

Отчёт по лабораторной работе №8

дисциплина: Архитектура компьютера

Баранова Анна Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	11
4.3	Задание для самостоятельной работы	16
5	Выводы	19

List of Figures

4.1	Создание каталога для программ лабораторной работы № 8 и создание в нём файла lab8-1.asm	8
4.2	Копирование файла in_out.asm в каталог с файлом lab8-1.asm с помощью функциональной клавиши F5	8
4.3	Изменения в файле lab8-1.asm	9
4.4	Создание исполняемого файла и его запуск	9
4.5	Создание исполняемого файла и его запуск	10
4.6	Изменения в файле lab8-1.asm	10
4.7	Создание исполняемого файла и его запуск	10
4.8	Изменения в файле lab8-1.asm	11
4.9	Создание исполняемого файла и его запуск	11
4.10	Создание файла lab8-2.asm	11
4.11	Создание файла lab8-2.asm	12
4.12	Изменения в файле lab8-2.asm	12
4.13	Создание исполняемого файла и его запуск	13
4.14	Создание исполняемого файла и его запуск	13
4.15	Создание файла lab8-3.asm	13
4.16	Создание файла lab8-3.asm	14
4.17	Изменения в файле lab8-3.asm	14
4.18	Создание исполняемого файла и его запуск	15
4.19	Создание исполняемого файла и его запуск	15
4.20	Изменения в файле lab8-3.asm	16
4.21	Создание исполняемого файла и его запуск	16
4.22	Создание файла lab8-4.asm	16
4.23	Создание файла lab8-4.asm	17
4.24	Написание программы в lab8-4.asm	17
4.25	Создание исполняемого файла и его запуск с указанными аргументами	18

1 Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

В ходе выполнения данной лабораторной работы необходимо изучить:

- Организацию стека;
- реализацию циклов в NASM;
- обработку аргументов командной строки.

Выполнив эту работу, мы приобретём навыки написания программ с использованием циклов и обработкой аргументов командной строки.

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.

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре еsx. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл.

Инструкция loop выполняется в два этапа. Сначала из регистра еsx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы.

При запуске программы в NASM аргументы командной строки загружаются в стек в обратном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы,

скомпилированной NASM, – это всегда имя программы и количество переданных аргументов.

Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создадим каталог для программ лабораторной работы № 8, перейдём в него и создадим файл lab8-1.asm и также создадим копию файла in_out.asm (рис. 4.1), (рис. 4.2).

```
aabaranova@dk8n63 ~ $ mkdir ~/work/arch-pc/lab08
aabaranova@dk8n63 ~ $ cd ~/work/arch-pc/lab08
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 4.1: Создание каталога для программ лабораторной работы № 8 и создание в нём файла lab8-1.asm

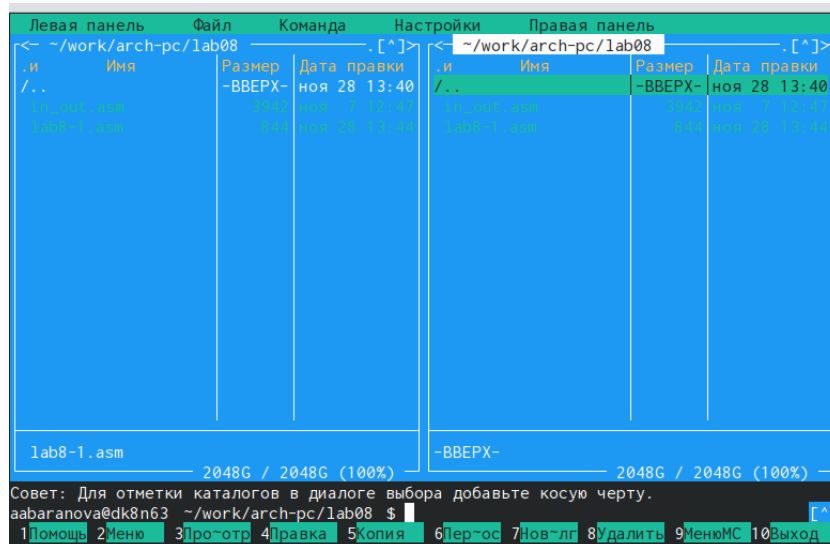


Рис. 4.2: Копирование файла in_out.asm в каталог с файлом lab8-1.asm с помощью функциональной клавиши F5

Введём в файл lab8-1.asm текст программы (рис. 4.3).


```

lab8-1.asm      [-M--] 13 L: [ 1+ 9 10/ 31] *(327 / 844b) 0010 0x00A      [*][X]
; 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'
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Удалить 9МенюМС 10Выход

```

Рис. 4.3: Изменения в файле lab8-1.asm

Создадим исполняемый файл и запустим его (рис. 4.4), (рис. 4.5).



Рис. 4.4: Создание исполняемого файла и его запуск

```

aabaranova@dk8n63 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
6
5
4
3
2
1

```

Рис. 4.5: Создание исполняемого файла и его запуск

Изменим текст программы файла lab8-1.asm (рис. 4.6).

```

lab8-1.asm [-M--] 11 L:[ 8+19 27/ 31] *(716 / 763b) 0010 0x00A [*][X]
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
call quit
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Удалить 9МенюМС 10Выход

```

Рис. 4.6: Изменения в файле lab8-1.asm

Создадим исполняемый файл и запустим его (рис. 4.7).

```

aabaranova@dk8n63 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
5
3
1

```

Рис. 4.7: Создание исполняемого файла и его запуск

Снова изменим текст программы файла lab8-1.asm, создадим исполняемый файл и запустим его (рис. 4.8), (рис. 4.9).

```
lab8-1.asm [-M--] 9 L: [ 10+23 33/ 33] *(882 / 882b) <EOF> [*][X]
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. 4.8: Изменения в файле lab8-1.asm

```
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
5
4
3
2
1
0
```

Рис. 4.9: Создание исполняемого файла и его запуск

4.2 Обработка аргументов командной строки

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (рис. 4.10), (рис. 4.11).

```
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ touch lab8-2.asm
aabaranova@dk8n63 ~/work/arch-pc/lab08 $
```

Рис. 4.10: Создание файла lab8-2.asm

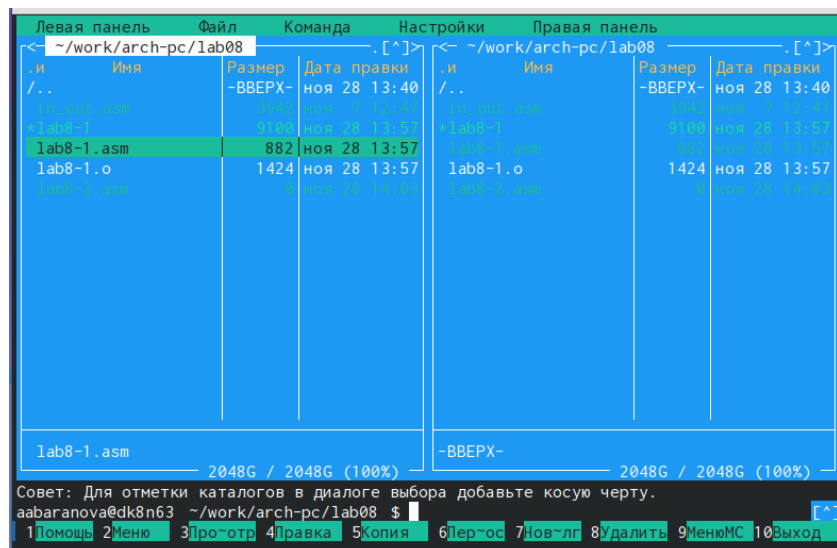


Рис. 4.11: Создание файла lab8-2.asm

Введём в файл lab8-2.asm текст программы (рис. 4.12).

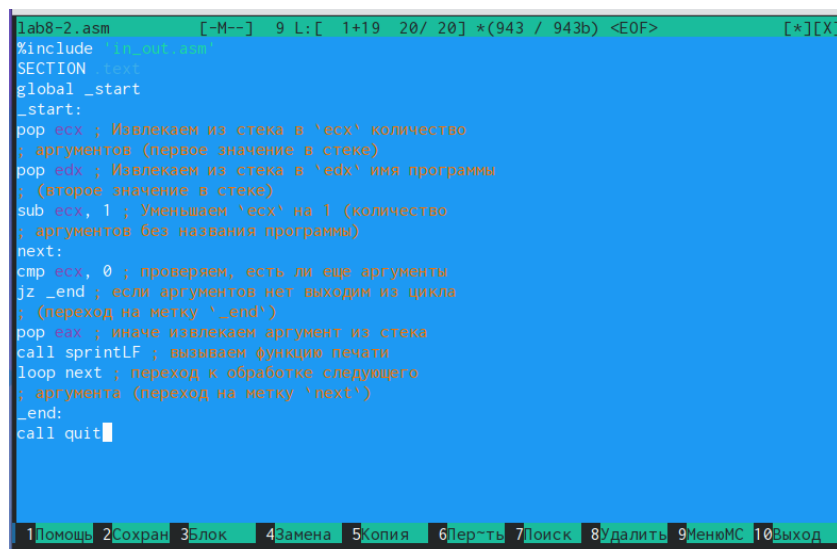


Рис. 4.12: Изменения в файле lab8-2.asm

Создадим исполняемый файл и запустим его, указав аргументы (рис. 4.13), (рис. 4.14).



Рис. 4.13: Создание исполняемого файла и его запуск

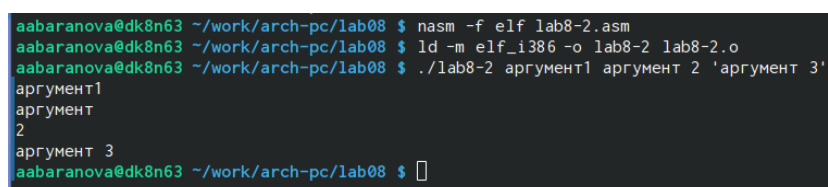


Рис. 4.14: Создание исполняемого файла и его запуск

Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. 4.15), (рис. 4.16).

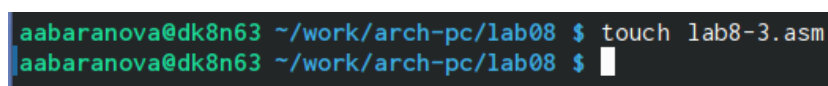


Рис. 4.15: Создание файла lab8-3.asm

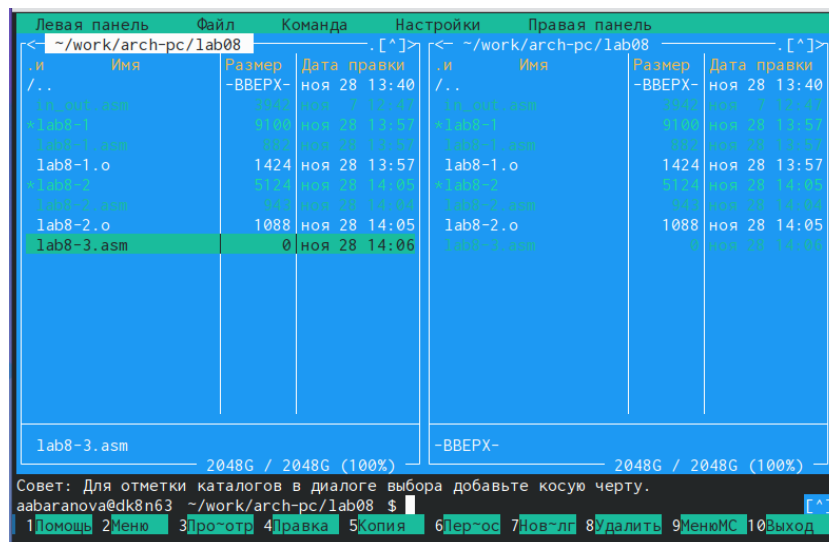


Рис. 4.16: Создание файла lab8-3.asm

Введём в файл lab8-3.asm текст программы (рис. 4.17).

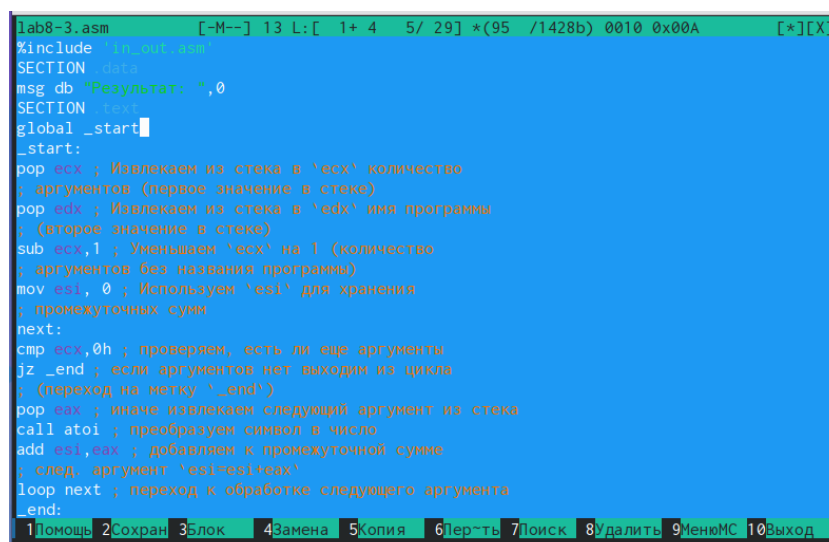


Рис. 4.17: Изменения в файле lab8-3.asm

Создадим исполняемый файл и запустим его, указав аргументы (рис. 4.18), (рис. 4.19).

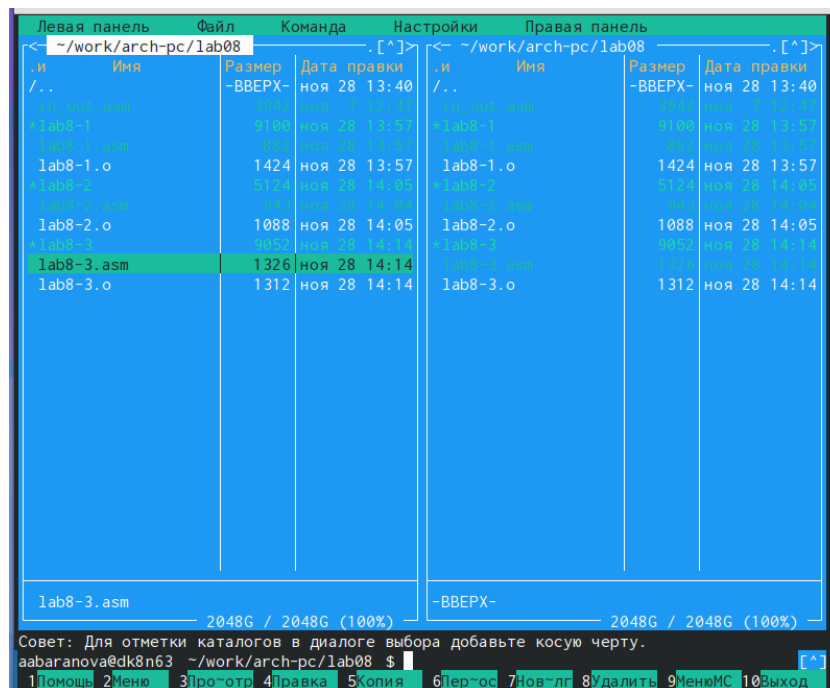


Рис. 4.18: Создание исполняемого файла и его запуск

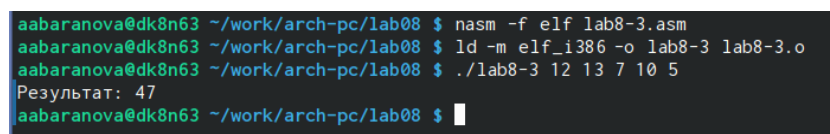


Рис. 4.19: Создание исполняемого файла и его запуск

Изменим в файле lab8-3.asm текст программы для вычисления произведения аргументов командной строки (рис. 4.20).

```

lab8-3.asm      [-M--]  5 L: [ 1+14  15/ 31] *(587 /1326b) 0010 0x00A  [*][X]
%include "lab8-1.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
mov eax,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
mov ebx,eax
mov eax,esi
mul ebx
mov 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. 4.20: Изменения в файле lab8-3.asm

Создадим исполняемый файл и запустим его, указав аргументы (рис. 4.21).

```

aabaranova@dk8n63 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aabaranova@dk8n63 ~/work/arch-pc/lab08 $ ./lab8-3 2 3 4
Результат: 24
aabaranova@dk8n63 ~/work/arch-pc/lab08 $

```

Рис. 4.21: Создание исполняемого файла и его запуск

4.3 Задание для самостоятельной работы

Напишем программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Создадим исполняемый файл и проверим его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$. (рис. 4.22), (рис. 4.23), (рис. 4.24), (рис. 4.25).

```

aabaranova@dk8n63 ~/work/arch-pc/lab08 $ touch lab8-4.asm
aabaranova@dk8n63 ~/work/arch-pc/lab08 $

```

Рис. 4.22: Создание файла lab8-4.asm

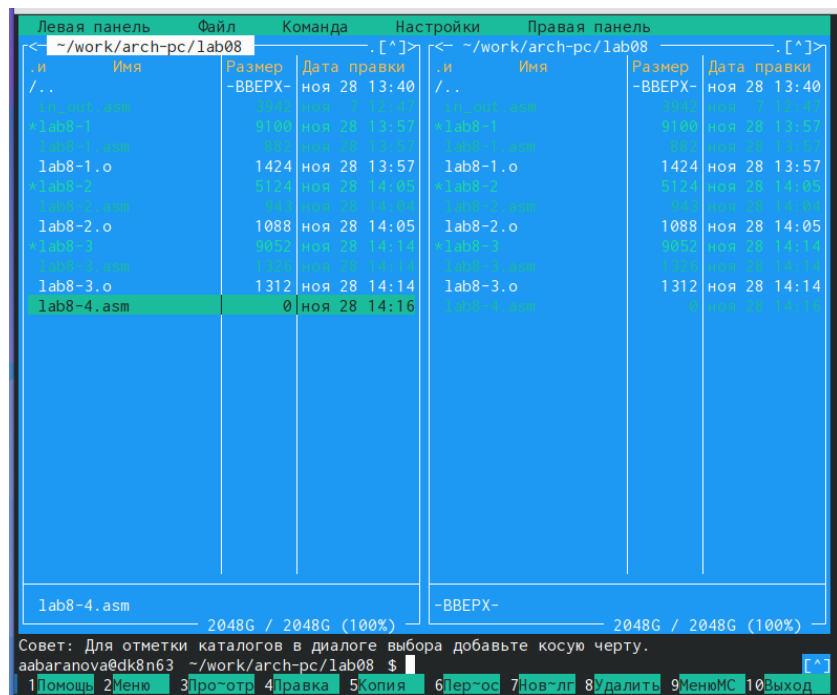


Рис. 4.23: Создание файла lab8-4.asm

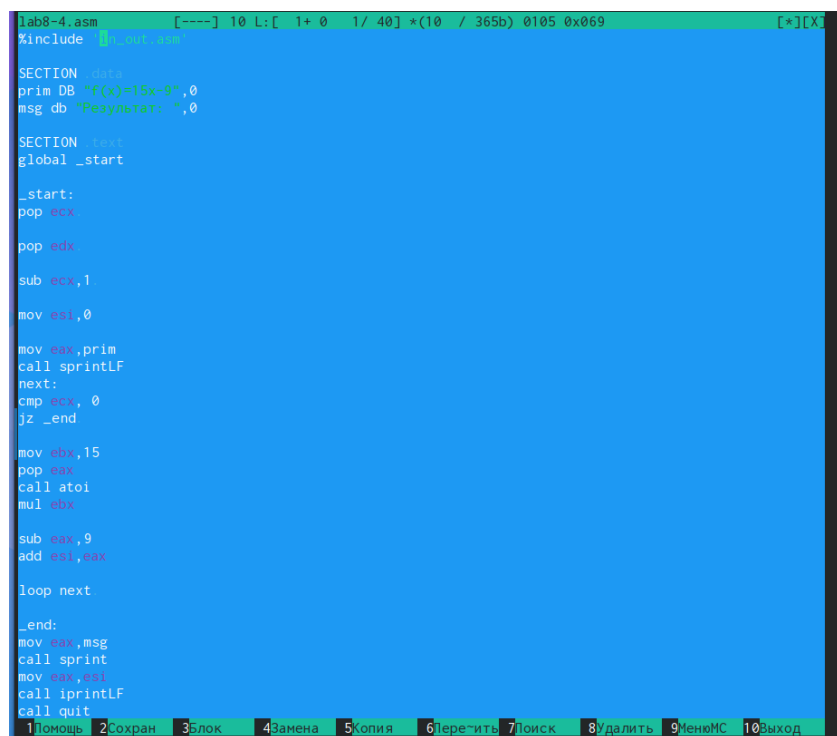


Рис. 4.24: Написание программы в lab8-4.asm

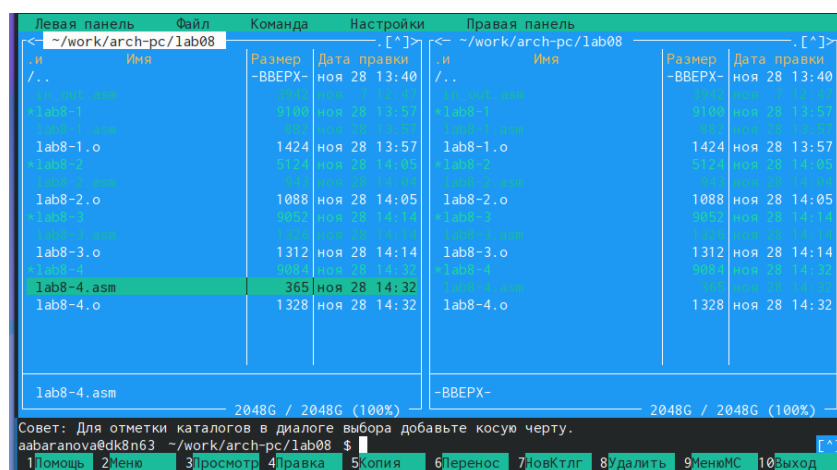


Рис. 4.25: Создание исполняемого файла и его запуск с указанными аргументами

5 Выводы

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