

Лабораторная работа №8

Архитектура компьютера

Мурашов Иван Вячеславович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация циклов в NASM	7
3.2	Обработка аргументов командной строки	16
3.3	Выполнение заданий для самостоятельной работы	24
4	Выводы	28

Список иллюстраций

3.1	Создание каталога и файла в нём	7
3.2	Редактирование файла	8
3.3	Трансляция, компоновка и запуск файлов	10
3.4	Редактирование файла	11
3.5	Трансляция, компоновка и запуск файлов	12
3.6	Трансляция, компоновка и запуск файлов	12
3.7	Трансляция, компоновка и запуск файлов	13
3.8	Редактирование файла	15
3.9	Трансляция, компоновка и запуск файлов	16
3.10	Создание файла	16
3.11	Редактирование файла	17
3.12	Трансляция, компоновка и запуск файлов	18
3.13	Создание файла	18
3.14	Редактирование файла	19
3.15	Трансляция, компоновка и запуск файлов	21
3.16	Создание файла	21
3.17	Редактирование файла	22
3.18	Трансляция, компоновка и запуск файлов	24
3.19	Создание файла	24
3.20	Редактирование файла	25
3.21	Трансляция, компоновка и запуск файлов	27
3.22	Трансляция, компоновка и запуск файлов	27

Список таблиц

1 Цель работы

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

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Выполнение заданий для самостоятельной работы

3 Выполнение лабораторной работы

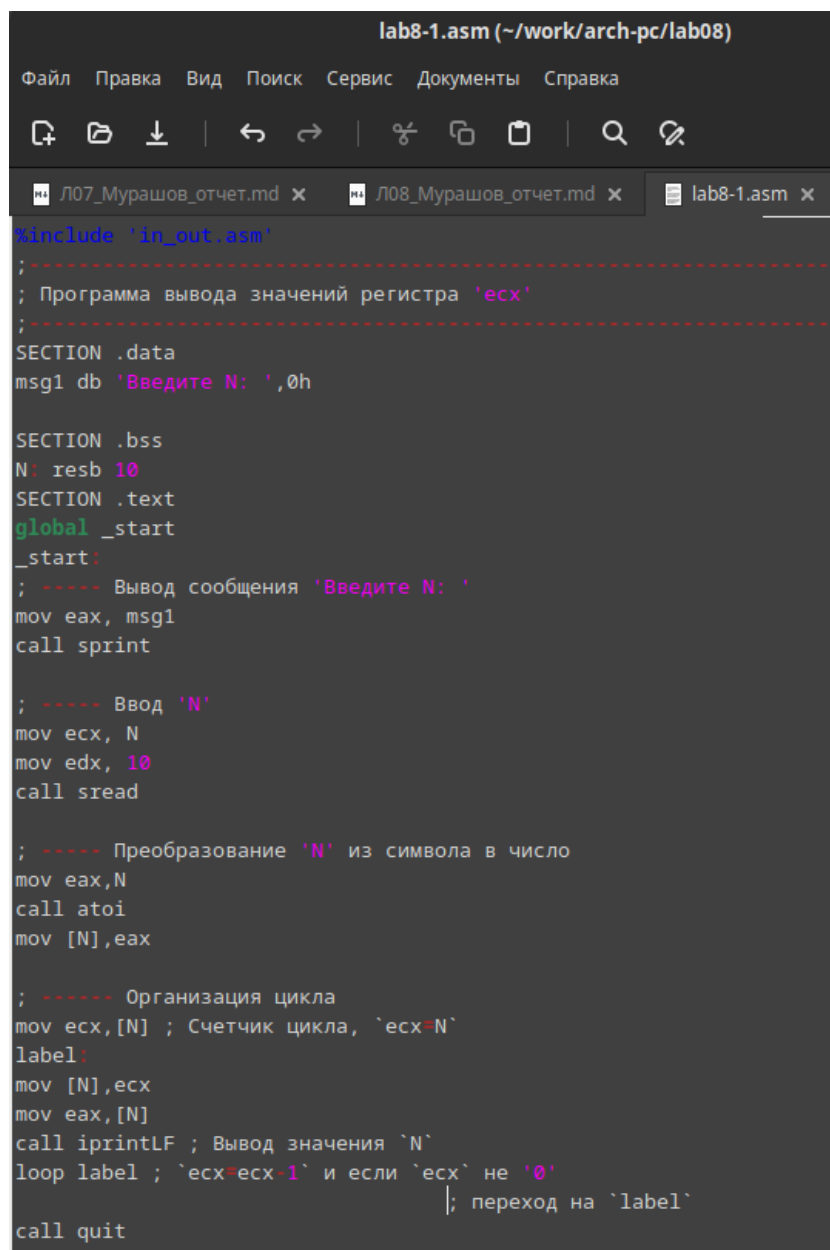
3.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm (рис. [3.1]).

```
[ivmurashov@fedora ~]$ mkdir ~/work/arch-pc/lab08  
[ivmurashov@fedora ~]$ cd ~/work/arch-pc/lab08  
[ivmurashov@fedora lab08]$ touch lab8-1.asm
```

Рис. 3.1: Создание каталога и файла в нём

Ввожу в файл lab8-1.asm текст программы из листинга 8.1 (рис. [3.2]).



```
lab8-1.asm (~/work/arch-pc/lab08)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Иконки: открыть, сохранить, отменить, повторить, вырезать, копировать, вставить, найти, заменить]
Л07_Мурашов_отчет.md x  Л08_Мурашов_отчет.md x  lab8-1.asm x

%include 'in_out.asm'
;-----
; Программа вывода значений регистра 'ecx'
;-----
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 iprintlnLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Рис. 3.2: Редактирование файла

Листинг 1. Программа вывода значений регистра ecx

```
;-----
; Программа вывода значений регистра 'ecx'
;-----
%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
```

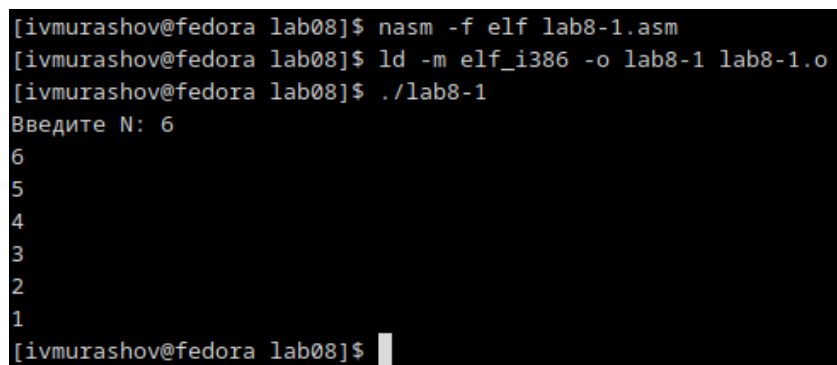
```
mov eax,[N]
```

```
call iprintLF ; Вывод значения `N`
```

```
loop label ; `ecx=ecx-1` и если `ecx` не '0' ; переход на `label`
```

`call quit`

Создаю исполняемый файл и запускаю его (рис. [3.3]).



```
[ivmurashov@fedora lab08]$ nasm -f elf lab8-1.asm
[ivmurashov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[ivmurashov@fedora lab08]$ ./lab8-1
Введите N: 6
6
5
4
3
2
1
[ivmurashov@fedora lab08]$
```

Рис. 3.3: Трансляция, компоновка и запуск файлов

Данная программа выводит все числа от N до 1 включительно. Изменяю текст программы, добавив изменение значения регистра `ecx` в цикле: (рис. [3.4]).

```
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
```

```
lab8-1.asm (~/.work/arch-pc/lab08)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Icons]
Л07_Мурашов_отчет.md x  Л08_Мурашов_отчет.md x  lab8-1.asm x

%include 'in_out.asm'
; -----
; Программа вывода значений регистра 'ecx'
; -----
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 ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit
```

Рис. 3.4: Редактирование файла

Создаю исполняемый файл и запускаю его. При вводе чётного числа программа выводит все числа от N (не включая) до 1 с интервалом в 2 - то есть все нечётные числа, так как мы дважды вычитаем 1 из значения регистра ecx в ходе реализации цикла (рис. [3.5]).

```
[ivmurashov@fedora lab08]$ ./lab8-1
Введите N: 20
19
17
15
13
11
9
7
5
3
1
[ivmurashov@fedora lab08]$
```

Рис. 3.5: Трансляция, компоновка и запуск файлов

А при вводе нечётного числа программа выводит бесконечную последовательность значений поскольку значение регистра `ecx` переваливает за 0 (рис. [3.6]), (рис. [3.7]).

```
[ivmurashov@fedora lab08]$ ./lab8-1
Введите N: 21
```

Рис. 3.6: Трансляция, компоновка и запуск файлов

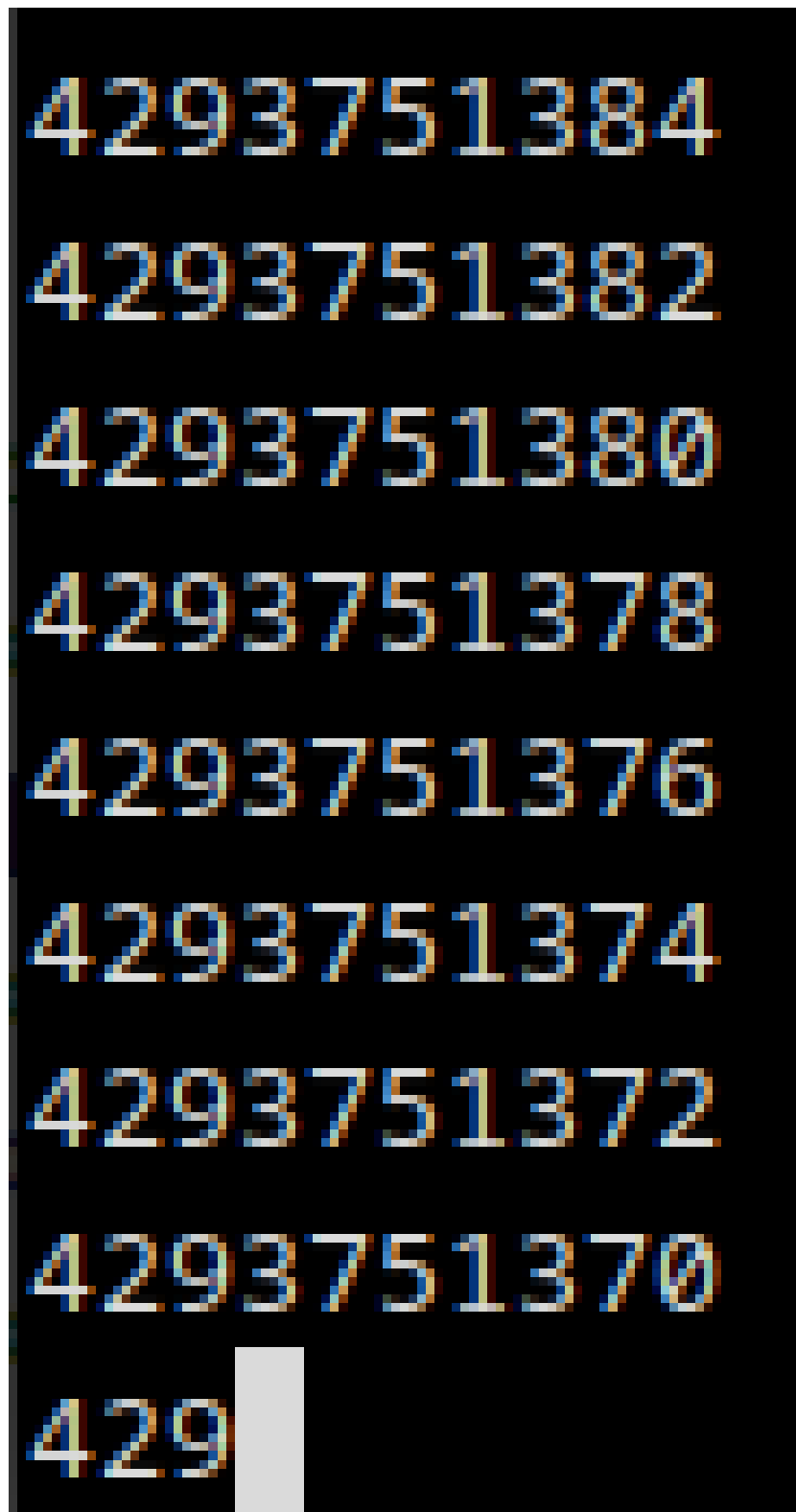
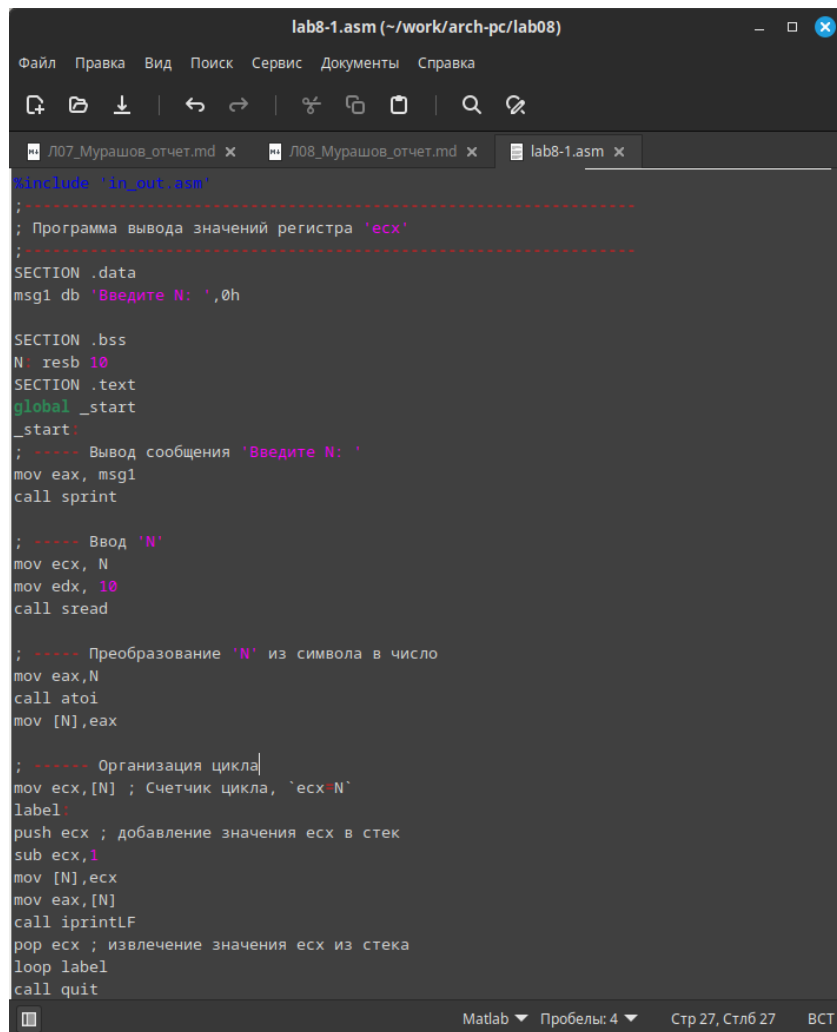


Рис. 3.7: Трансляция, компоновка и запуск файлов

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

```
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
```



```
lab8-1.asm (~/work/arch-pc/lab08)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[+] [-] [v] [x] | [↶] [↷] | [✂] [↵] [📄] | [🔍] [🔗]
L07_Мурашов_отчет.md x  L08_Мурашов_отчет.md x  lab8-1.asm x
%include 'in_out.asm'
; -----
; Программа вывода значений регистра 'ecx'
; -----
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
```

Matlab ▾ Пробелы: 4 ▾ Стр 27, Стлб 27 ВСТ

Рис. 3.8: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. [3.9]).

```

[ivmurashov@fedora lab08]$ nasm -f elf lab8-1.asm
[ivmurashov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[ivmurashov@fedora lab08]$ ./lab8-1
Введите N: 7
6
5
4
3
2
1
0
[ivmurashov@fedora lab08]$

```

Рис. 3.9: Трансляция, компоновка и запуск файлов

В данном случае число проходов цикла соответствует значению N. Программа выводит значения от (N-1) до 0 включительно.

3.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (рис. [3.10]).

```

[ivmurashov@fedora lab08]$ touch lab8-2.asm

```

Рис. 3.10: Создание файла

Ввожу в файл lab8-2.asm текст программы из листинга 8.2 (рис. [3.11]).

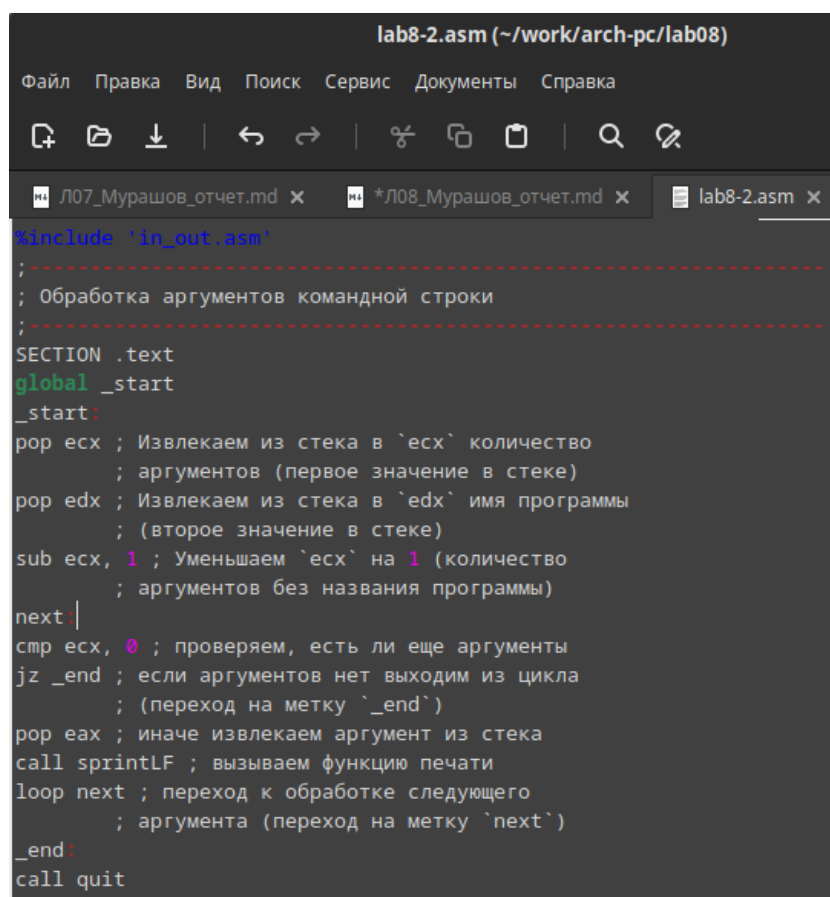


Рис. 3.11: Редактирование файла

Листинг 2. Программа выводящая на экран аргументы командной строки

```

;-----
; Обработка аргументов командной строки
;-----

%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

```

Создаю исполняемый файл и запускаю его (рис. [3.12]).

```

[ivmurashov@fedora lab08]$ nasm -f elf lab8-2.asm
[ivmurashov@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[ivmurashov@fedora lab08]$ ./lab8-2
[ivmurashov@fedora lab08]$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[ivmurashov@fedora lab08]$

```

Рис. 3.12: Трансляция, компоновка и запуск файлов

Программой был обработан 1 аргумент (выражение 'аргумент 2' воспринимается как 2 отдельных аргумента).

Создаю файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. [3.13]).

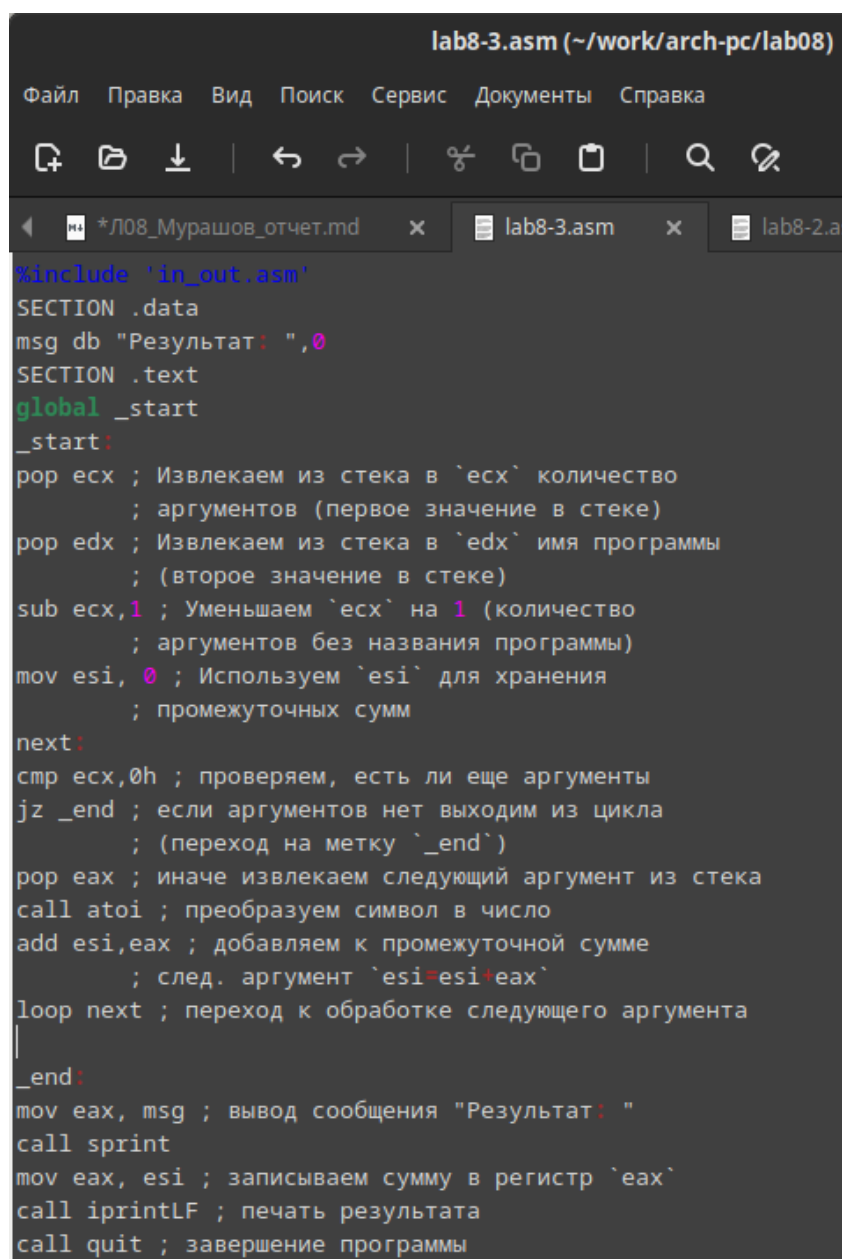
```

[ivmurashov@fedora lab08]$ touch lab8-3.asm

```

Рис. 3.13: Создание файла

Ввожу в файл lab8-3.asm текст программы из листинга 8.3 (рис. [3.14]).



```
lab8-3.asm (~/work/arch-pc/lab08)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка

%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. 3.14: Редактирование файла

Листинг 3. Программа вычисления суммы аргументов командной строки

```
%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

```

Создаю исполняемый файл и запускаю его (рис. [3.15]).

```
[ivmurashov@fedora lab08]$ nasm -f elf lab8-3.asm
[ivmurashov@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[ivmurashov@fedora lab08]$ ./lab8-3 5 9 3 1 5 7
Результат: 30
[ivmurashov@fedora lab08]$
```

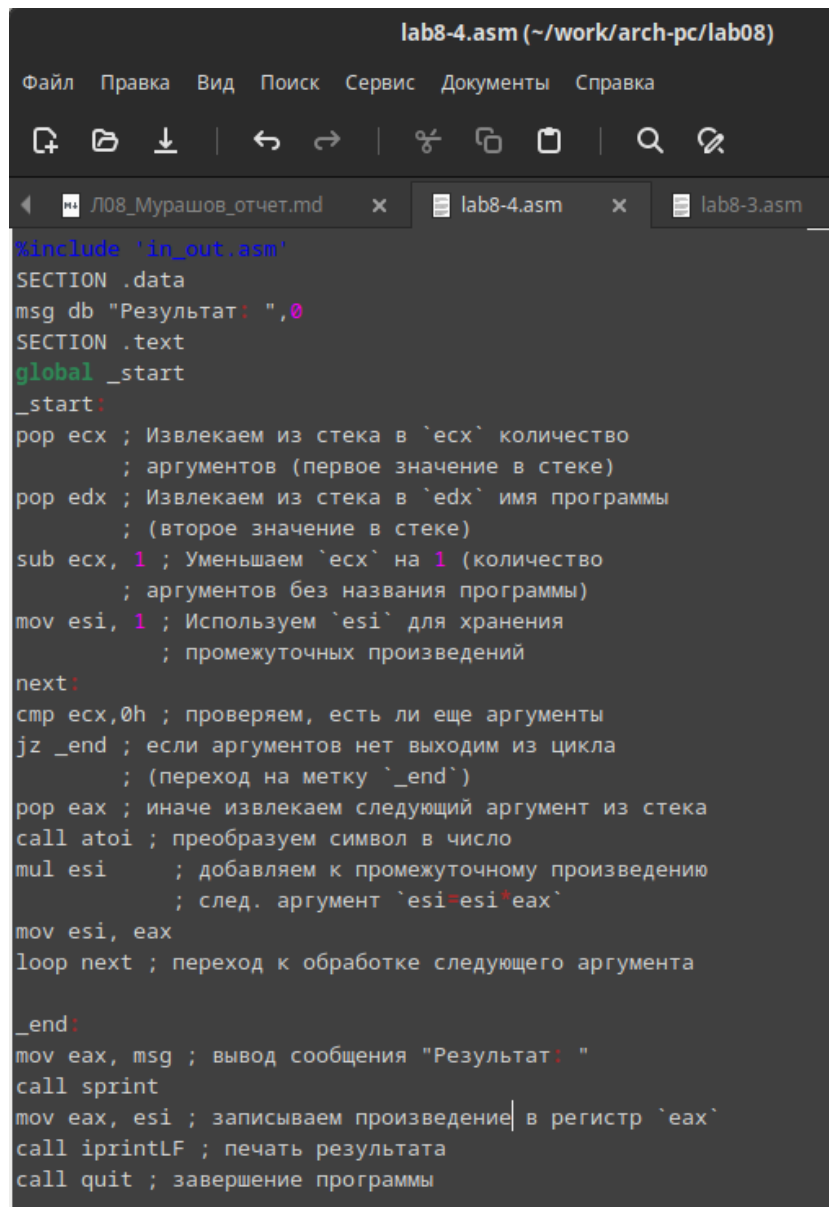
Рис. 3.15: Трансляция, компоновка и запуск файлов

Создаю файл lab8-4.asm в каталоге ~/work/arch-pc/lab08 (рис. [3.16]).

```
[ivmurashov@fedora lab08]$ touch lab8-4.asm
```

Рис. 3.16: Создание файла

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. [3.17]).



```
lab8-4.asm (~/.work/arch-pc/lab08)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Icons]
lab08_Мурашов_отчет.md x lab8-4.asm x lab8-3.asm
%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 произведений
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
mul esi ; добавляем к промежуточному произведению
        ; след. аргумент `esi=esi*eax`
mov esi, eax
loop next ; переход к обработке следующего аргумента

_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем произведение в регистр `eax`
call iprintf ; печать результата
call quit ; завершение программы
```

Рис. 3.17: Редактирование файла

Листинг 4. Программа вычисления произведения аргументов командной строки

```
%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 произведений
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
mul esi ; добавляем к промежуточному произведению
          ; след. аргумент `esi=esi*eax`
mov esi, eax
loop next ; переход к обработке следующего аргумента

_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем произведение в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы

```

Создаю исполняемый файл и запускаю его, вводя несколько аргументов. При

самостоятельном подсчёте полученные результаты совпали (рис. [3.18]).

```
[ivmurashov@fedora lab08]$ nasm -f elf lab8-4.asm
[ivmurashov@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[ivmurashov@fedora lab08]$ ./lab8-4 2 3 8
Результат: 48
[ivmurashov@fedora lab08]$
```

Рис. 3.18: Трансляция, компоновка и запуск файлов

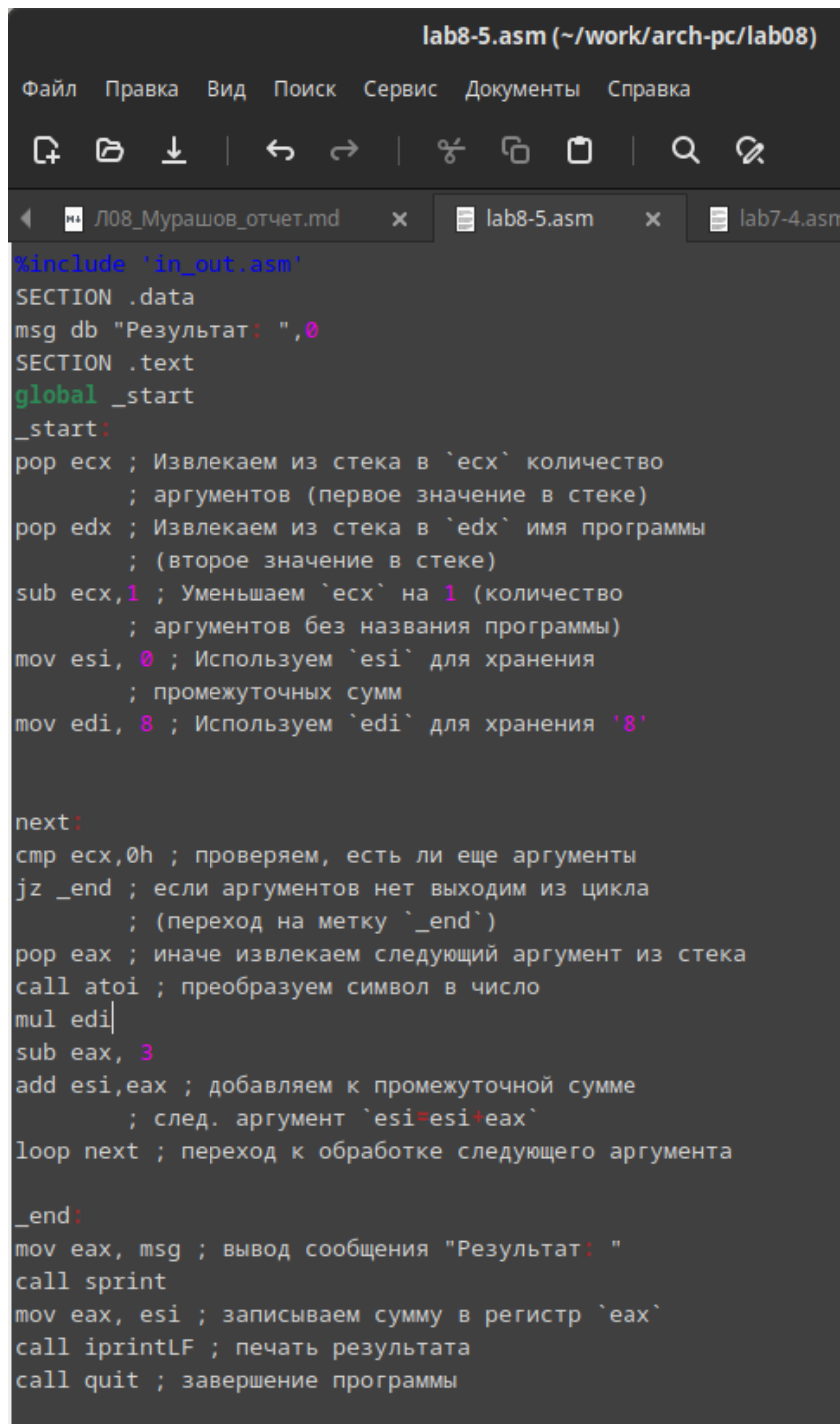
3.3 Выполнение заданий для самостоятельной работы

Создаю файл lab8-5.asm в каталоге ~/work/arch-pc/lab08 (рис. [3.19]).

```
[ivmurashov@fedora lab08]$ touch lab8-5.asm
```

Рис. 3.19: Создание файла

Пишу программу, которая находит сумму значений функции $f(x)=8x-3$ (19 вариант) для $x=x_1, x_2, x_3, \dots, x_N$, которые передаются как аргументы (рис. [3.20]).



```
lab8-5.asm (~/work/arch-pc/lab08)
Файл  Плавка  Вид  Поиск  Сервис  Документы  Справка
[+] [F] [D] | [L] [R] | [C] [P] [A] | [Q] [E]
L08_Мурашов_отчет.md x lab8-5.asm x lab7-4.asm
%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
mov edi, 8 ; Используем `edi` для хранения '8'

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
mul edi
sub eax, 3
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. 3.20: Редактирование файла

Листинг 5. Программа вычисления суммы функций для аргументов командной строки

```

%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
mov edi, 8 ; Используем `edi` для хранения '8'

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
mul edi
sub eax, 3
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

```

Создаю исполняемый файл и запускаю его, вводя следующий набор аргументов: 2, 3, 6. При самостоятельном подсчёте полученные результаты совпали (рис. [3.21]).

```

[ivmurashov@fedora lab08]$ nasm -f elf lab8-5.asm
[ivmurashov@fedora lab08]$ ld -m elf_i386 -o lab8-5 lab8-5.o
[ivmurashov@fedora lab08]$ ./lab8-5 2 3 6
Результат: 79
[ivmurashov@fedora lab08]$

```

Рис. 3.21: Трансляция, компоновка и запуск файлов

Проверяю работу программы на других наборах аргументов. При самостоятельном подсчёте полученные результаты также совпали (рис. [3.22]).

```

[ivmurashov@fedora lab08]$ ./lab8-5 1 1 1 1
Результат: 20
[ivmurashov@fedora lab08]$ ./lab8-5 3 7
Результат: 74
[ivmurashov@fedora lab08]$

```

Рис. 3.22: Трансляция, компоновка и запуск файлов

4 Выводы

В ходе выполнения данной лабораторной работы я приобрёл навыки написания программ с использованием циклов и обработкой аргументов командной строки.