

Отчёт по лабораторной работе 8

дисциплина: Архитектура компьютеров

Байрамов Керим Сапарович

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	Создан каталог	6
2.2	Программа lab8-1.asm	7
2.3	Запуск программы lab8-1.asm	8
2.4	Программа lab8-1.asm	9
2.5	Запуск программы lab8-1.asm	10
2.6	Программа lab8-1.asm	11
2.7	Запуск 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 lab8-task1.asm	18
2.15	Запуск программы lab8-task1.asm	19

Список таблиц

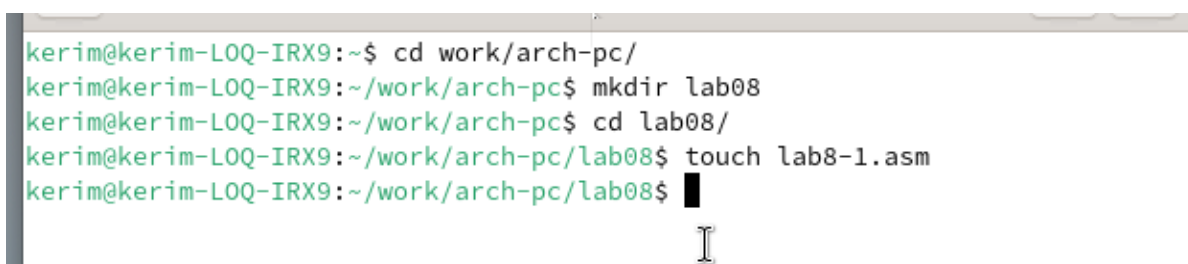
1 Цель работы

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

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

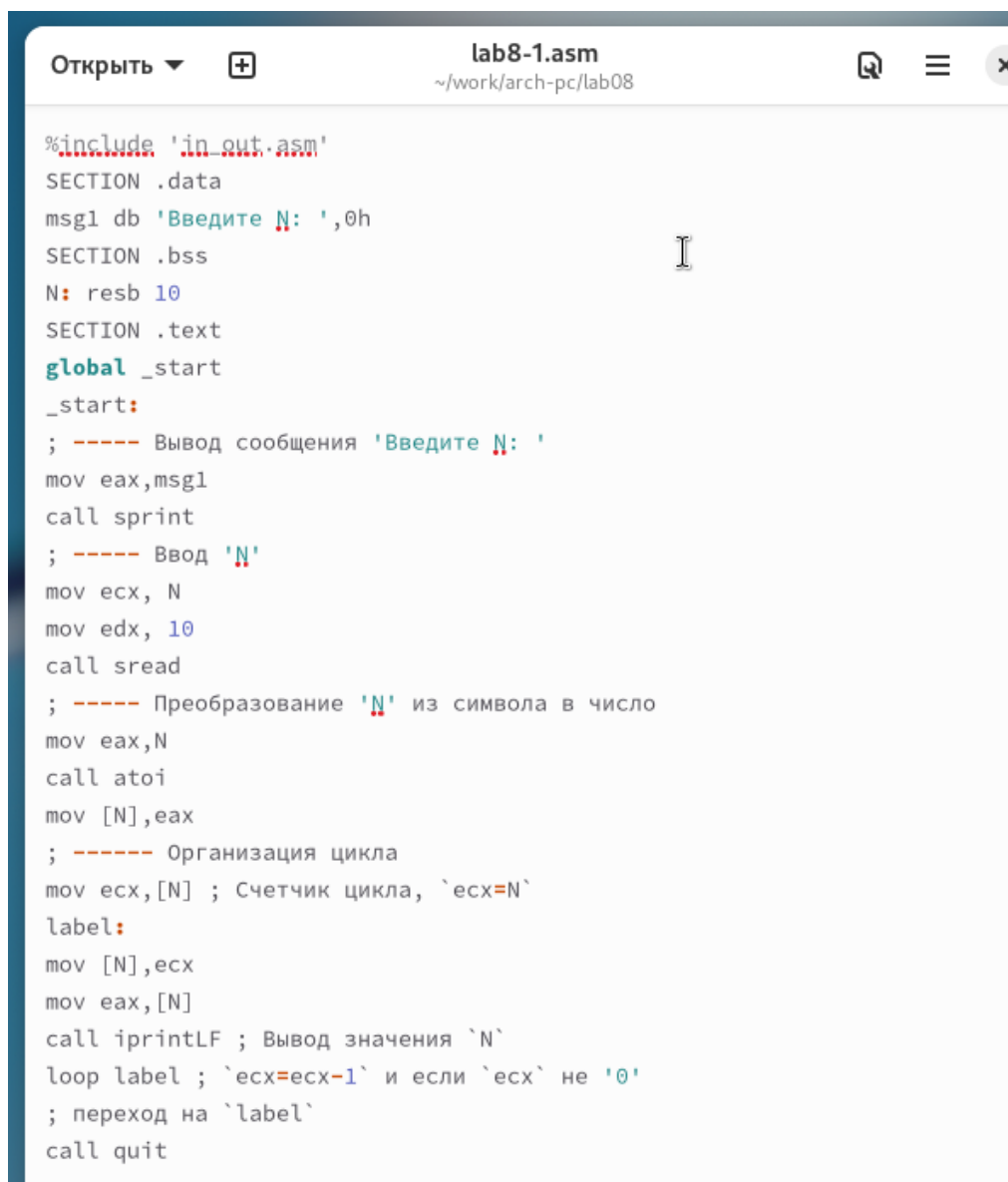


```
kerim@kerim-LOQ-IRX9:~$ cd work/arch-pc/  
kerim@kerim-LOQ-IRX9:~/work/arch-pc$ mkdir lab08  
kerim@kerim-LOQ-IRX9:~/work/arch-pc$ cd lab08/  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ touch lab8-1.asm  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ █
```

Рис. 2.1: Создан каталог

При реализации циклов в NASM с использованием инструкции `loop` необходимо помнить о том, 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 `ecx`.

Написал в файл `lab8-1.asm` текст программы из листинга 8.1 (рис. 2.2). Создал исполняемый файл и проверил 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

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 6  
6  
5  
4  
3  
2  
1  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 5  
5  
4  
3  
2  
1  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
```

Рис. 2.3: 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 значения регистра `ecx` в цикле (рис. 2.4). Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N` (рис. 2.5).



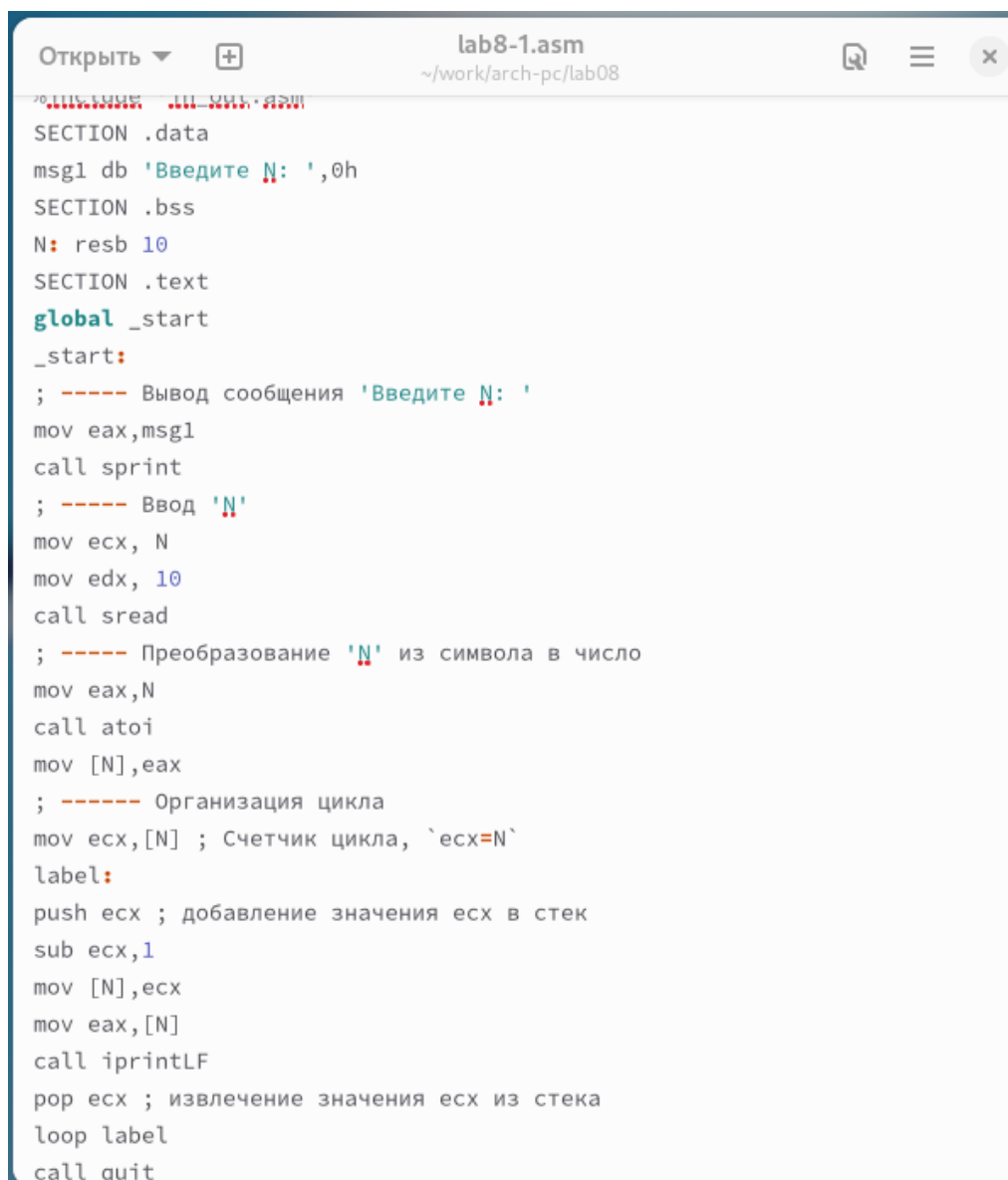
```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
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: Программа lab8-1.asm

```
4294944800
4294944858
4294944856
4294944854
4294944852
4294944850
4294944848
4294944846
429^C
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
```

Рис. 2.5: Запуск программы lab8-1.asm

Для корректной работы программы с использованием регистра `ecx` можно применить стек. Внес изменения в программу, добавив команды `push` и `pop` для сохранения значения счетчика цикла `loop` (рис. 2.6). Создал исполняемый файл и проверил его работу (рис. 2.7). Программа выводит числа от $N-1$ до 0, количество проходов цикла соответствует N .



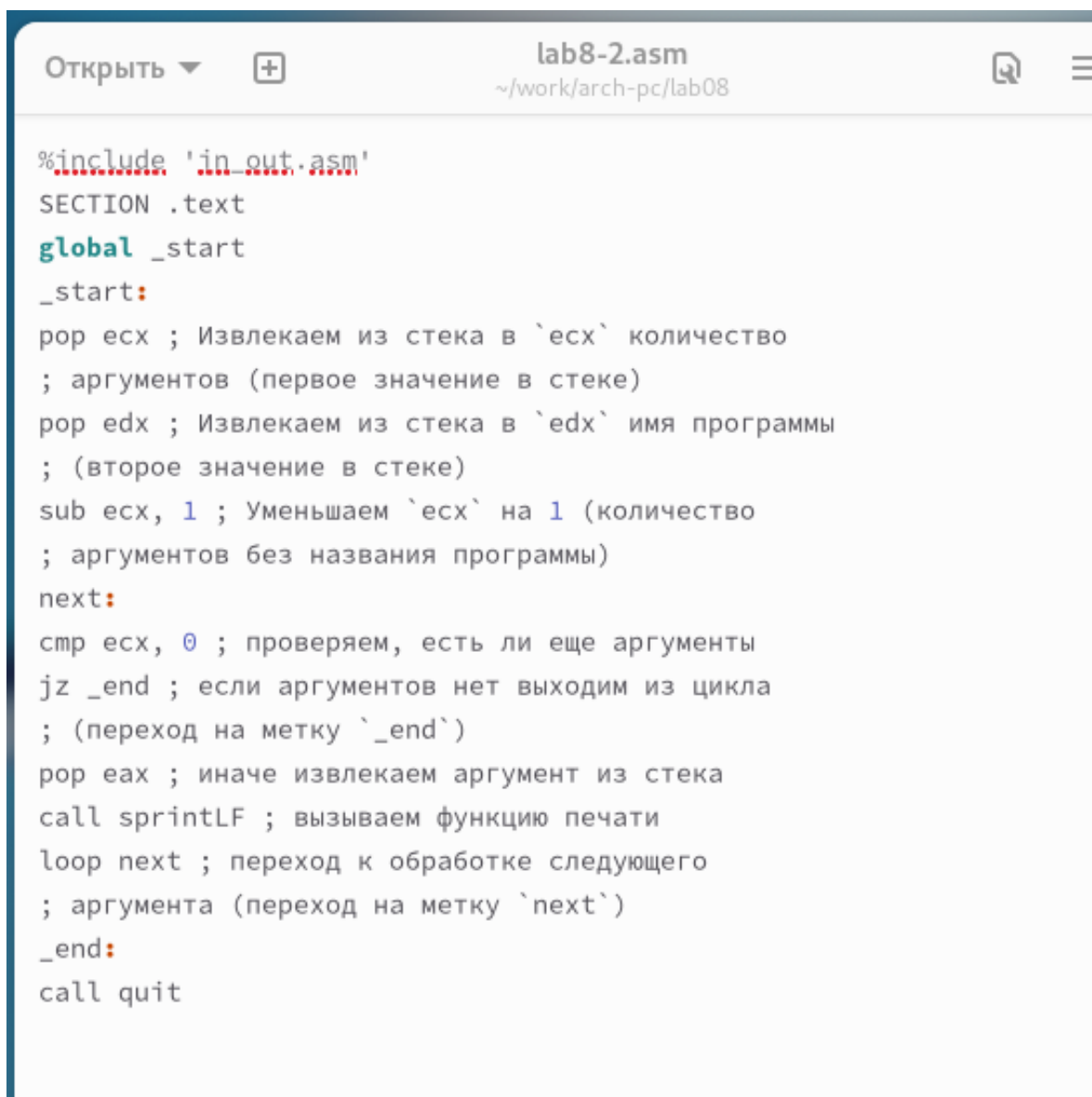
```
%include "io.inc"
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
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: Программа lab8-1.asm

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 5  
4  
3  
2  
1  
0  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 6  
5  
4  
3  
2  
1  
0  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
```

Рис. 2.7: Запуск программы lab8-1.asm

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



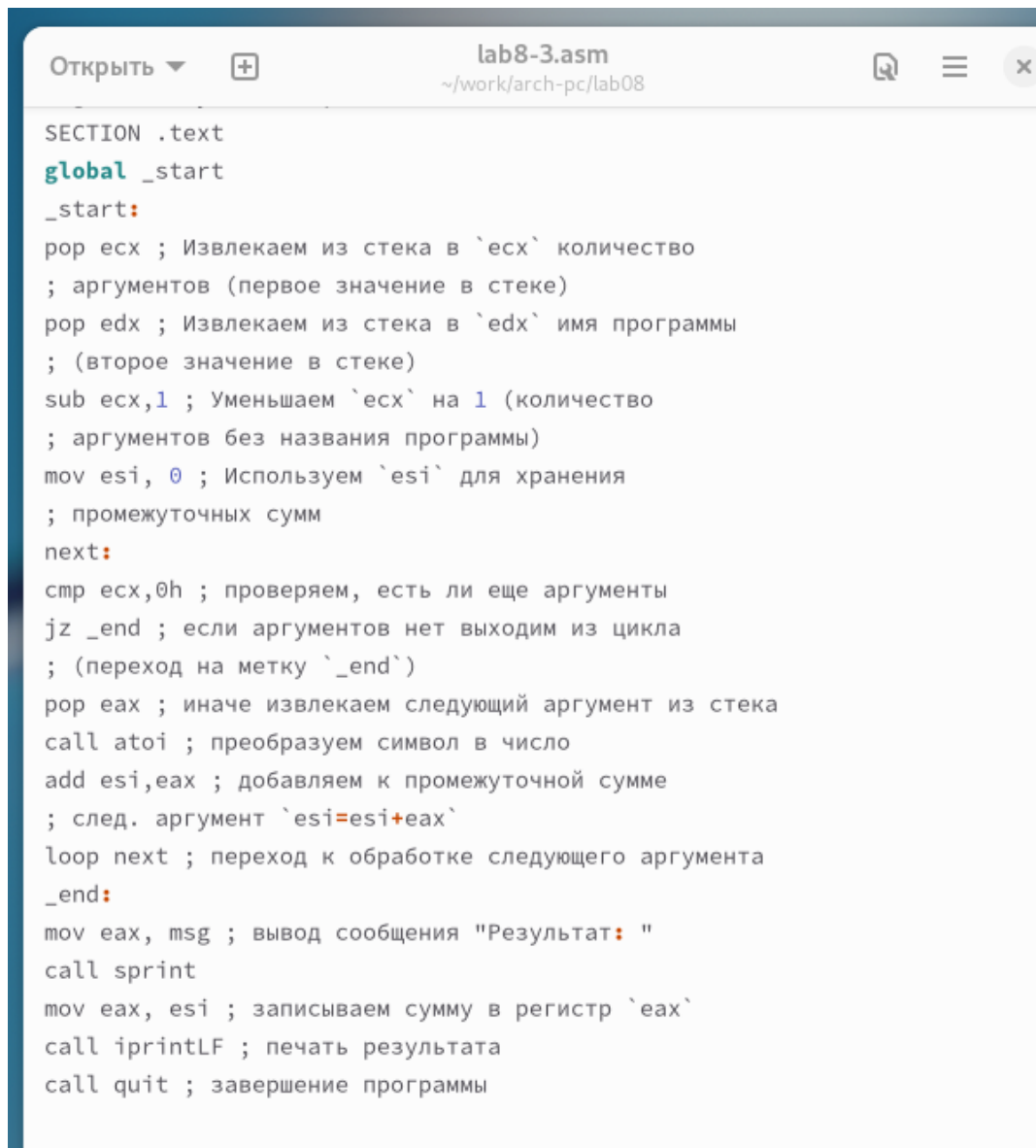
```
%include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
              ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
              ; (второе значение в стеке)
    sub ecx, 1 ; Уменьшаем `ecx` на 1 (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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
              ; аргумента (переход на метку `next`)
    _end:
    call quit
```

Рис. 2.8: Программа lab8-2.asm

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргуме  
нт 3'  
аргумент1  
аргумент  
2  
аргумент 3  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
```

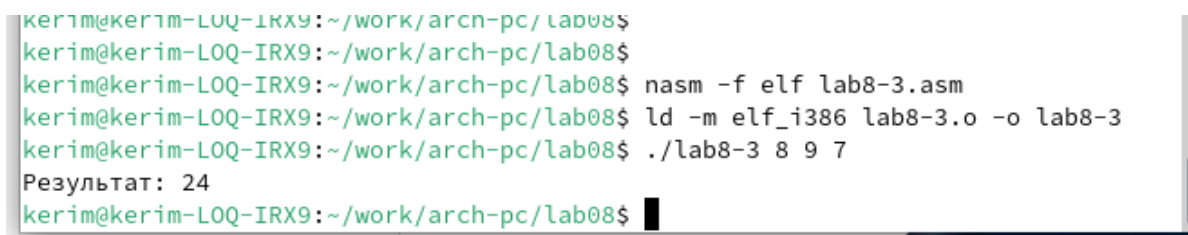
Рис. 2.9: Запуск программы lab8-2.asm

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



```
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

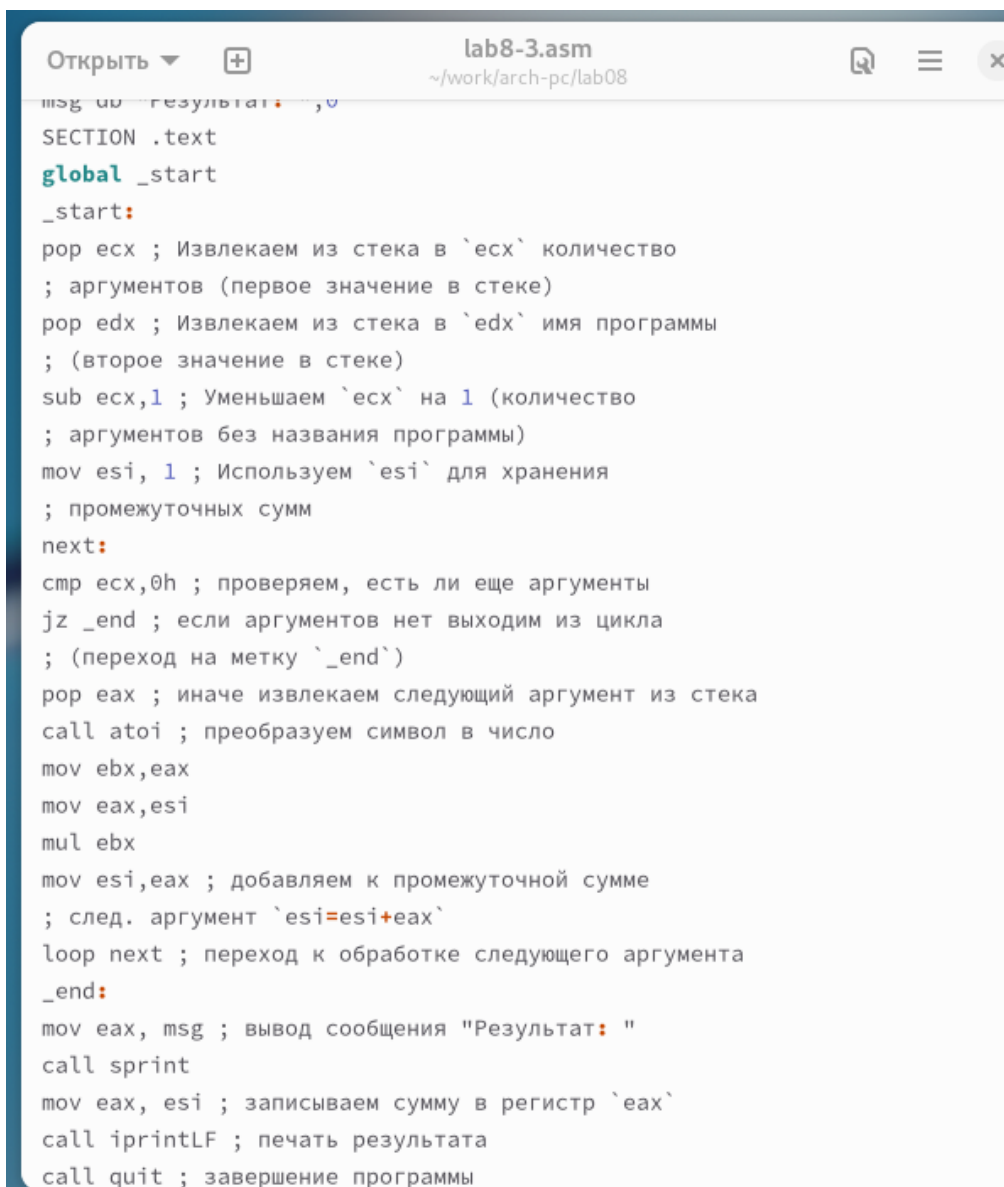


```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-3 8 9 7
Результат: 24
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
```

Рис. 2.11: Запуск программы lab8-3.asm

Изменил текст программы из листинга 8.3 для вычисления произведения аргу-

ментов командной строки (рис. 2.12, рис. 2.13).



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


```

kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-3 8 9 7
Результат: 24
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-3 8 9 7
Результат: 504
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ █

```

Рис. 2.13: Запуск программы lab8-3.asm

2.2 Самостоятельное задание

Написал 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 $f(x)$ выбирается в соответствии с вариантом, полученным при выполнении лабораторной работы № 7.

Создал исполняемый файл и проверил его работу на нескольких наборах x (рис. 2.14, рис. 2.15).

Для варианта 7:

$$f(x) = 3(x + 2)$$

```
Открыть ▾ + lab8-task1.asm  
~/.work/arch-pc/lab08  
fx: db 'f(x)= 3(x + 2)',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  
add eax,2  
mov ebx,3  
mul ebx  
add esi,eax  
  
loop next  
  
_end:  
mov eax, msg  
call sprint  
mov eax, esi  
call iprintLF
```

Рис. 2.14: Программа lab8-task1.asm

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ nasm -f elf lab8-task1.asm  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-task1.o -o lab8-task1  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-task1  
f(x)= 3(x + 2)  
Результат: 0  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-task1 1  
f(x)= 3(x + 2)  
Результат: 9  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-task1 0  
f(x)= 3(x + 2)  
Результат: 6  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$ ./lab8-task1 0 2 3 4 5 6  
f(x)= 3(x + 2)  
Результат: 96  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab08$
```

Рис. 2.15: Запуск программы lab8-task1.asm

Убедился, что программа считает правильно $f(0) = 6$, $f(1) = 9$.

3 Выводы

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