

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Самойлова Софья Дмитриевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Реализация циклов в NASM	7
4.2	Обработка аргументов командной строки	12
4.3	Выполнение самостоятельной работы	17
5	Выводы	20

Список иллюстраций

4.1	Создание каталога	7
4.2	Программа из листинга	8
4.3	Работа программы	8
4.4	Редактирование программы	9
4.5	Результат изменений программы	10
4.6	Редактирование программы	11
4.7	Работа программы	11
4.8	Редактирование программы	12
4.9	Запуск программы	13
4.10	Создание программы	14
4.11	Работа программы	15
4.12	Изменение программы	16
4.13	Работа программы	17
4.14	Работа программы	19

1 Цель работы

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

2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

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.

Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре *esp* (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции:

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

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

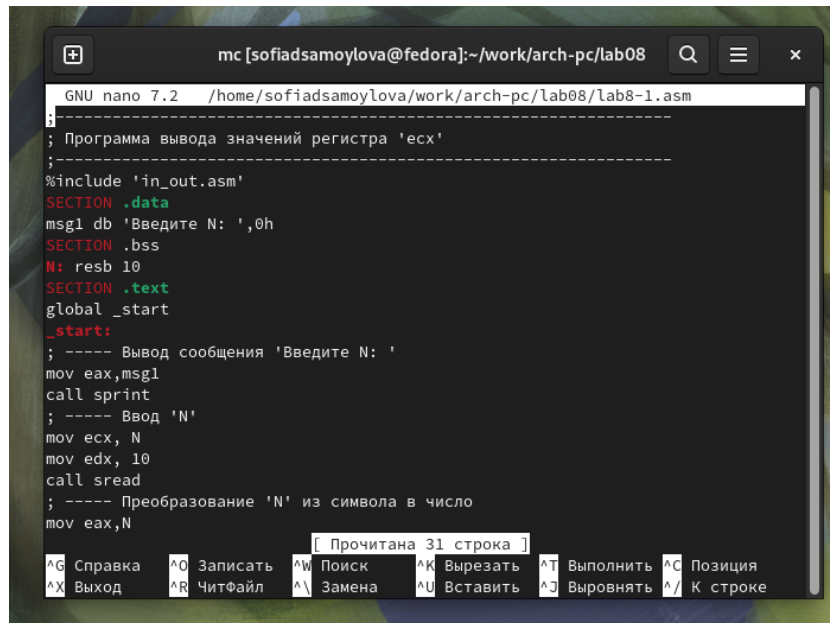
Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл `lab8-1.asm`, дополнительно копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах (рис. 4.1).

A screenshot of a terminal window with a dark background. The window title is "sofiadsamoylova@fedora:~/work/arch-pc/lab08". The terminal shows the following commands and their outputs:

```
sofiadsamoylova@fedora:~$ mkdir ~/work/arch-pc/lab08
sofiadsamoylova@fedora:~$ cd ~/work/arch-pc/lab08
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ cp ~/Загрузки/in_out.asm in_out.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.1: Создание каталога

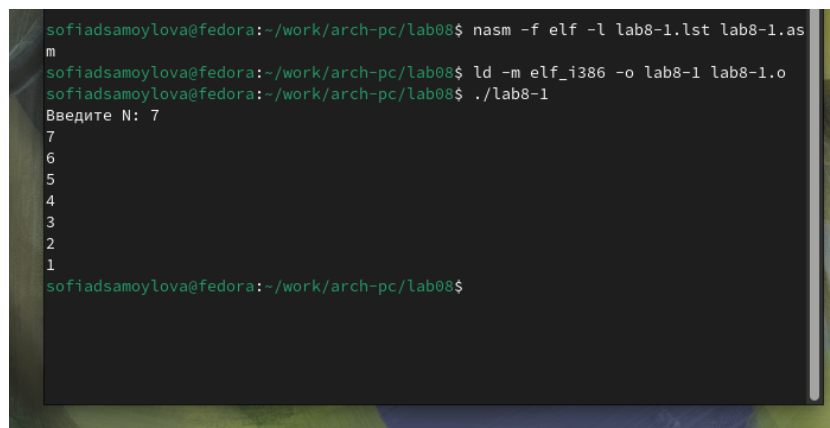
В качестве примера рассмотрю программу, которая выводит значение регистра `eax`. Ввожу в файл `lab8-1.asm` текст программы (рис. 4.2).



```
mc [sofiadsamoylova@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab08/lab8-1.asm
;
; 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
[ Прочитана 31 строка ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке
```

Рис. 4.2: Программа из листинга

Создаю исполняемый файл и проверяю его работу (рис. 4.3).



```
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ nasm -f elf -l lab8-1.lst lab8-1.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 7
7
6
5
4
3
2
1
sofiadsamoylova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.3: Работа программы

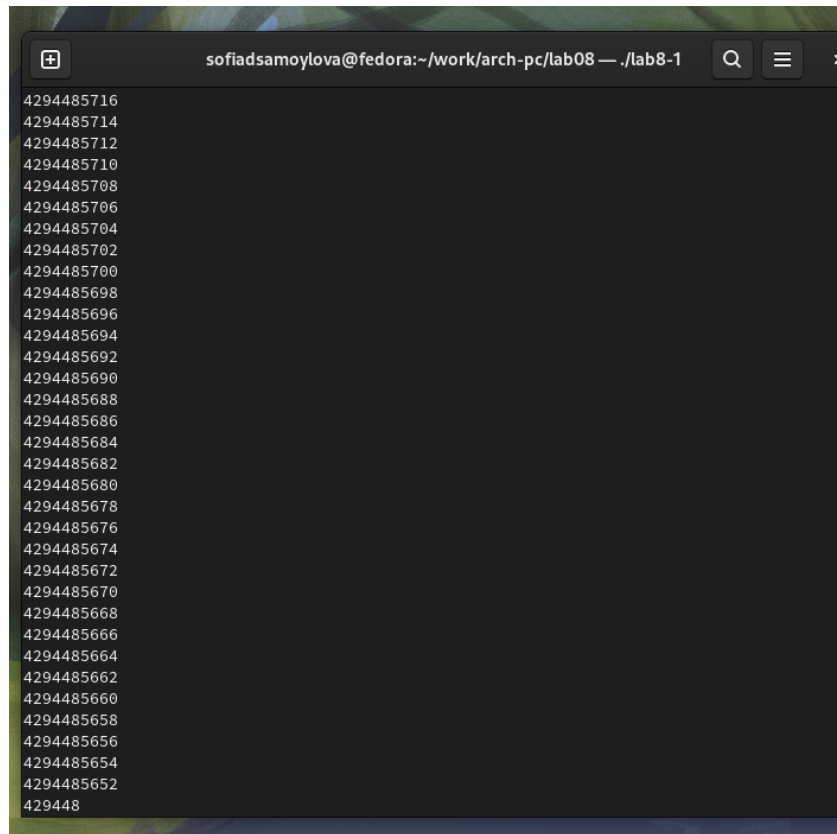
Программа показывает работу циклов в NASM.

Изменяю программу изначальную таким образом, что в теле цикла я изменяю значение регистра ecx (рис. 4.4).


```
mc [sofiadsamoylova@fedora]:~/work/arch-pc/lab08
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab08/lab8-1.asm
;-----
; 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:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
[ Прочитано 30 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выводить ^_ К строке
```

Рис. 4.4: Редактирование программы

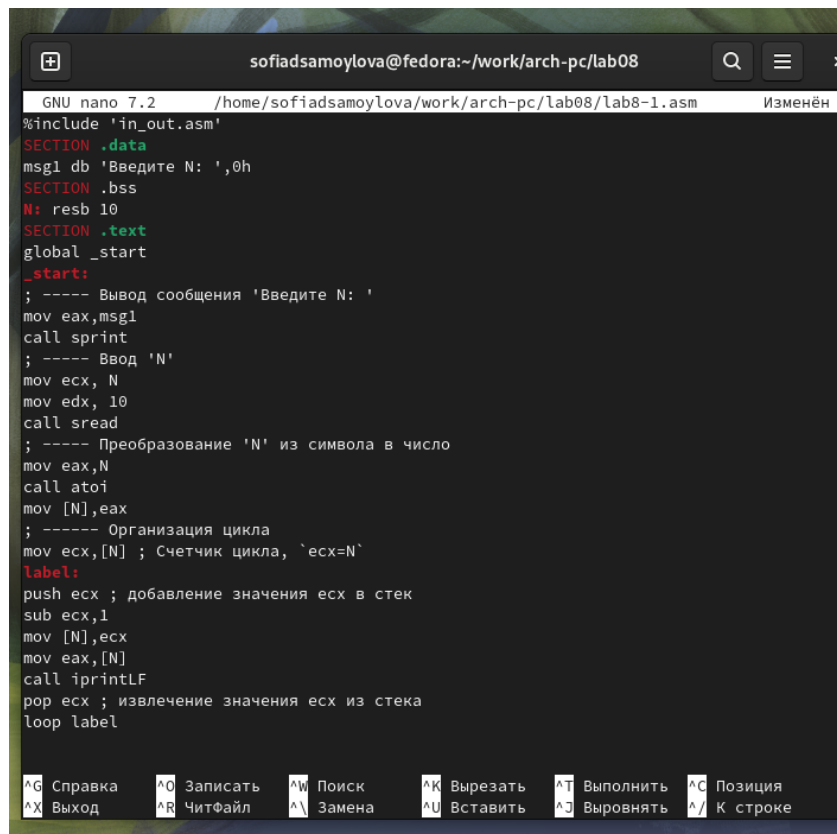
Как результат цикл программы стал бесконечным, несмотря на то, что значение для N было задано 5 (рис. 4.5).



```
sofiadsamoylova@fedora:~/work/arch-pc/lab08 — ./lab8-1
4294485716
4294485714
4294485712
4294485710
4294485708
4294485706
4294485704
4294485702
4294485700
4294485698
4294485696
4294485694
4294485692
4294485690
4294485688
4294485686
4294485684
4294485682
4294485680
4294485678
4294485676
4294485674
4294485672
4294485670
4294485668
4294485666
4294485664
4294485662
4294485660
4294485658
4294485656
4294485654
4294485652
429448
```

Рис. 4.5: Результат изменений программы

Для использования регистра `ecx` в цикле и сохранения корректности работы программы попробую использовать стек. Вношу изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`(рис. 4.6).

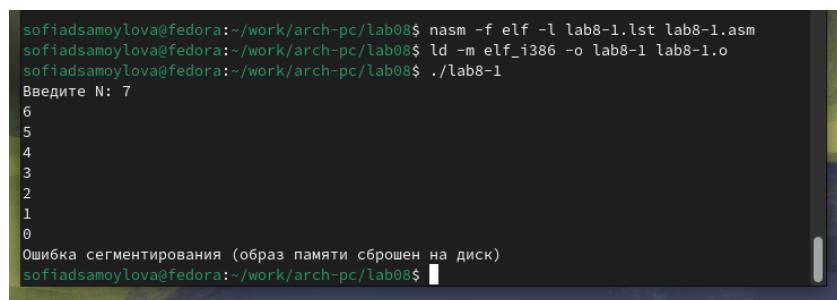


```
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab08/lab8-1.asm
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- 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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/_ К строке

Рис. 4.6: Редактирование программы

Создаю исполняемый файл и проверяю его работу (рис. 4.7).



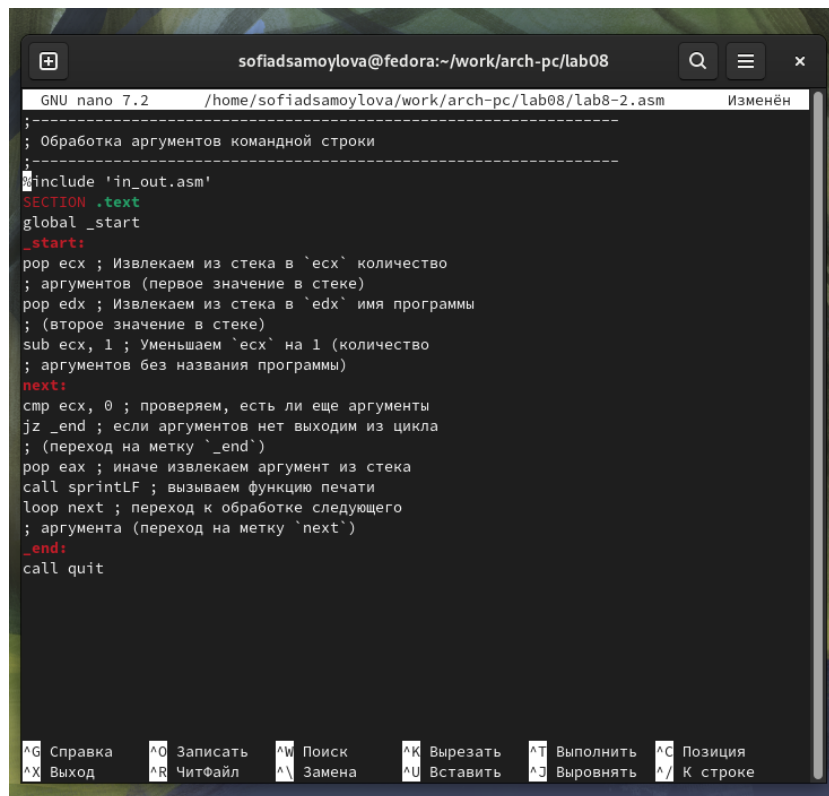
```
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ nasm -f elf -l lab8-1.lst lab8-1.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 7
6
5
4
3
2
1
0
Ошибка сегментирования (образ памяти сброшен на диск)
sofiadsamoylova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.7: Работа программы

Число проходов по циклу соответствует значению N, с учетом 0.

4.2 Обработка аргументов командной строки

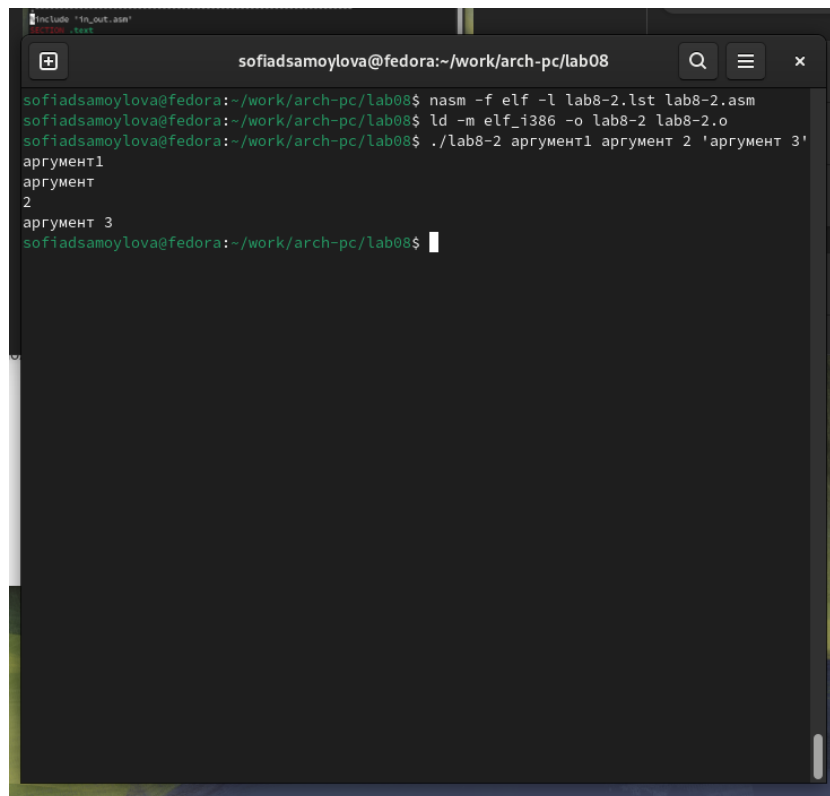
Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы (рис. 4.8).



```
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab08/lab8-2.asm Изменён
;
;-----
; 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 sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 4.8: Редактирование программы

Запускаю его, указав аргументы (рис. 4.9).

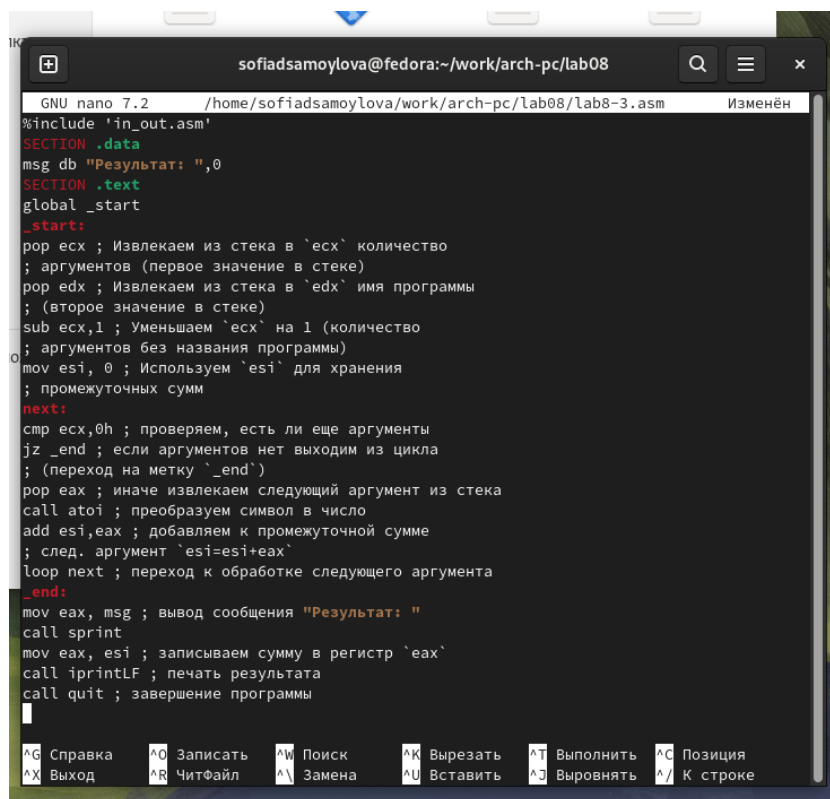
A terminal window titled 'sofiadsamoylova@fedora:~/work/arch-pc/lab08'. The terminal shows the following commands and output:

```
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ nasm -f elf -l lab8-2.lst lab8-2.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
sofiadsamoylova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.9: Запуск программы

Все аргументы были обработаны программой.

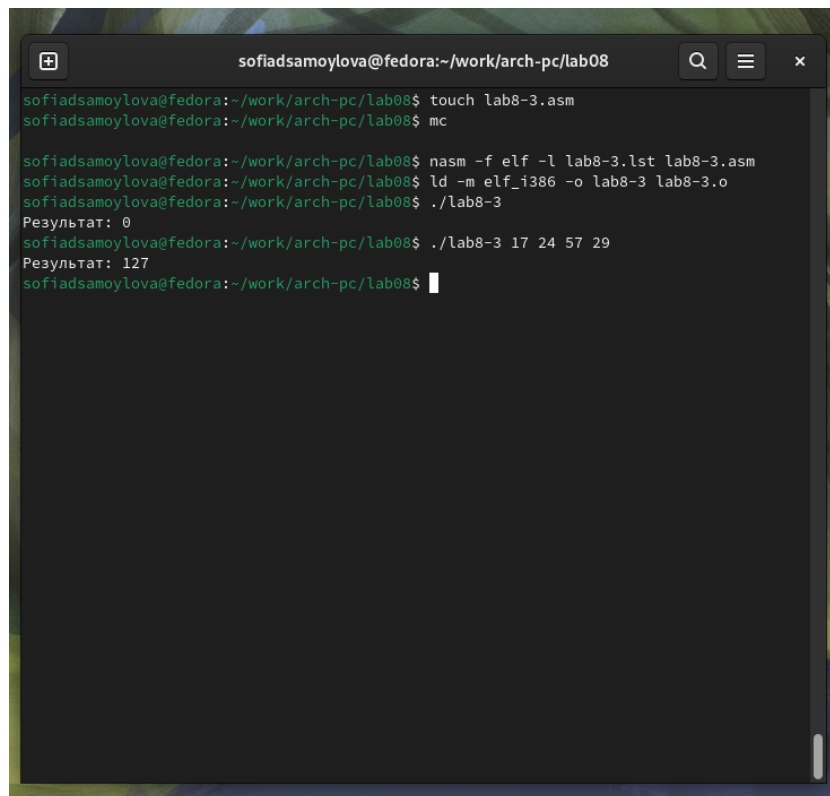
Создаю файл lab8-3.asm в каталоге ~/work/archpc/lab08 и ввожу в него текст программы, которая выводит сумму чисел, которые передаются в программу как аргументы (рис. 4.10).



```
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab08/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, 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. 4.10: Создание программы

Создаю исполняемый файл и запускаю его, указав аргументы (рис. 4.11).

A terminal window titled 'sofiadsamoylova@fedora:~/work/arch-pc/lab08' with search, menu, and close icons. It shows a series of commands and their outputs: 'touch lab8-3.asm', 'mc', 'nasm -f elf -l lab8-3.lst lab8-3.asm', 'ld -m elf_i386 -o lab8-3 lab8-3.o', and './lab8-3'. The output for the last command is 'Результат: 0'. A second execution of './lab8-3' with arguments '17 24 57 29' results in 'Результат: 127'.

```
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ mc

sofiadsamoylova@fedora:~/work/arch-pc/lab08$ nasm -f elf -l lab8-3.lst lab8-3.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ./lab8-3
Результат: 0
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ./lab8-3 17 24 57 29
Результат: 127
sofiadsamoylova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.11: Работа программы

Изменяю текст программы для вычисления произведения аргументов командной строки (рис. 4.12).

```
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab08/lab8-3.asm Изменён
#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 промежуточного произведения, начинаем с 1

next:
    cmp ecx, 0h ; Проверяем, есть ли еще аргументы
    jz _end ; Если аргументов нет, выходим из цикла (переход на метку _end)

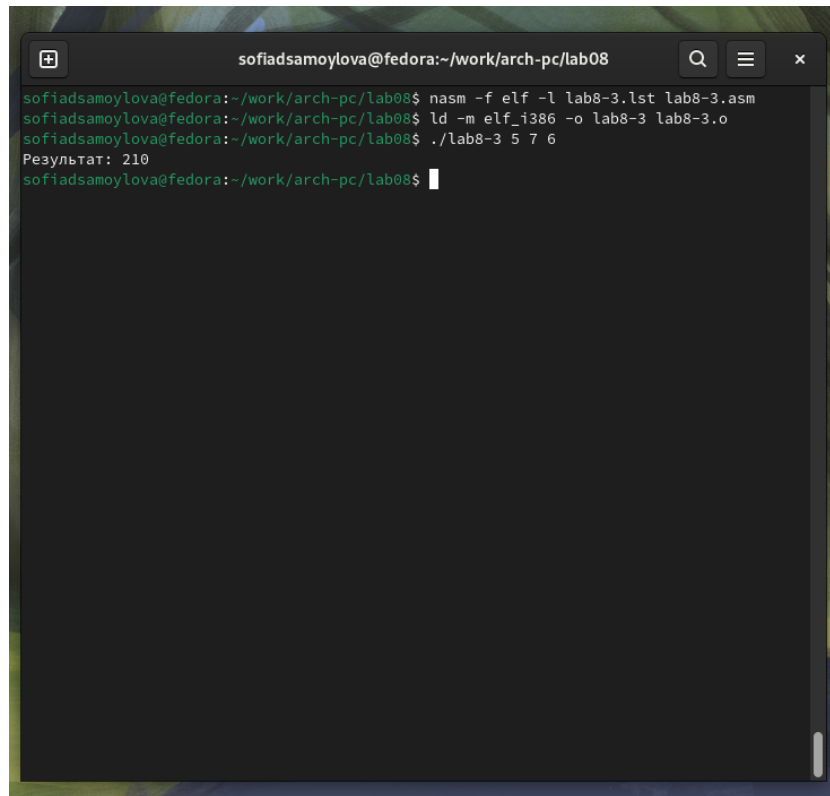
    pop eax ; Иначе извлекаем следующий аргумент из стека
    call atoi ; Преобразуем символ в число
    imul esi, eax ; Умножаем промежуточное произведение на текущее значение
    ; След. аргумент esi = esi * eax
    loop next ; Переход к обработке следующего аргумента

_end:
    mov eax, msg ; Вывод сообщения "Результат: "
    call sprint
    mov eax, esi ; Записываем произведение в регистр eax
    call iprintfLF ; Печать результата
    call quit ; Завершение программы
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^D Выровнять ^_/ К строке

Рис. 4.12: Изменение программы

Проверяю работу программы (рис. 4.13).

A screenshot of a terminal window with a dark background. The window title is 'sofiadsamoylova@fedora:~/work/arch-pc/lab08'. The terminal shows the following commands and output:

```
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ nasm -f elf -l lab8-3.lst lab8-3.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ./lab8-3 5 7 6
Результат: 210
sofiadsamoylova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.13: Работа программы

4.3 Выполнение самостоятельной работы

Вариант задания номер 17. Код программы:

```
%include 'in_out.asm'
SECTION .data
msg_func db "Функция:  $f(x) = 10(x - 1)$ ", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
    mov eax, msg_func
    call sprintf
    pop ecx           ; Извлекаем количество аргументов
```

```

    pop edx          ; Извлекаем имя программы
    sub ecx, 1       ; Уменьшаем на 1, чтобы исключить имя программы
    mov esi, 0       ; Инициализируем сумму (esi) нулем

next:
    cmp ecx, 0h      ; Проверяем, есть ли еще аргументы
    jz _end          ; Если нет, переходим к окончанию

    pop eax          ; Извлекаем следующий аргумент
    call atoi        ; Преобразуем строку в число

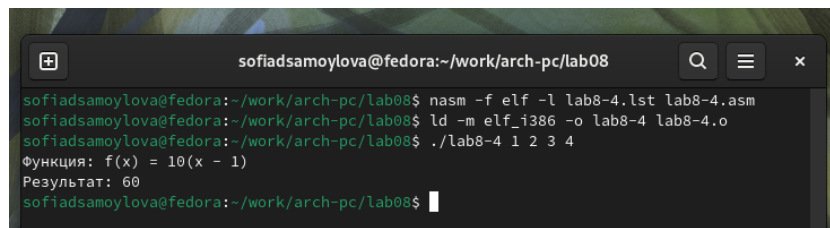
    mov ebx, eax      ; Сохраняем значение x в ebx
    sub ebx, 1        ; Вычисляем (x - 1)
    mov eax, 10       ; Загружаем 10 в eax
    imul eax, ebx     ; Умножаем 10 на (x - 1)

    add esi, eax      ; Добавляем результат функции к сумме

    loop next        ; Переход к следующему аргументу
_end:
    mov eax, msg_result
    call sprint       ; Выводим сообщение "Результат: "
    mov eax, esi      ; Загружаем сумму в eax
    call iprintLF     ; Печатаем результат
    call quit         ; Завершаем программу

```

Результат работы программы (рис. 4.14).



```
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ nasm -f elf -l lab8-4.lst lab8-4.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
sofiadsamoylova@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция:  $f(x) = 10(x - 1)$ 
Результат: 60
sofiadsamoylova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.14: Работа программы

5 Выводы

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