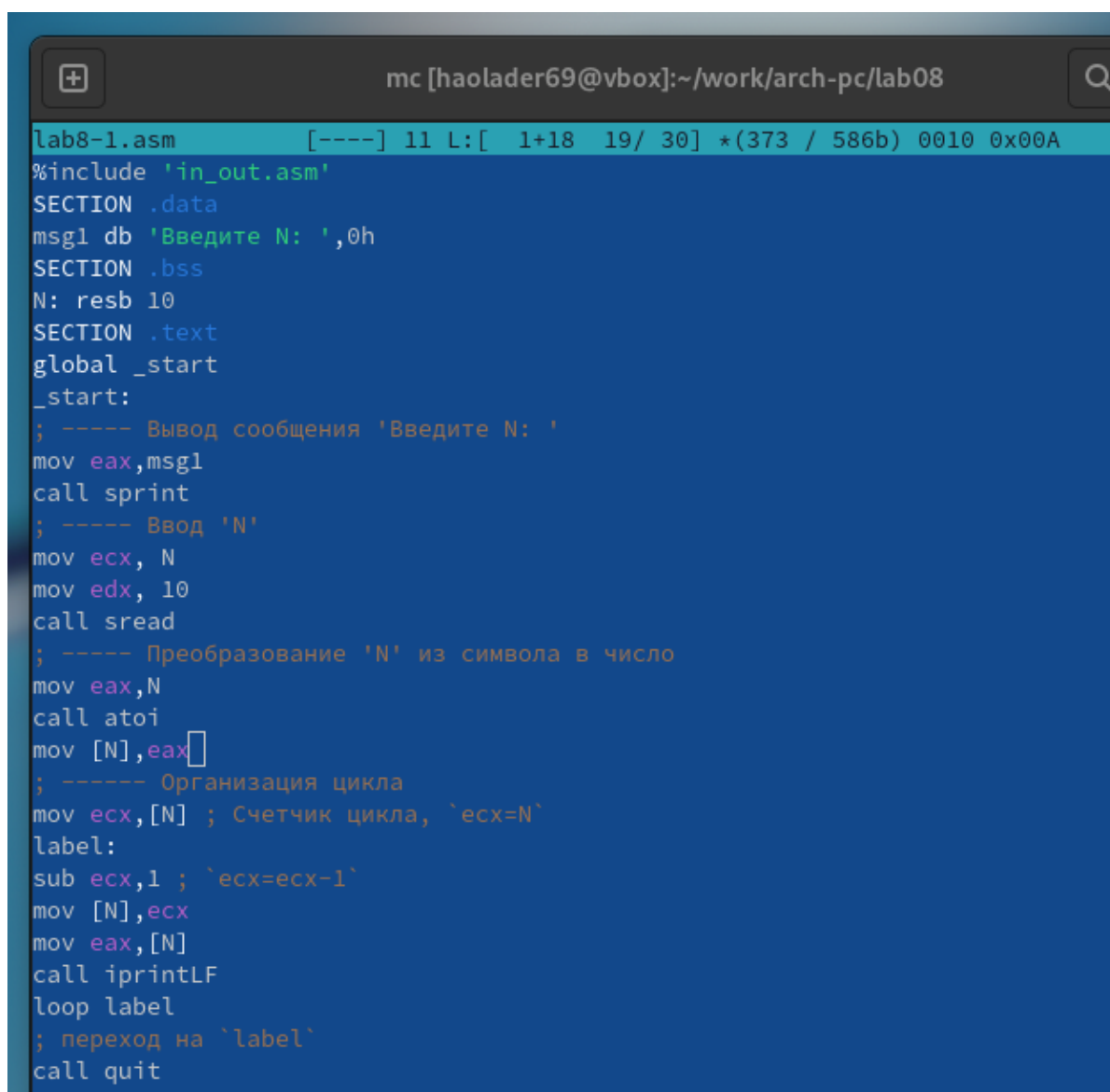
Рис. 2.1: Программа в файле lab8-1.asm

```
haolader69@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
haolader69@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
haolader69@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
haolader69@vbox:~/work/arch-pc/lab08$
```

Рис. 2.2: Запуск программы lab8-1.asm

Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Изменил текст программы добавив изменение значение регистра `ecx` в цикле.

Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N`.



```
lab8-1.asm [----] 11 L: [ 1+18 19/ 30] *(373 / 586b) 0010 0x00A
#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
```

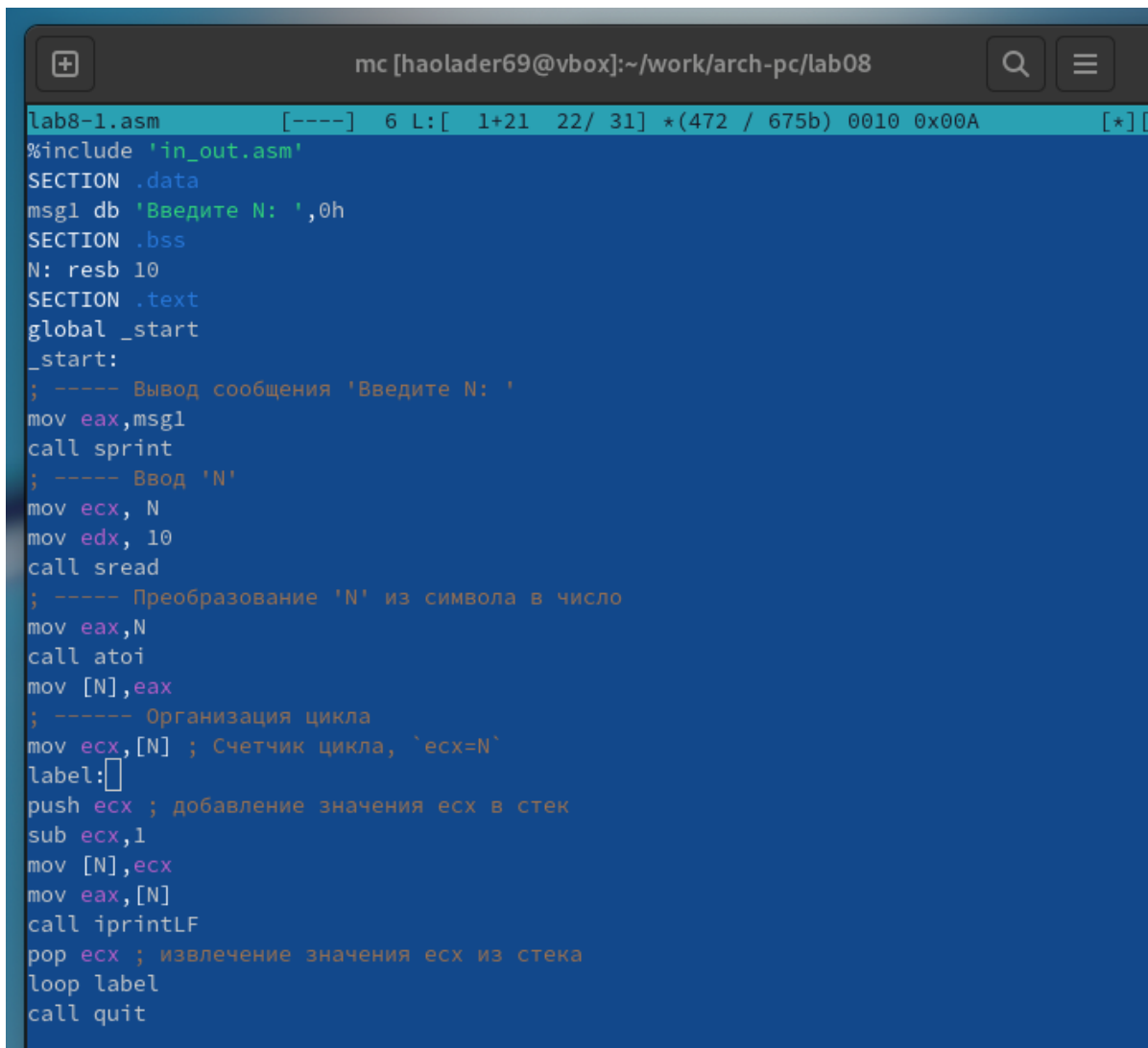
Рис. 2.3: Программа в файле lab8-1.asm

```
4294937888
4294937886
4294937884
4294937882
4294937880
4294937878
4294937^C
haolader69@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
haolader69@vbox:~/work/arch-pc/lab08$
```

Рис. 2.4: Запуск программы lab8-1.asm

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внес изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`. Создал исполняемый файл и проверьте его работу.

Программа выводит числа от $N-1$ до 0, число проходов цикла соответствует N .



```
lab8-1.asm      [----]  6 L: [  1+21  22/ 31] *(472 / 675b) 0010 0x00A  [*]  
%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. 2.5: Программа в файле lab8-1.asm

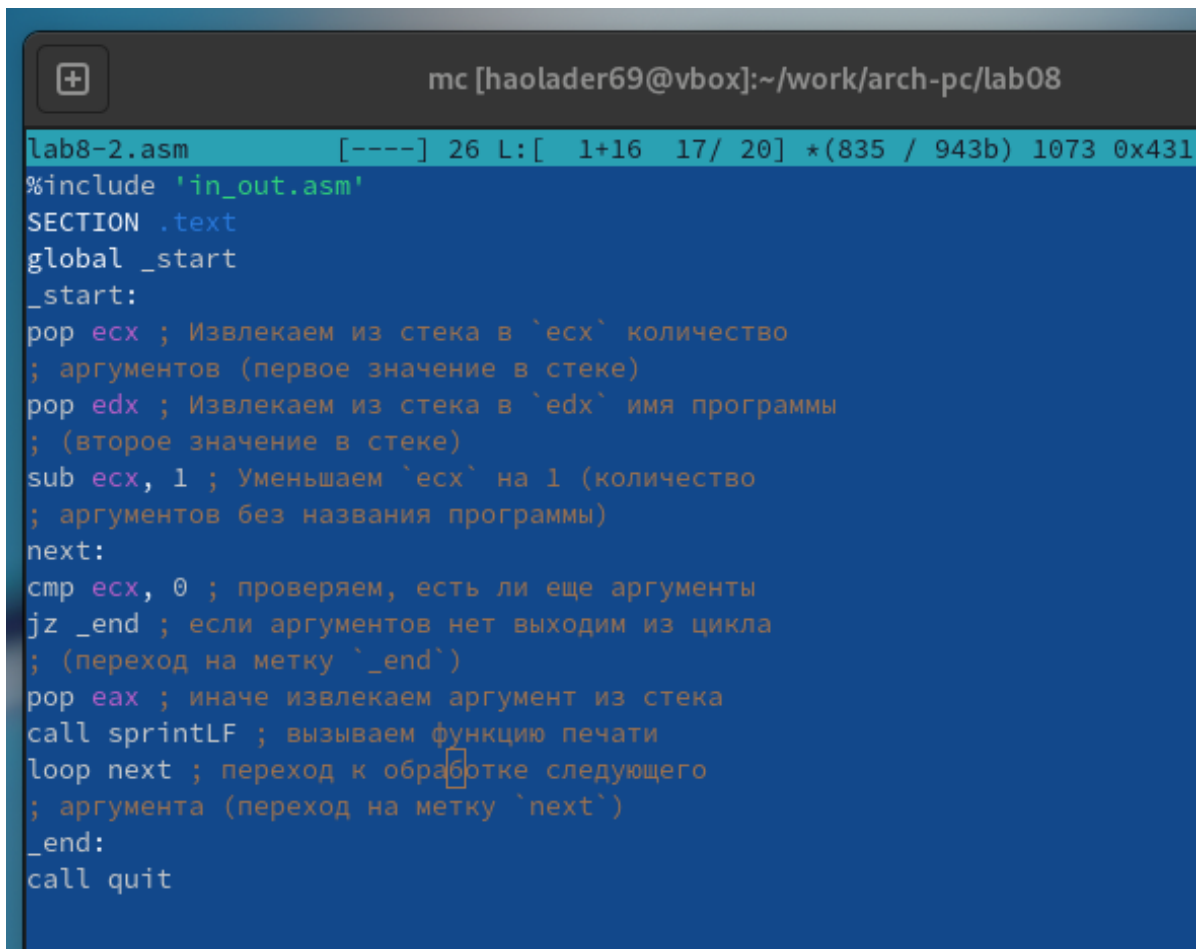
```
haolader69@vbox:~/work/arch-pc/lab08$  
haolader69@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
haolader69@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1  
haolader69@vbox:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 5  
4  
3  
2  
1  
0  
haolader69@vbox:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 4  
3  
2  
1  
0  
haolader69@vbox:~/work/arch-pc/lab08$
```

Рис. 2.6: Запуск программы lab8-1.asm

2.2 Обработка аргументов командной строки

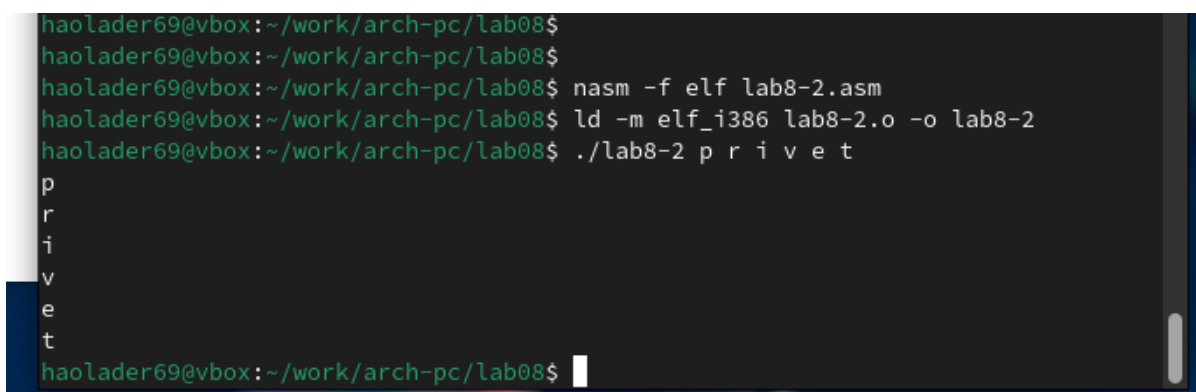
Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2.

Создал исполняемый файл и запустил его, указав аргументы. Программа обработала 5 аргументов. Аргументами считаются слова/числа, разделенные пробелом.



```
lab8-2.asm [----] 26 L:[ 1+16 17/ 20] *(835 / 943b) 1073 0x431
#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 sprintfLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

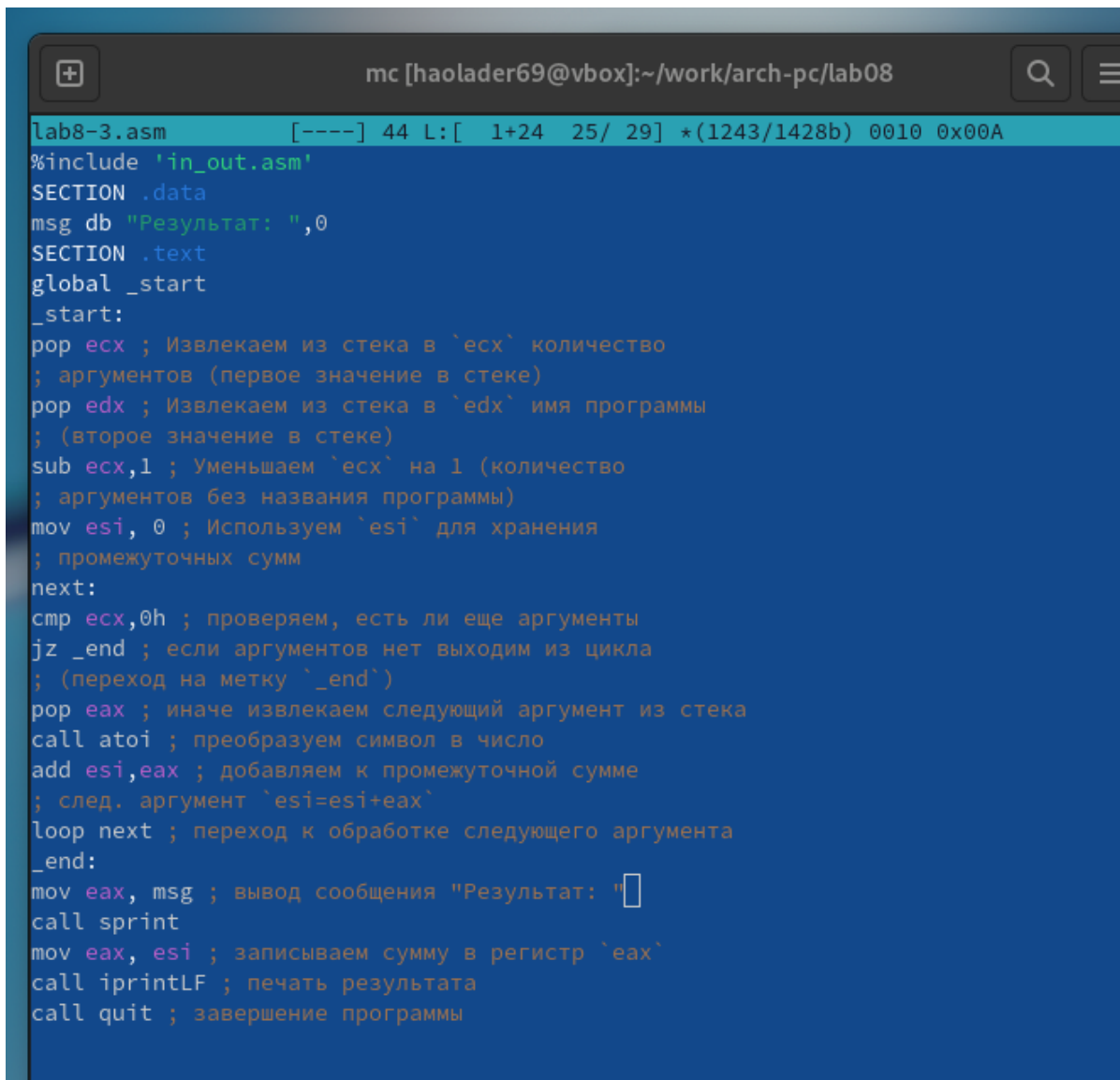
Рис. 2.7: Программа в файле lab8-2.asm



```
haolader69@vbox:~/work/arch-pc/lab08$
haolader69@vbox:~/work/arch-pc/lab08$
haolader69@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
haolader69@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
haolader69@vbox:~/work/arch-pc/lab08$ ./lab8-2 p r i v e t
p
r
i
v
e
t
haolader69@vbox:~/work/arch-pc/lab08$
```

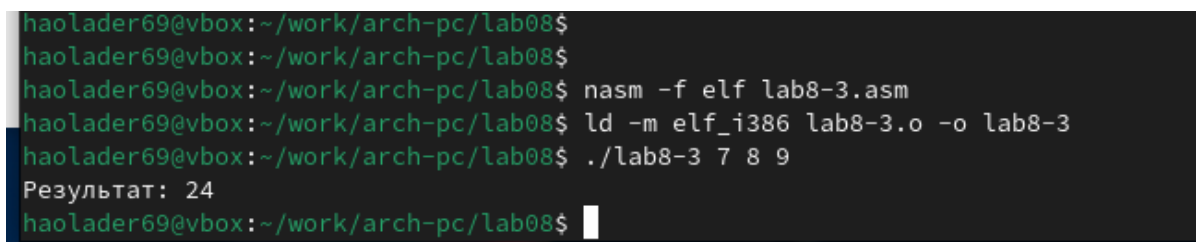
Рис. 2.8: Запуск программы lab8-2.asm

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.



```
lab8-3.asm [----] 44 L: [ 1+24 25/ 29] *(1243/1428b) 0010 0x00A
#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. 2.9: Программа в файле lab8-3.asm

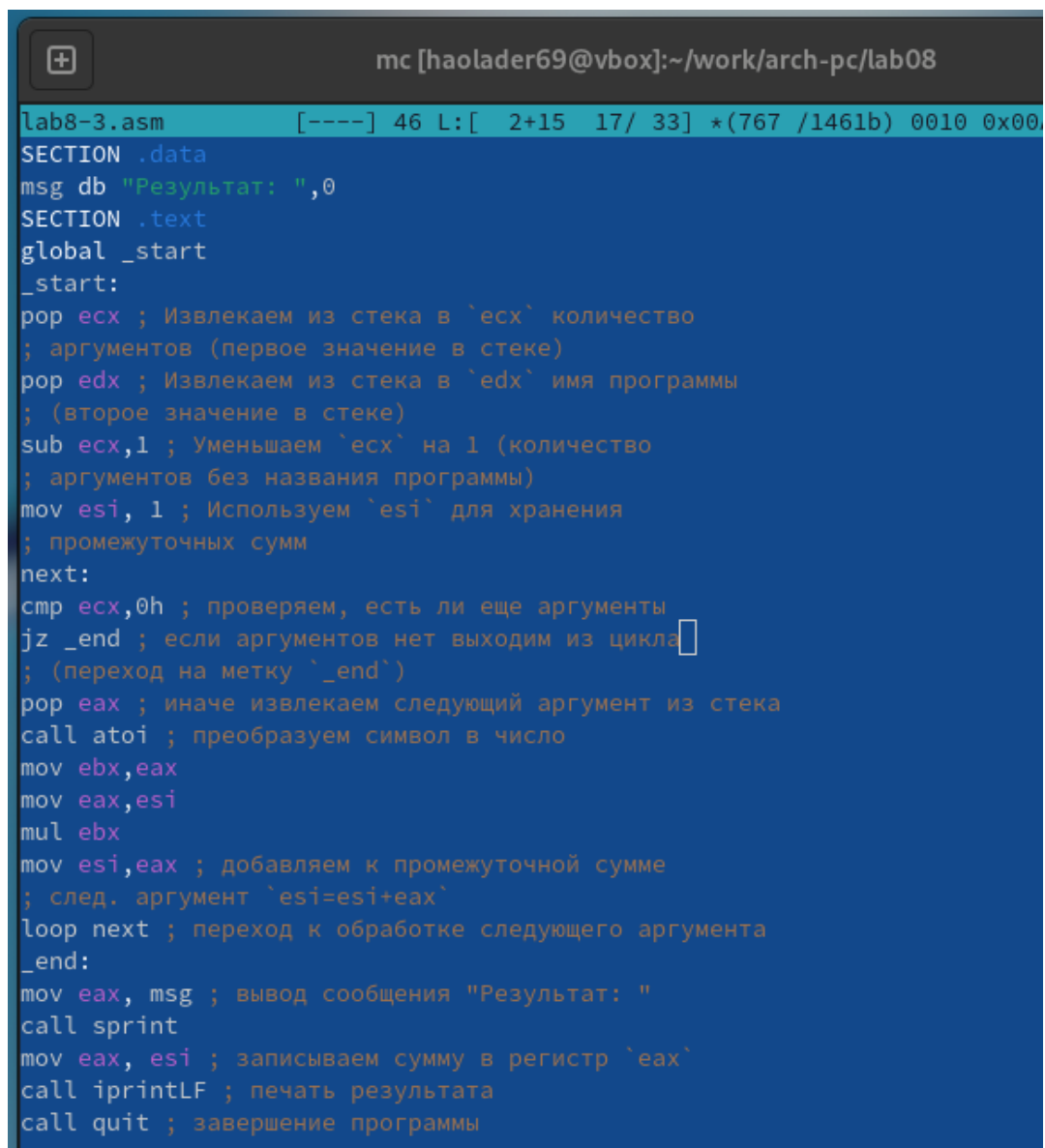


```
haolader69@vbox:~/work/arch-pc/lab08$
haolader69@vbox:~/work/arch-pc/lab08$
haolader69@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
haolader69@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
haolader69@vbox:~/work/arch-pc/lab08$ ./lab8-3 7 8 9
Результат: 24
haolader69@vbox:~/work/arch-pc/lab08$
```

Рис. 2.10: Запуск программы lab8-3.asm

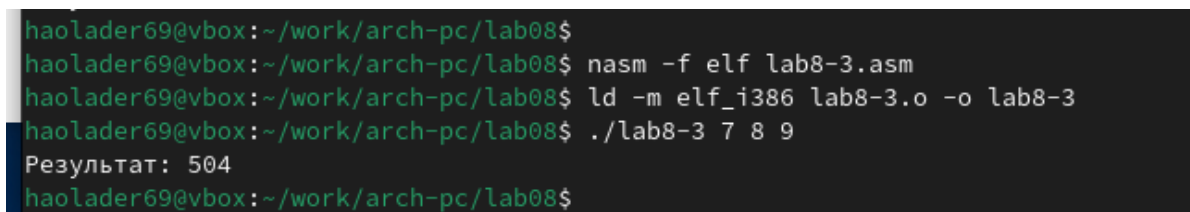
Изменил текст программы из листинга 8.3 для вычисления произведения аргу-

ментов командной строки.



```
mc [haolader69@vbox]:~/work/arch-pc/lab08
lab8-3.asm [----] 46 L:[ 2+15 17/ 33] *(767 /1461b) 0010 0x00
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. 2.11: Программа в файле lab8-3.asm



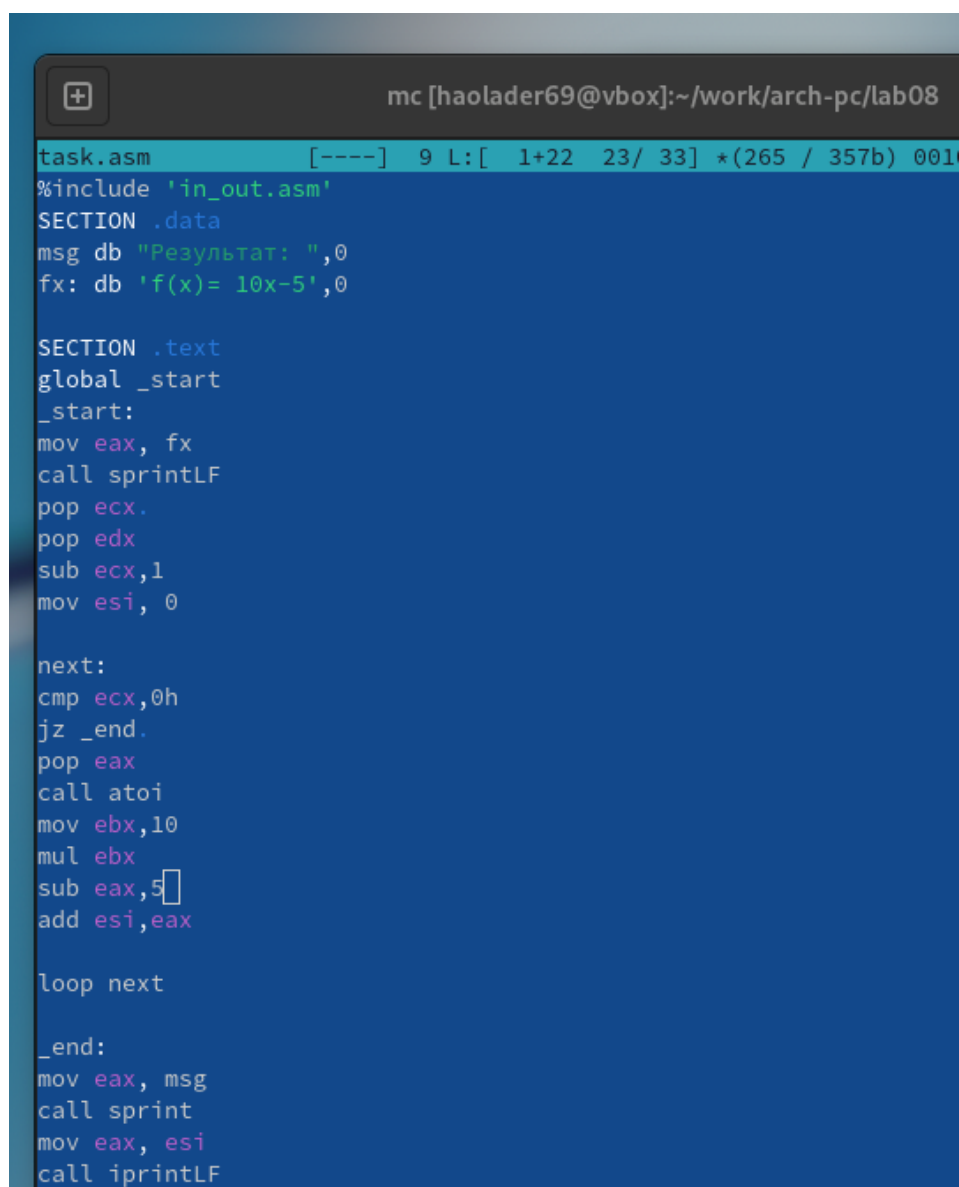
```
haolader69@vbox:~/work/arch-pc/lab08$
haolader69@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
haolader69@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
haolader69@vbox:~/work/arch-pc/lab08$ ./lab8-3 7 8 9
Результат: 504
haolader69@vbox:~/work/arch-pc/lab08$
```

Рис. 2.12: Запуск программы lab8-3.asm

2.3 Задание для самостоятельной работы

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 $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x .

для варианта 3 $f(x) = 10x - 5$



```
mc [haolader69@vbox]:~/work/arch-pc/lab08
task.asm [----] 9 L:[ 1+22 23/ 33] *(265 / 357b) 001
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 10x-5',0

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

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx,10
mul ebx
sub eax,5
add esi,eax

loop next

_end:
mov eax, msg
call sprintf
mov eax, esi
call iprintLF
```

Рис. 2.13: Программа в файле task.asm

Для проверки я запустил сначала с одним аргументом.

Затем подал несколько аргументов и получил сумму значений функции.

```
haolader69@vbox:~/work/arch-pc/lab08$  
haolader69@vbox:~/work/arch-pc/lab08$ nasm -f elf task.asm  
haolader69@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 task.o -o task  
haolader69@vbox:~/work/arch-pc/lab08$ ./task  
f(x)= 10x-5  
Результат: 0  
haolader69@vbox:~/work/arch-pc/lab08$ ./task 1  
f(x)= 10x-5  
Результат: 5  
haolader69@vbox:~/work/arch-pc/lab08$ ./task 1  
f(x)= 10x-5  
Результат: 5  
haolader69@vbox:~/work/arch-pc/lab08$ ./task 4  
f(x)= 10x-5  
Результат: 35  
haolader69@vbox:~/work/arch-pc/lab08$ ./task 5 6 1 2  
f(x)= 10x-5  
Результат: 120  
haolader69@vbox:~/work/arch-pc/lab08$
```

Рис. 2.14: Запуск программы task.asm

3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере `naasm`.