

Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Курушин Георгий Романович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	14
4.3	Задание для самостоятельной работы	20
5	Выводы	24
6	Список литературы	25

Список иллюстраций

4.1	Создание каталога	8
4.2	Копирование программы из листинга	9
4.3	Запуск программы	10
4.4	Изменение программы	11
4.5	Запуск измененной программы	12
4.6	Добавление push и pop в цикл программы	13
4.7	Запуск измененной программы	14
4.8	Копирование программы из листинга	15
4.9	Запуск второй программы	16
4.10	Копирование программы из третьего листинга	17
4.11	Запуск третьей программы	18
4.12	Изменение третьей программы	19
4.13	Запуск измененной третьей программы	20
4.14	Написание программы для самостоятельной работы	21
4.15	Запуск программы для самостоятельной работы	23

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной работы

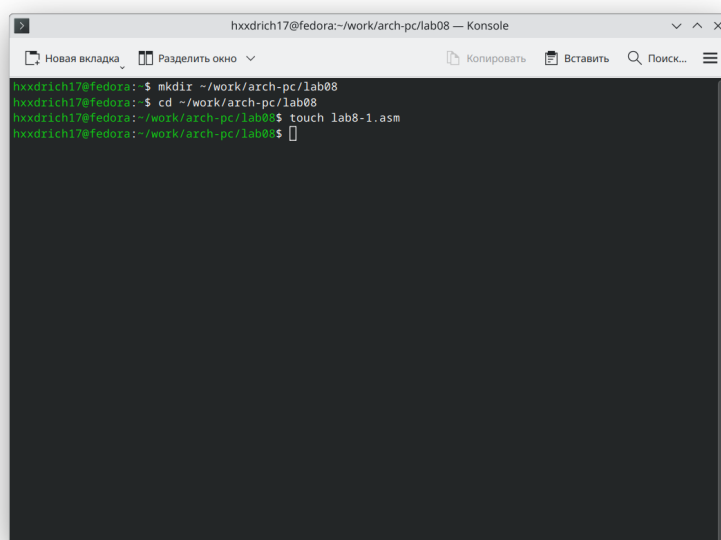
3 Теоретическое введение

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. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

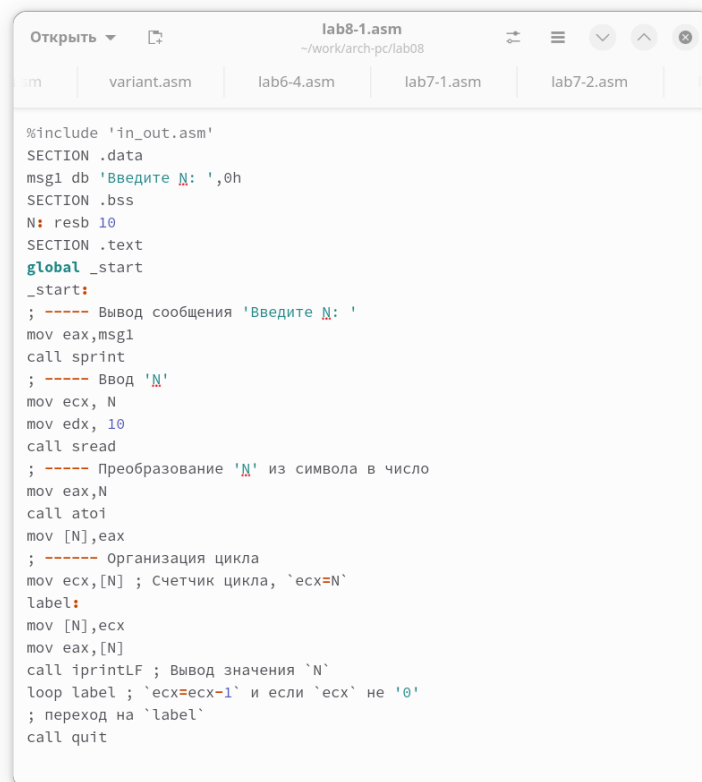
Создаю каталог для программ лабораторной работы №8 (рис. -fig. 4.1).



```
hxxdrich17@fedora:~/work/arch-pc/lab08 — Konsole
Новая вкладка  Разделить окно  Копировать  Вставить  Поиск...
hxxdrich17@fedora: $ mkdir ~/work/arch-pc/lab08
hxxdrich17@fedora: $ cd ~/work/arch-pc/lab08
hxxdrich17@fedora: ~/work/arch-pc/lab08$ touch lab8-1.asm
hxxdrich17@fedora: ~/work/arch-pc/lab08$
```

Рис. 4.1: Создание каталога

Копирую в созданный файл программу из листинга. (рис. -fig. 4.2).



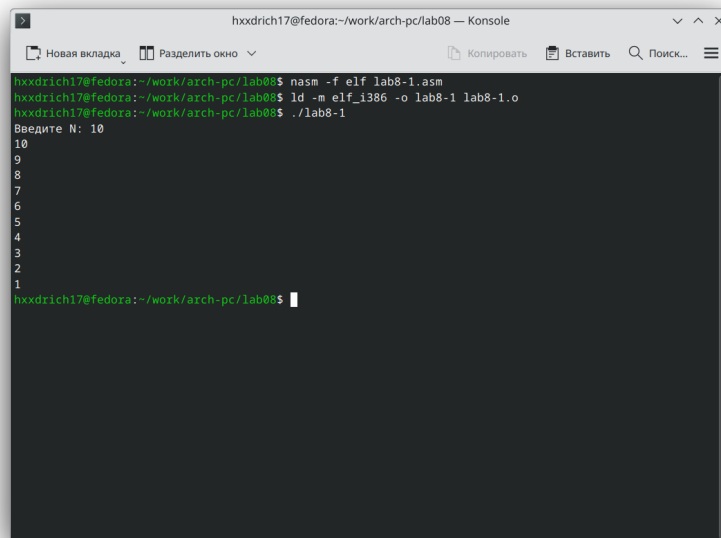
```
lab8-1.asm
~/work/arch-pc/lab08

asm | variant.asm | lab6-4.asm | lab7-1.asm | lab7-2.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. 4.2: Копирование программы из листинга

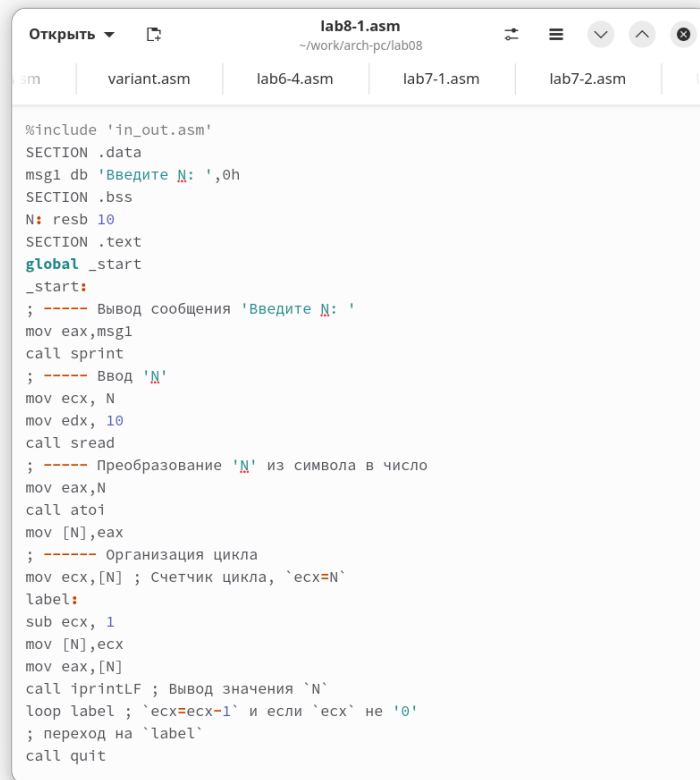
Запускаю программу, она показывает работу циклов в NASM (рис. -fig. 4.3).



```
hxxdrich17@fedora:~/work/arch-pc/lab08 — Konsole
Новая вкладка Разделить окно Копировать Вставить Поиск...
hxxdrich17@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
hxxdrich17@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
hxxdrich17@fedora:~/work/arch-pc/lab08$
```

Рис. 4.3: Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра `ecx` (рис. -fig. 4.4).

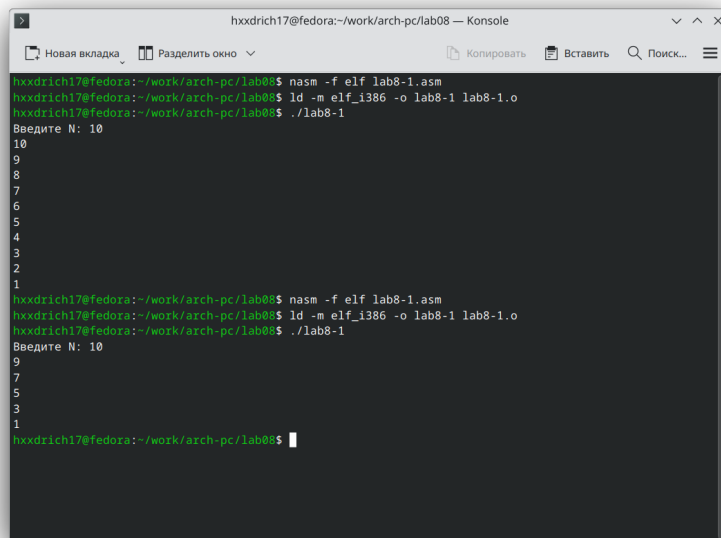


```
lab8-1.asm
~/work/arch-pc/lab08

%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
mov [N],ecx
mov eax,[N]
call iprintf ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit
```

Рис. 4.4: Изменение программы

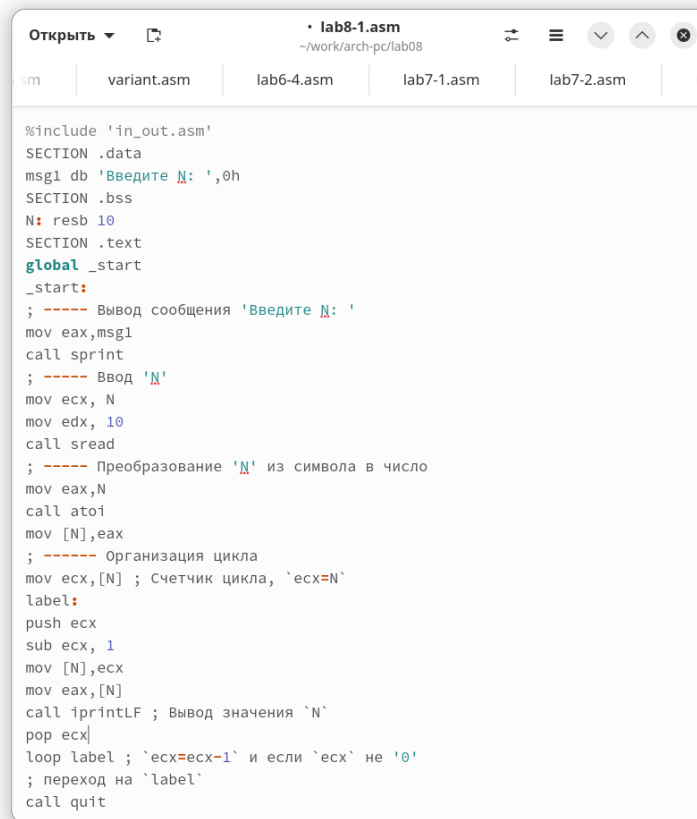
Из-за того, что теперь регистр `ecx` на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. -fig. 4.5).



```
hxxdrich17@fedora:~/work/arch-pc/lab08 — Konsole
hxxdrich17@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
hxxdrich17@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
hxxdrich17@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
hxxdrich17@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
hxxdrich17@fedora:~/work/arch-pc/lab08$
```

Рис. 4.5: Запуск измененной программы

Добавляю команды push и pop в программу (рис. -fig. 4.6).



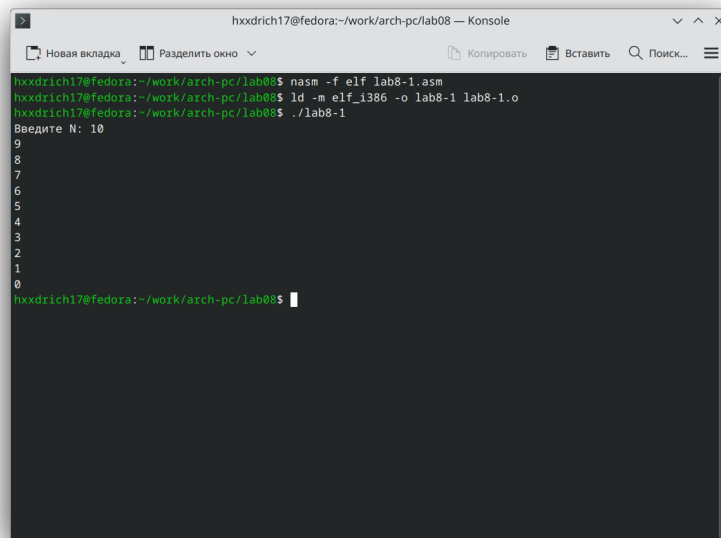
```
lab8-1.asm
~/work/arch-pc/lab08

asm | variant.asm | lab6-4.asm | lab7-1.asm | lab7-2.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
sub ecx, 1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
pop ecx
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Рис. 4.6: Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. -fig. 4.7).

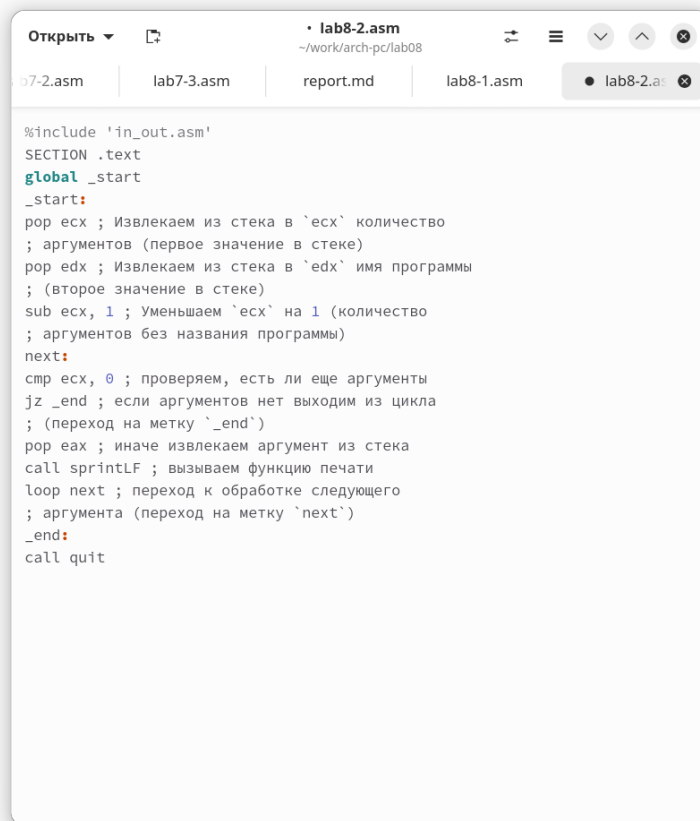


```
hxxdrich17@fedora:~/work/arch-pc/lab08 — Konsole
hxxdrich17@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
hxxdrich17@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
hxxdrich17@fedora:~/work/arch-pc/lab08$
```

Рис. 4.7: Запуск измененной программы

4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. -fig. 4.8).

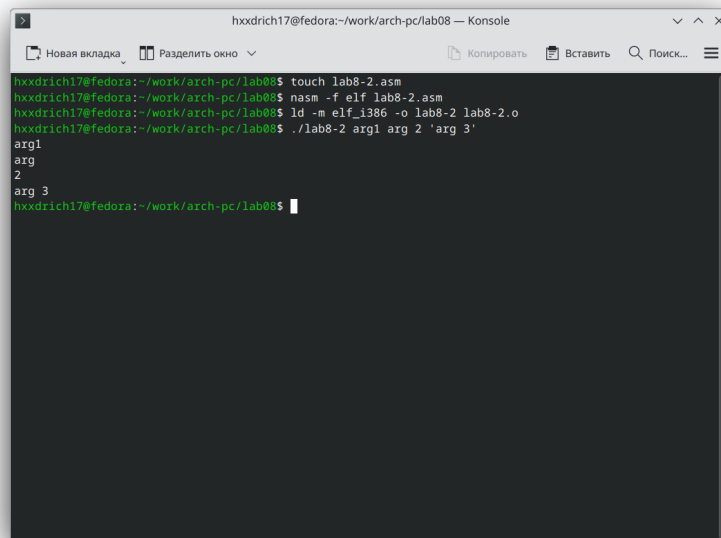


```
Открыть ▾ [icon] • lab8-2.asm
~/work/arch-pc/lab08
lab7-2.asm | lab7-3.asm | report.md | lab8-1.asm | ● lab8-2.asm [x]

#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. 4.8: Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было обработано то же количество аргументов, что и было введено (рис. -fig. 4.9).



```
hxxdrich17@fedora:~/work/arch-pc/lab08 — Konsole
hxxdrich17@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
hxxdrich17@fedora:~/work/arch-pc/lab08$ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
hxxdrich17@fedora:~/work/arch-pc/lab08$
```

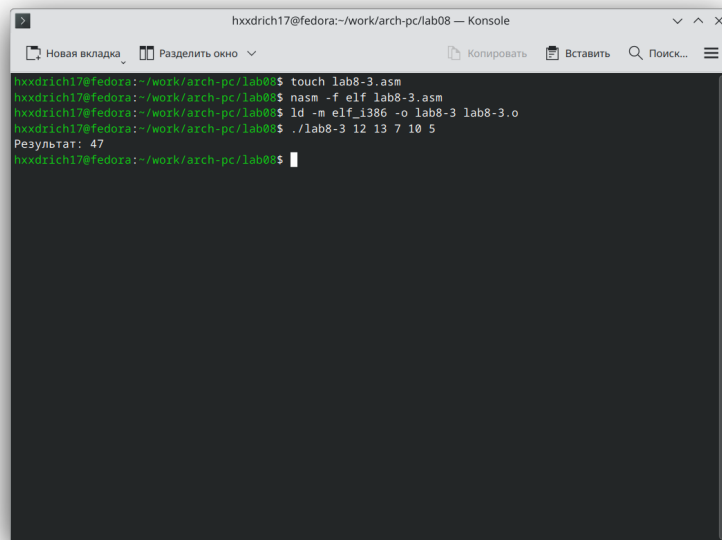
Рис. 4.9: Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. -fig. 4.10).


```
%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 iprintlnLF ; печать результата
    call quit ; завершение программы
```

Рис. 4.10: Копирование программы из третьего листинга

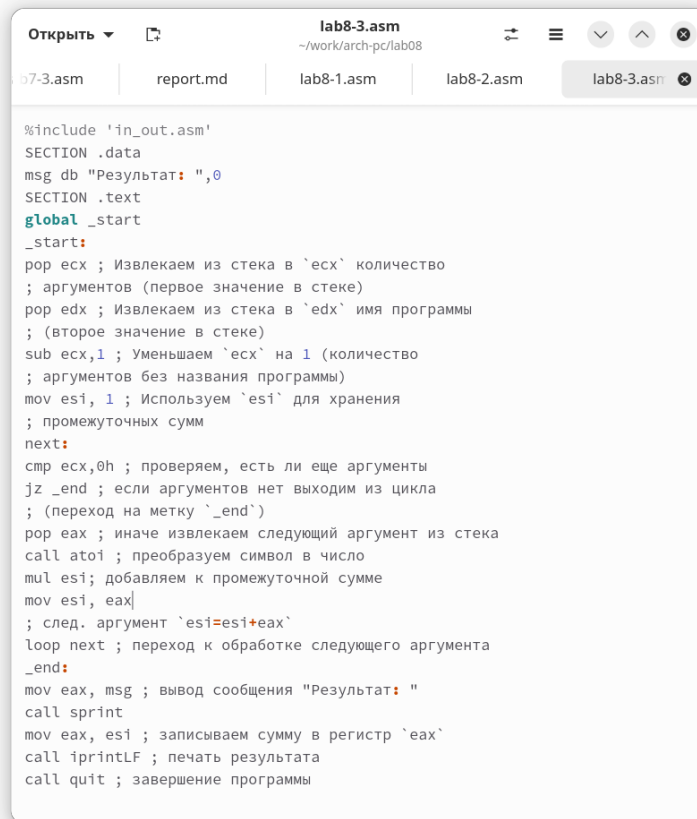
Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. -fig. 4.11).



```
hxxdrich17@fedora:~/work/arch-pc/lab08 — Konsole
hxxdrich17@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
hxxdrich17@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
hxxdrich17@fedora:~/work/arch-pc/lab08$
```

Рис. 4.11: Запуск третьей программы

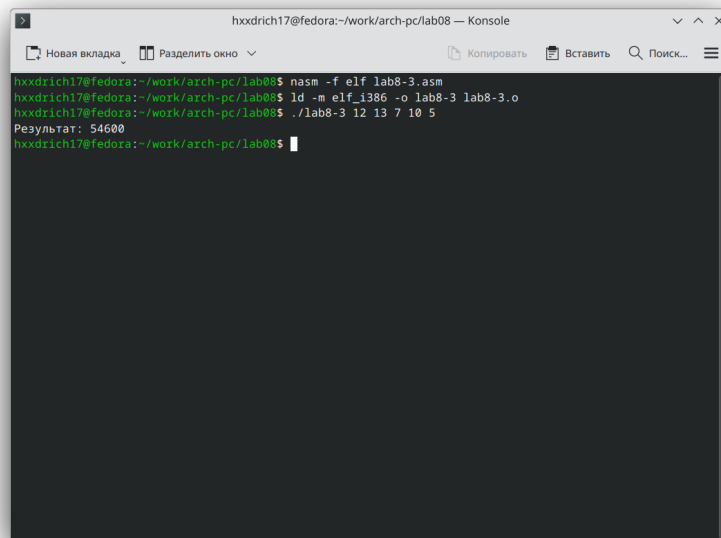
Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. -fig. 4.12).



```
%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
    mul esi ; добавляем к промежуточной сумме
    mov esi, eax
             ; 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 iprintf ; печать результата
    call quit ; завершение программы
```

Рис. 4.12: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. 4.13).



```
hxxdrich17@fedora:~/work/arch-pc/lab08 — Konsole
hxxdrich17@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
hxxdrich17@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
hxxdrich17@fedora:~/work/arch-pc/lab08$
```

Рис. 4.13: Запуск измененной третьей программы

4.3 Задание для самостоятельной работы

Пишу программму, которая будет находить сумма значений для функции $f(x) = 30x - 11$, которая совпадает с моим 16 вариантом (рис. -fig. 4.14).

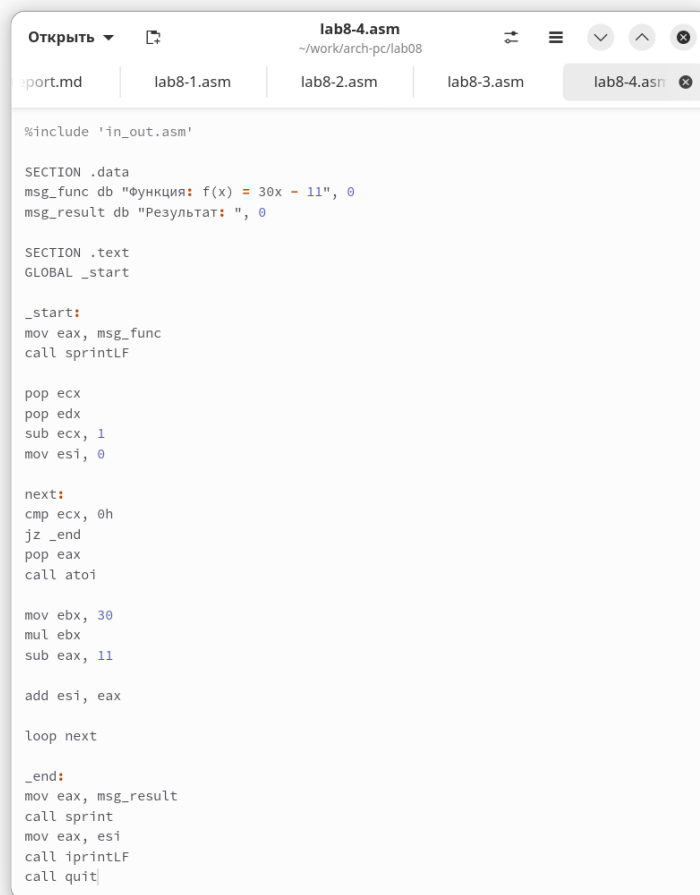


Рис. 4.14: Написание программы для самостоятельной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 30x - 11", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start
```

```

_start:
mov eax, msg_func
call sprintf

pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

mov ebx, 30
mul ebx
sub eax, 11

add esi, eax

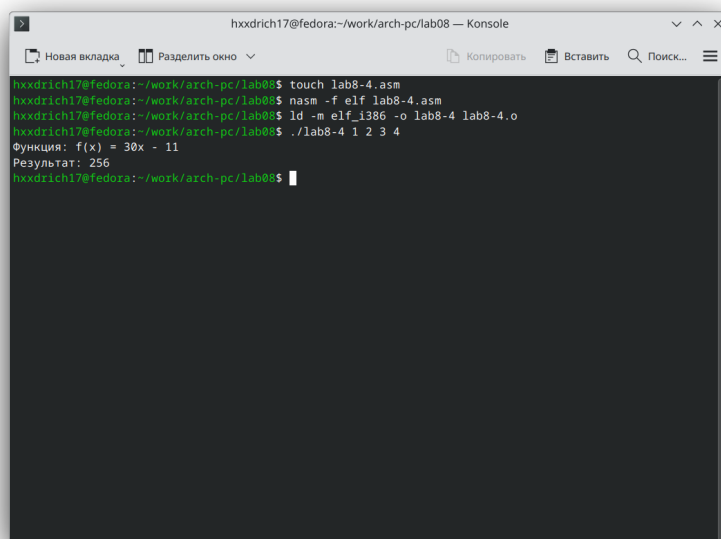
loop next

_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintLF

```

call quit

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. -fig. 4.15).



```
hxxdrich17@fedora:~/work/arch-pc/lab08 — Konsole
Новая вкладка  Разделить окно  Копировать  Вставить  Поиск...
hxxdrich17@fedora:~/work/arch-pc/lab08$ touch lab8-4.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
hxxdrich17@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
hxxdrich17@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: f(x) = 30x - 11
Результат: 256
hxxdrich17@fedora:~/work/arch-pc/lab08$
```

Рис. 4.15: Запуск программы для самостоятельной работы

5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №8
3. Программирование на языке ассемблера NASM Столяров А. В.