

Отчёт по лабораторной работе 8

Дисциплина: Архитектура компьютеров

Баранов Георгий Павлович

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
3.1	Неализация циклов в NASM	7
3.2	Обработка аргументов командной строки	13
3.3	Задание для самостоятельной работы	17
4	Выводы	20

Список иллюстраций

3.1	Программа в файле lab8-1.asm	8
3.2	Запуск программы lab8-1.asm	9
3.3	Программа в файле lab8-1.asm	10
3.4	Запуск программы lab8-1.asm	11
3.5	Программа в файле lab8-1.asm	12
3.6	Запуск программы lab8-1.asm	13
3.7	Программа в файле lab8-2.asm	14
3.8	Запуск программы lab8-2.asm	14
3.9	Программа в файле lab8-3.asm	15
3.10	Запуск программы lab8-3.asm	15
3.11	Программа в файле lab8-3.asm	16
3.12	Запуск программы lab8-3.asm	17
3.13	Программа в файле task.asm	18
3.14	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 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды.

Для стека существует две основные операции:

- добавление элемента в вершину стека (push);
- извлечение элемента из вершины стека (pop).

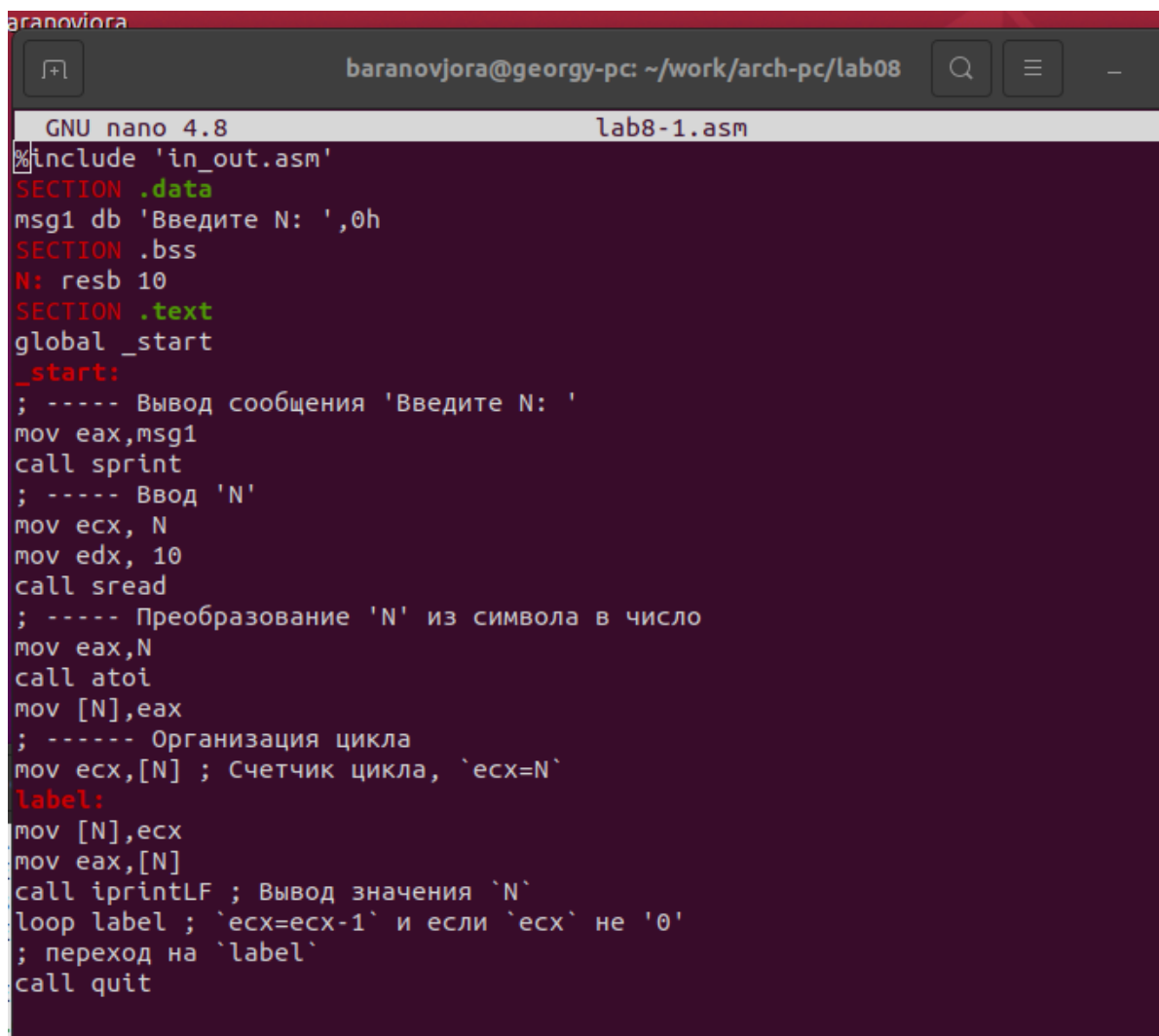
3 Выполнение лабораторной работы

3.1 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 `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. Создал исполняемый файл и проверил его работу.



```
baranovjora
baranovjora@georgy-pc: ~/work/arch-pc/lab08
GNU nano 4.8 lab8-1.asm
%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
```

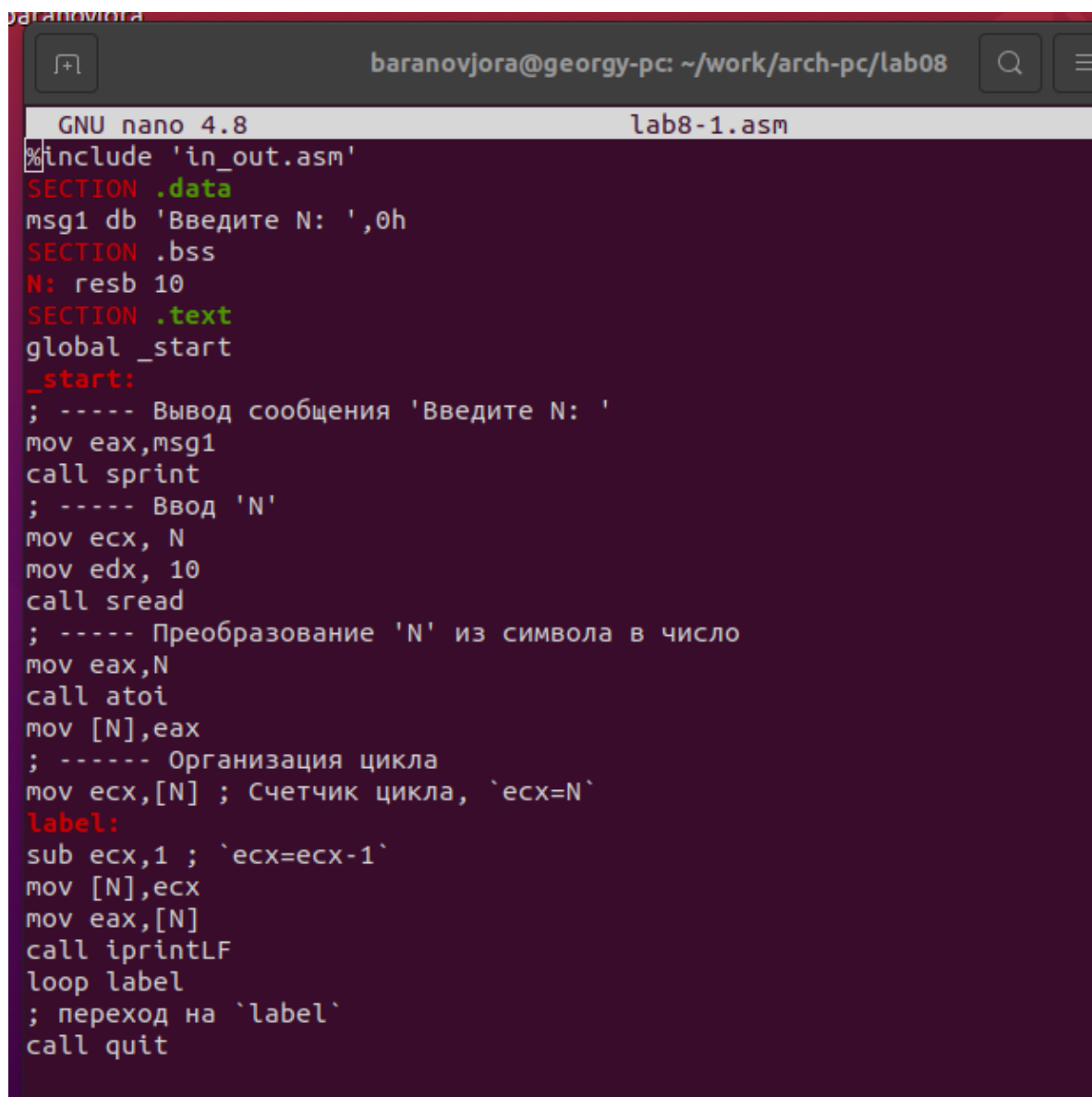
Рис. 3.1: Программа в файле lab8-1.asm


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

Рис. 3.2: 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`.



```
GNU nano 4.8 lab8-1.asm
%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. 3.3: Программа в файле lab8-1.asm

```
4294928794
4294928792
4294928790
4294928788
4294928786
4294928784
4294928782
4294928780
42949^C
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
baranovjora@georgy-pc:~/work/arch-pc/lab08$
```

Рис. 3.4: 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 .

```
baranovjora
baranovjora@georgy-pc: ~/work/arch-pc/lab08
GNU nano 4.8 lab8-1.asm
%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. 3.5: Программа в файле lab8-1.asm

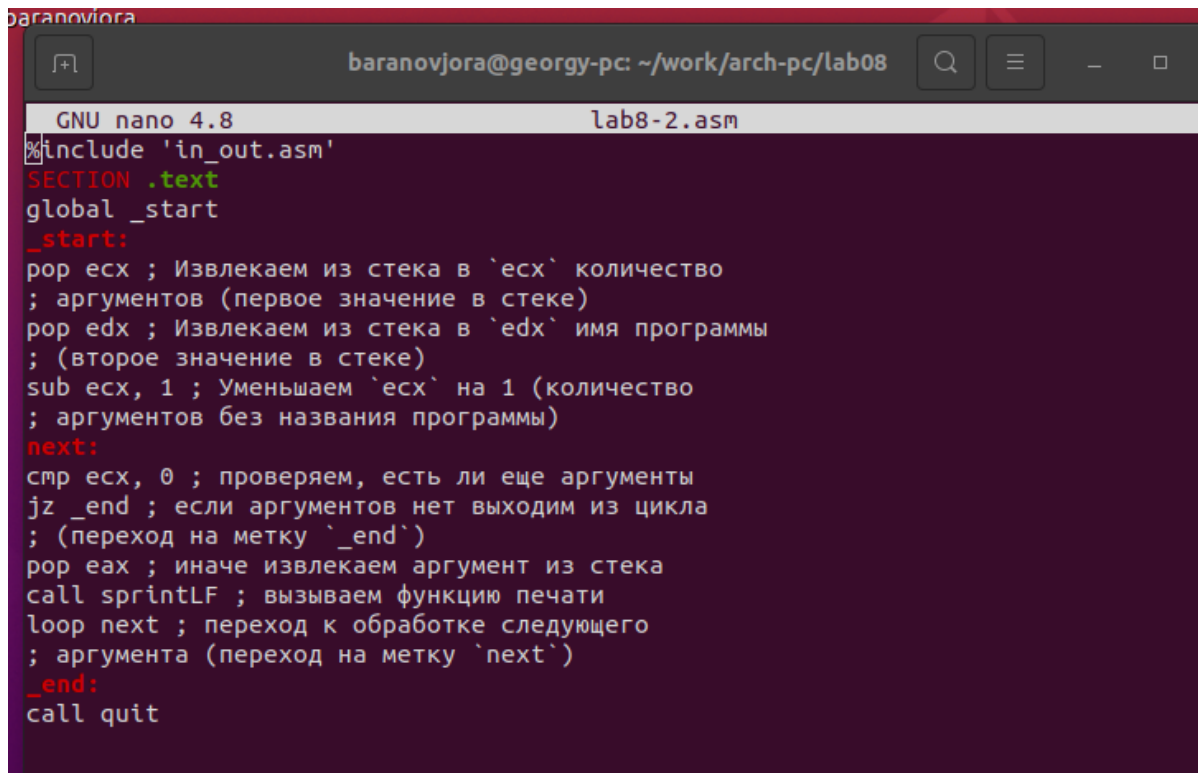
```
baranovjora@georgy-pc:~/work/arch-pc/lab08$  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 3  
2  
1  
0  
baranovjora@georgy-pc:~/work/arch-pc/lab08$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

Рис. 3.6: Запуск программы lab8-1.asm

3.2 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 4 аргумента. Аргументами считаются слова/числа, разделенные пробелом.



```
baranovjora@georgy-pc: ~/work/arch-pc/lab08
GNU nano 4.8 lab8-2.asm
%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 sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 3.7: Программа в файле lab8-2.asm



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

Рис. 3.8: Запуск программы lab8-2.asm

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

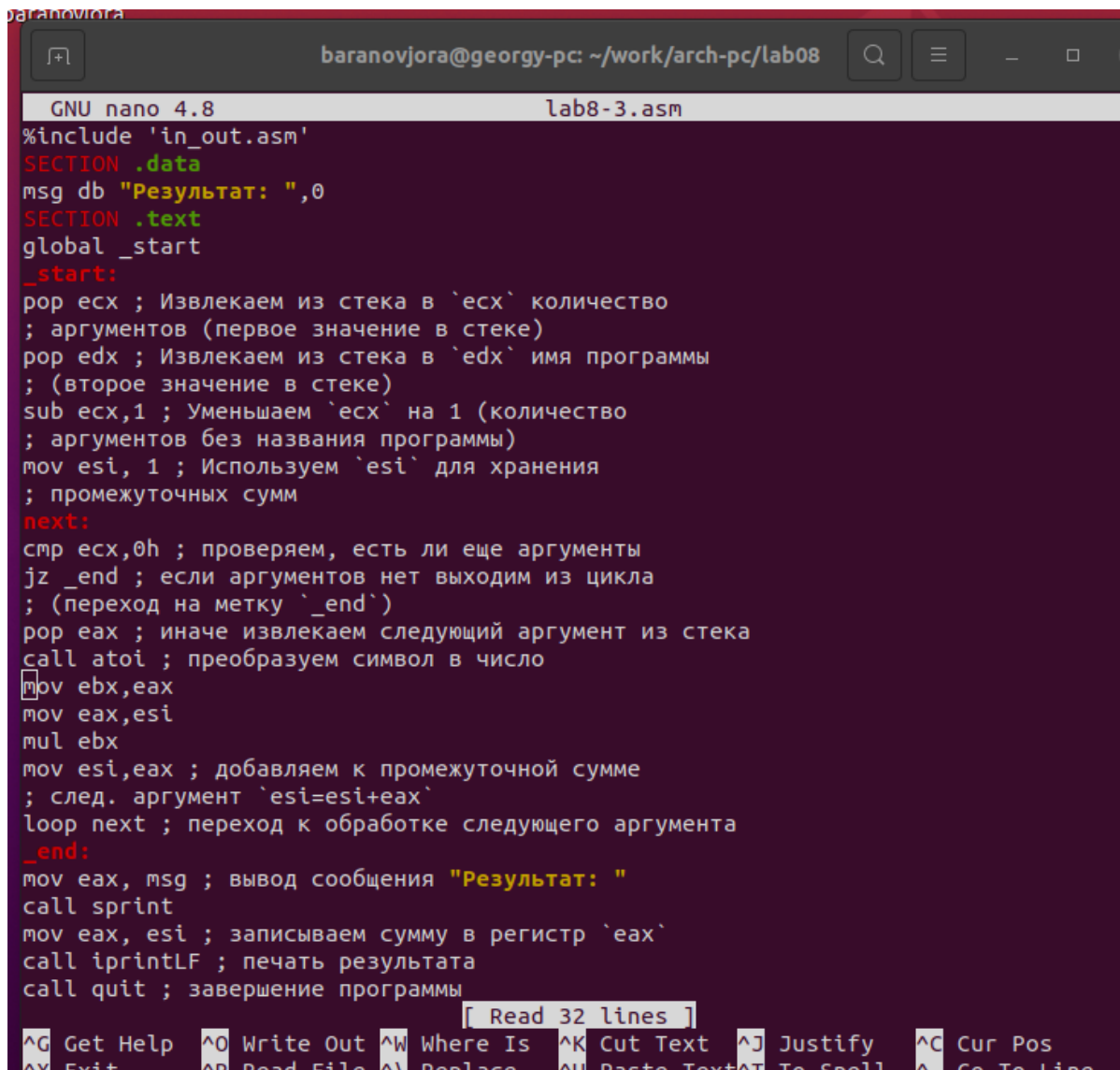
```
baranovjora
baranovjora@georgy-pc: ~/work/arch-pc/lab08
GNU nano 4.8 lab8-3.asm
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por edx ; Извлекаем из стека в `edx` 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`)
por 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. 3.9: Программа в файле lab8-3.asm

```
baranovjora@georgy-pc: ~/work/arch-pc/lab08
baranovjora@georgy-pc:~/work/arch-pc/lab08$
baranovjora@georgy-pc:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./lab8-3
Результат: 0
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./lab8-3 3 4 5 6
Результат: 18
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./lab8-3 7 8 9
Результат: 24
baranovjora@georgy-pc:~/work/arch-pc/lab08$
```

Рис. 3.10: Запуск программы lab8-3.asm

Изменл текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.



```
GNU nano 4.8 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 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
```

[Read 32 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^B Read File ^L Replace ^U Paste Text ^T To Spell ^_ Go To Line

Рис. 3.11: Программа в файле lab8-3.asm

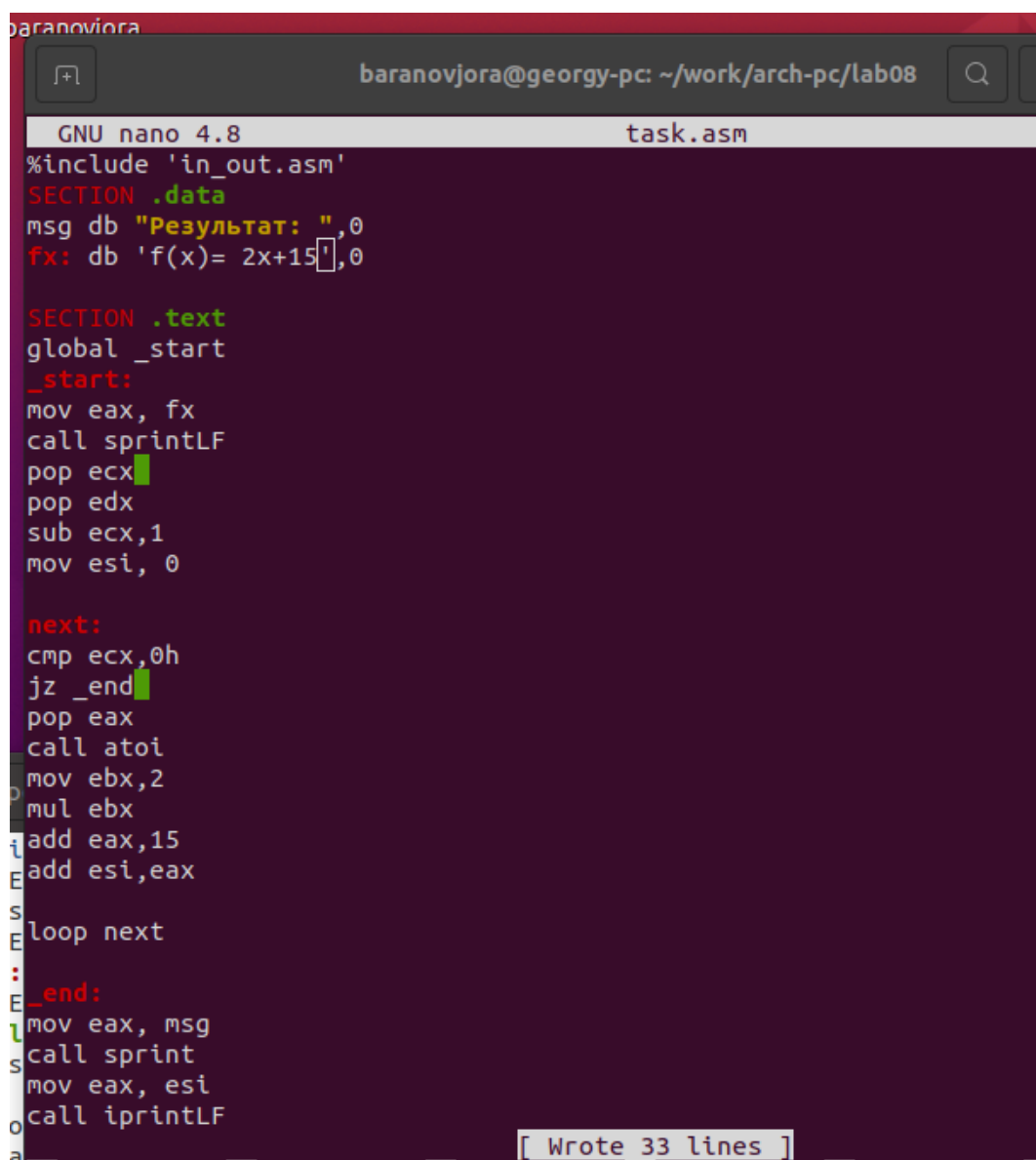

```
baranovjora@georgy-pc:~/work/arch-pc/lab08$  
baranovjora@georgy-pc:~/work/arch-pc/lab08$  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./lab8-3  
Результат: 1  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./lab8-3 3 4 5 6  
Результат: 360  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./lab8-3 7 8 9  
Результат: 504  
baranovjora@georgy-pc:~/work/arch-pc/lab08$  
baranovjora@georgy-pc:~/work/arch-pc/lab08$
```

Рис. 3.12: Запуск программы lab8-3.asm

3.3 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 1 $f(x) = 2x + 15$



```
baranovjora
baranovjora@georgy-pc: ~/work/arch-pc/lab08
GNU nano 4.8 task.asm
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 2x+15',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
mov ebx,2
mul ebx
add eax,15
add esi,eax
loop next

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

[ Wrote 33 lines ]
```

Рис. 3.13: Программа в файле task.asm

Для проверки я запустил сначала с одним аргументом. Так, при подстановке $f(0) = 15, f(1) = 17$ Затем подал несколько аргументов и получил сумму значений функции.

```
baranovjora@georgy-pc:~/work/arch-pc/lab08$  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ nasm -f elf task.asm  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ld -m elf_i386 task.o -o task  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./task  
f(x)= 2x+15  
Результат: 0  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./task 1  
f(x)= 2x+15  
Результат: 17  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./task 2  
f(x)= 2x+15  
Результат: 19  
baranovjora@georgy-pc:~/work/arch-pc/lab08$ ./task 2 3 4 6 7 9  
f(x)= 2x+15  
Результат: 152  
baranovjora@georgy-pc:~/work/arch-pc/lab08$
```

Рис. 3.14: Запуск программы task.asm

4 Выводы

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