

Отчёт по лабораторной работе №8

дисциплина: Архитектура компьютера

Веретенников Дмитрий Олегович

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	23
	Список литературы	24

Список иллюстраций

4.1	Создание каталога и файла	8
4.2	Ввод текста программы	9
4.3	Проверка	10
4.4	Изменение значения регистра esx в цикле	11
4.5	Проверка	12
4.6	Изменения программы	13
4.7	Запуск программы	14
4.8	Ввод программы из листинга	15
4.9	Запуск программы	16
4.10	Ввод программы в файл	17
4.11	Создание исполняемого файла и его запуск	18
4.12	Изменение программы	19
4.13	Проверка	20
4.14	Текст программы	21
4.15	Проверка	22

Список таблиц

1 Цель работы

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

2 Задание

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

3 Теоретическое введение

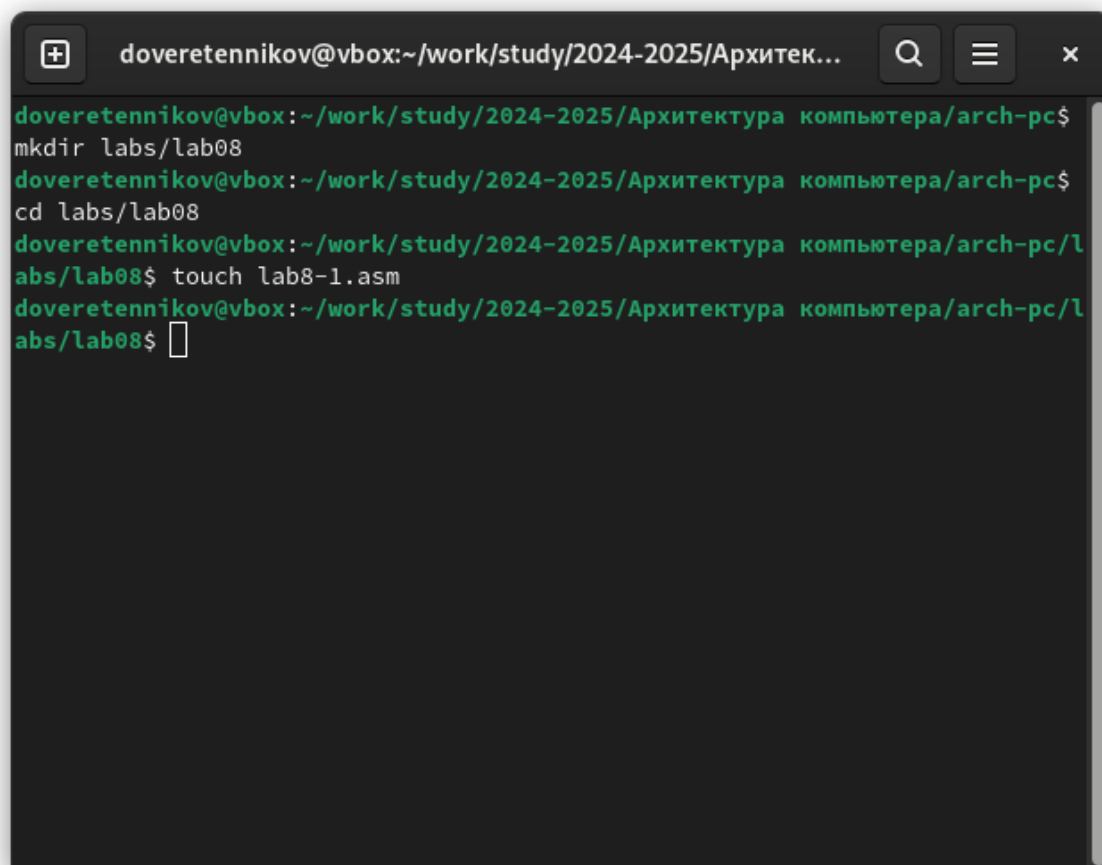
Стек – это структура данных, организованная по принципу LIFO («Last In – First Out»

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

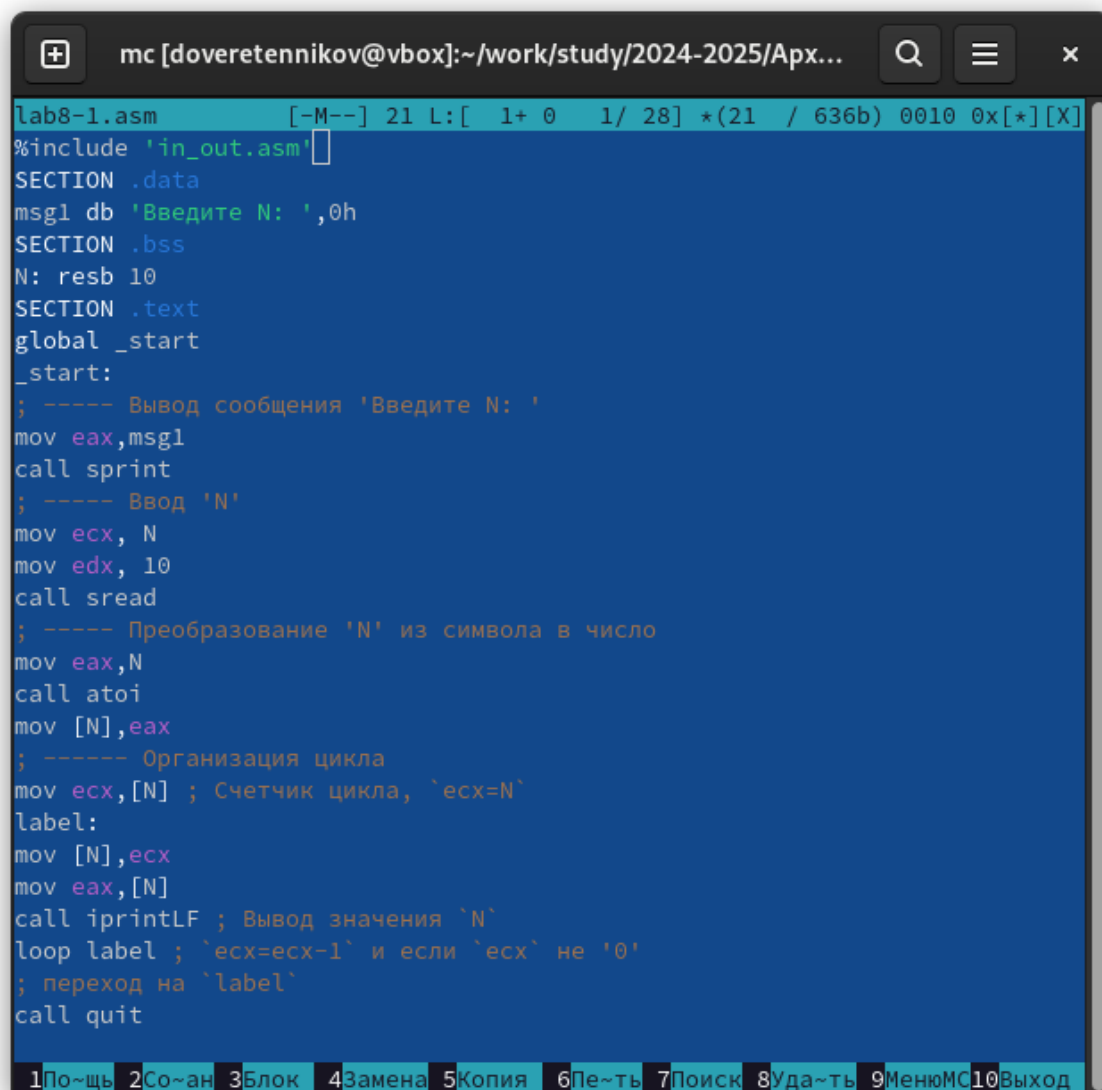
Создаю каталог для программ лабораторной работы №8 и создаю файл lab8-1.asm (рис. 4.1).

A screenshot of a terminal window with a dark background. The window title is "doveretennikov@vbox:~/work/study/2024-2025/Архитек...". The terminal shows the following commands and their outputs: 1. "mkdir labs/lab08" is executed. 2. "cd labs/lab08" is executed. 3. "touch lab8-1.asm" is executed. The prompt is now at the directory "abs/lab08".

```
doveretennikov@vbox:~/work/study/2024-2025/Архитек...$ mkdir labs/lab08
doveretennikov@vbox:~/work/study/2024-2025/Архитек...$ cd labs/lab08
doveretennikov@vbox:~/work/study/2024-2025/Архитек.../labs/lab08$ touch lab8-1.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитек.../labs/lab08$
```

Рис. 4.1: Создание каталога и файла

Ввожу в файл lab8-1.asm текст программы из листинга 8.1 (рис. 4.2).

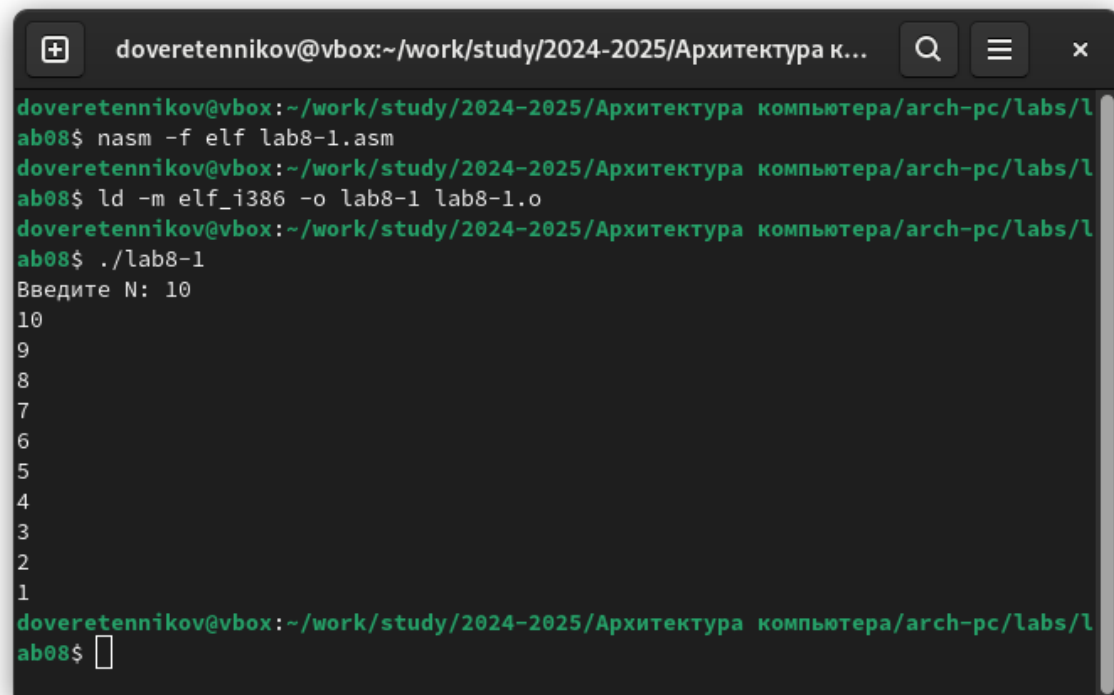


```
lab8-1.asm [-M--] 21 L:[ 1+ 0 1/ 28] *(21 / 636b) 0010 0x[*][X]
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- 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
```

1По~щъ 2Со~ан 3Блок 4Замена 5Копия 6Пе~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 4.2: Ввод текста программы

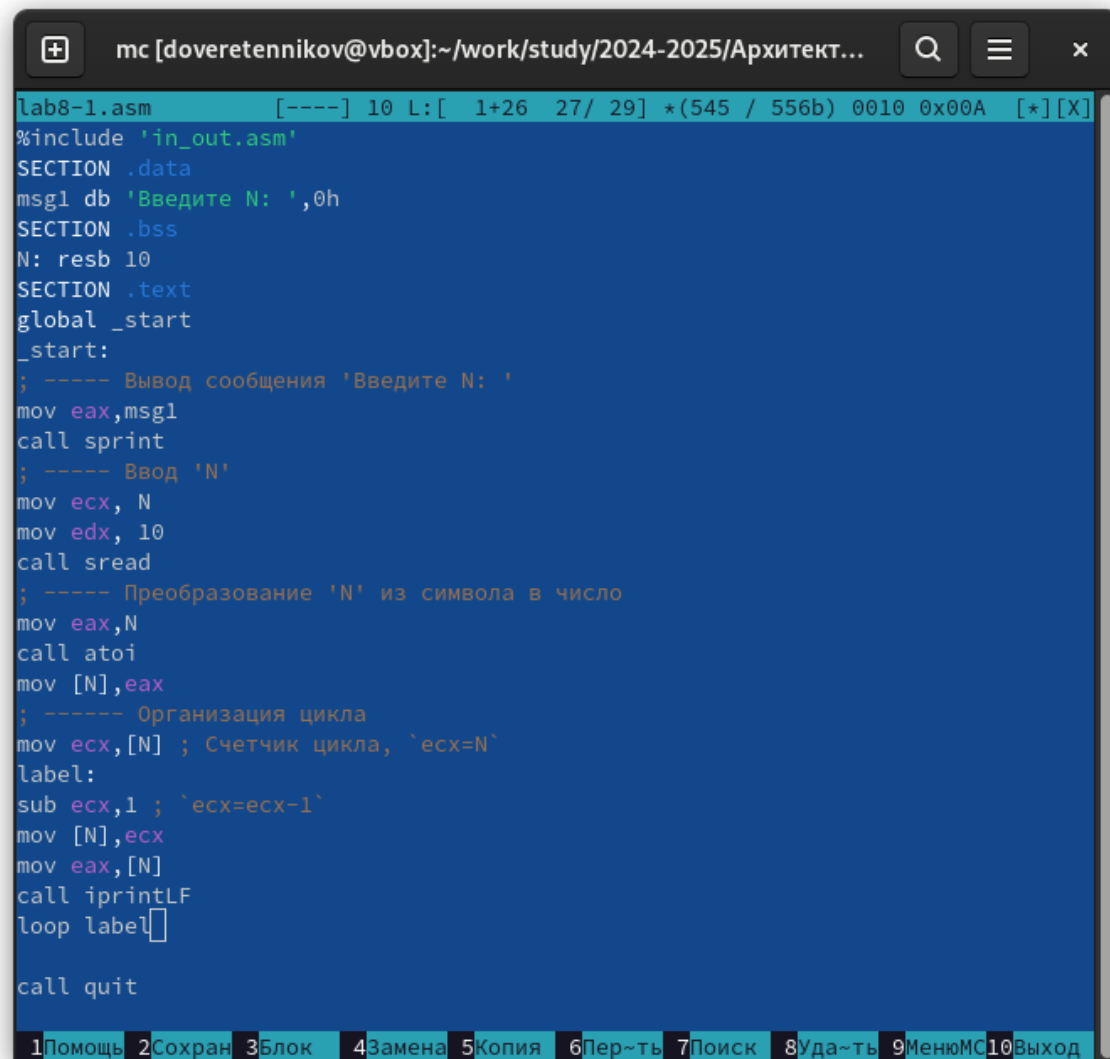
Проверяю работу файла (рис. 4.3).

A terminal window with a dark background and green text. The window title is "doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...". The terminal shows the following commands and output:

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ nasm -f elf lab8-1.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$
```

Рис. 4.3: Проверка

Изменяю текст программы добавив изменение значения регистра `ecx` в цикле (рис. 4.4).



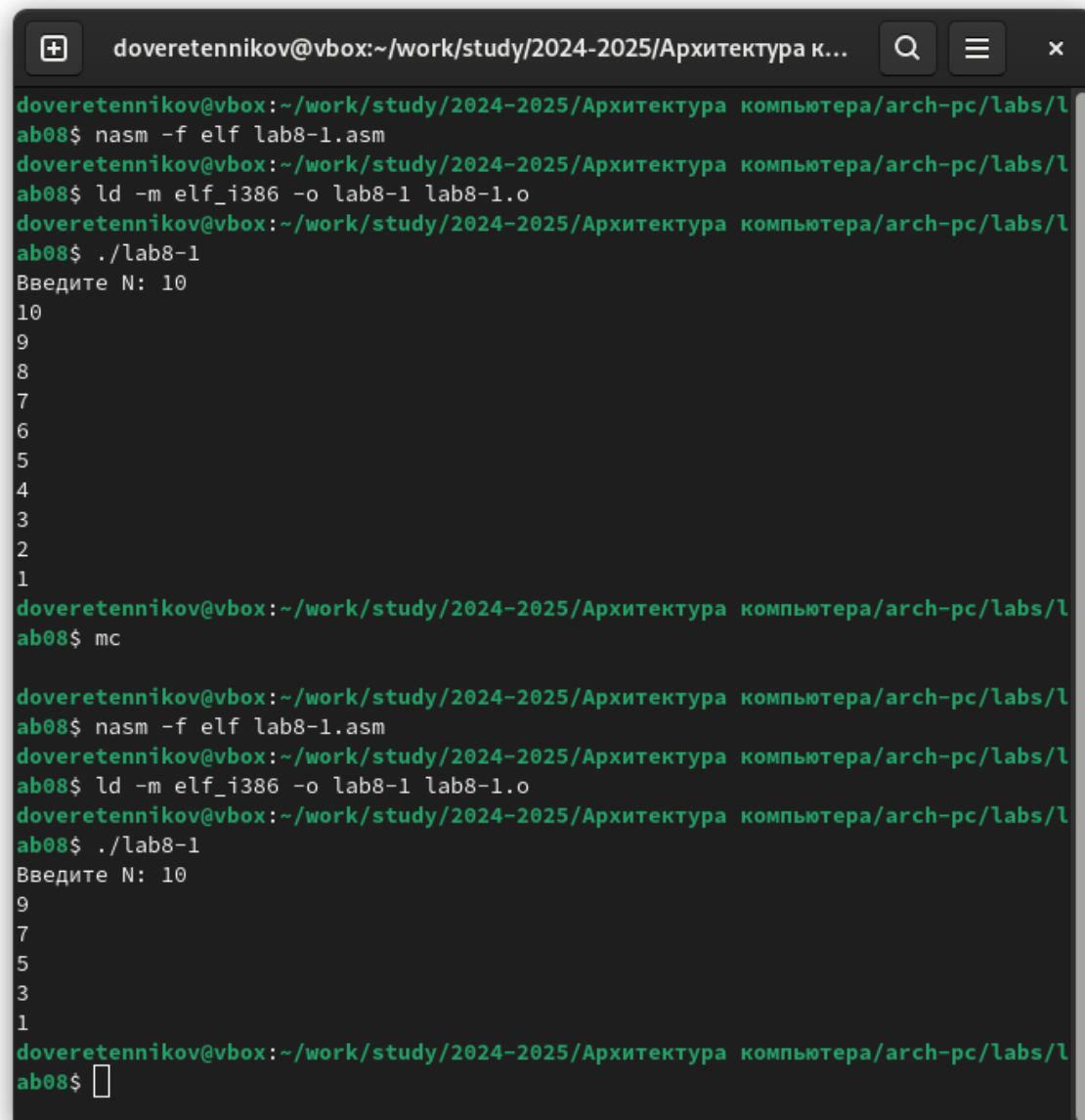
```
lab8-1.asm [----] 10 L: [ 1+26 27/ 29] *(545 / 556b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call 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

call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход

Рис. 4.4: Изменение значения регистра ecx в цикле

Проверяю работу программы (рис. 4.5).



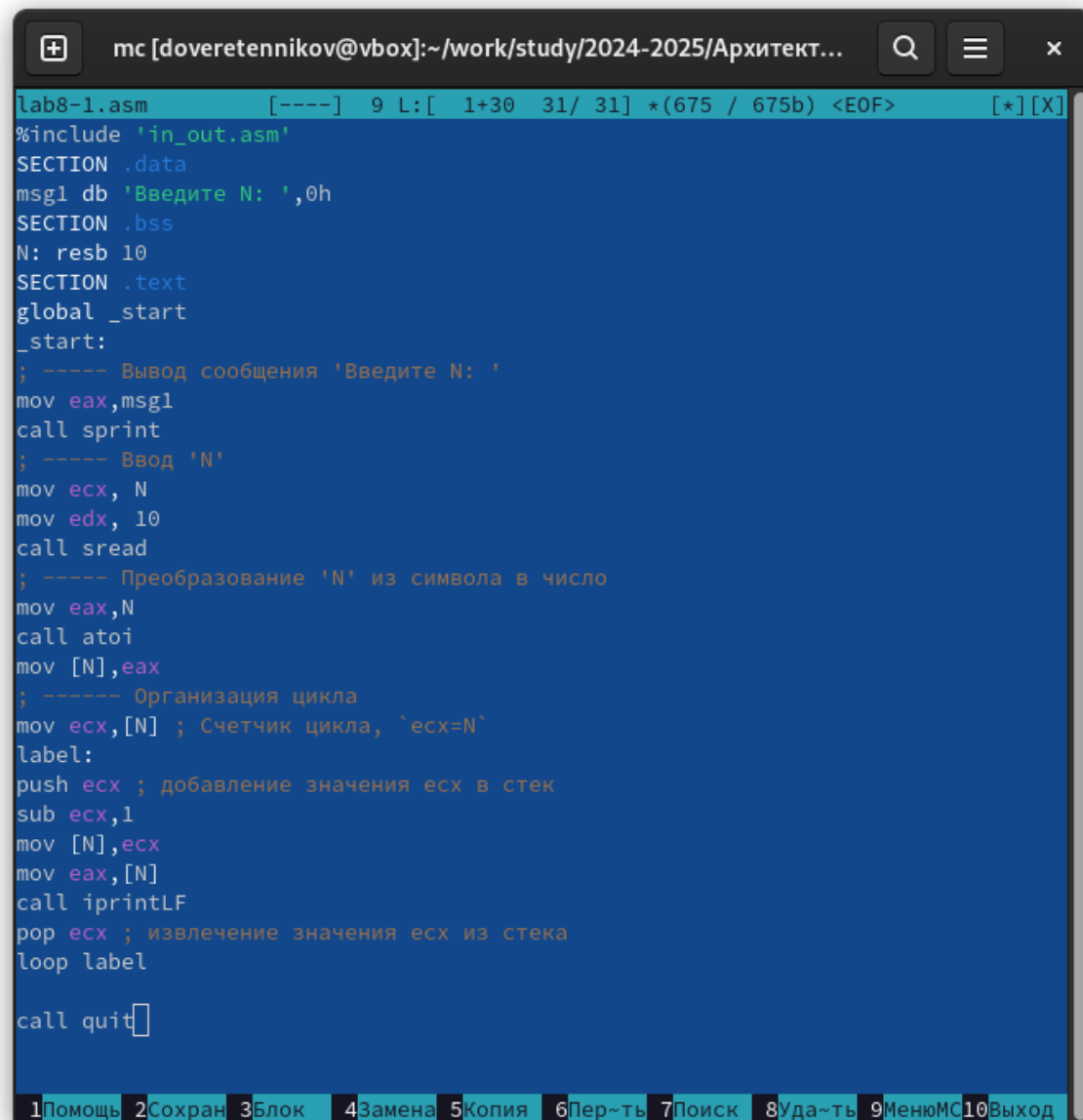
```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ nasm -f elf lab8-1.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ mc

doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ nasm -f elf lab8-1.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$
```

Рис. 4.5: Проверка

Так как регистр `ecx` на каждой итерации уменьшается на 2, то и количество итераций уменьшается в 2 раза.

Вношу изменения в текст программы добавив команды `push` и `pop` (рис. 4.6).



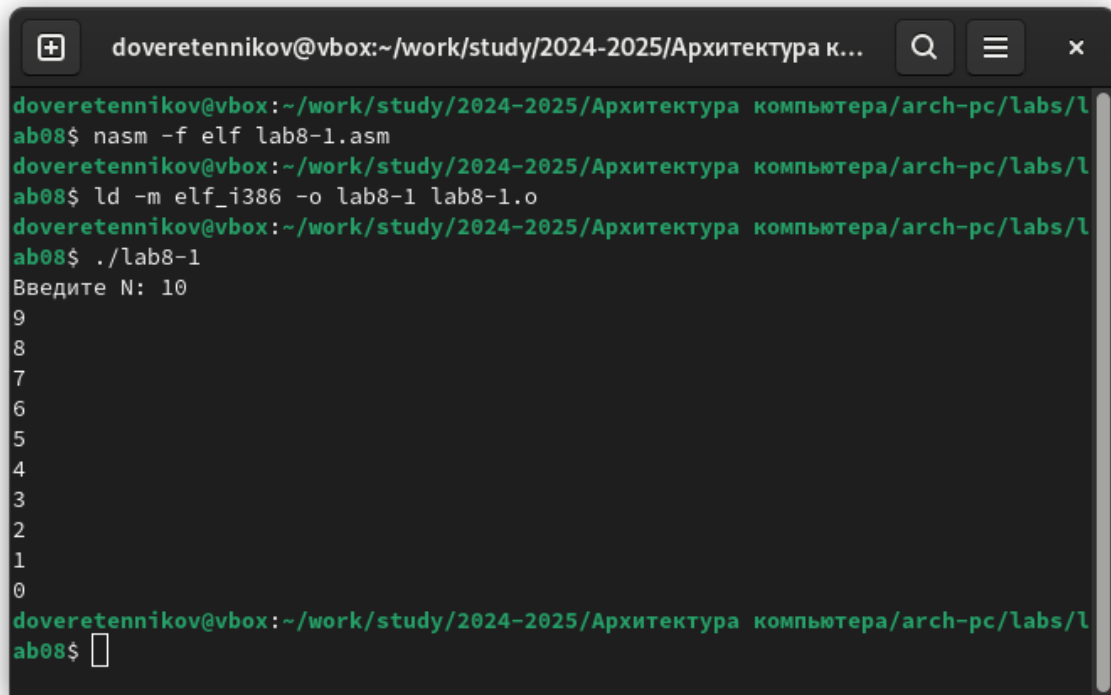
```
lab8-1.asm [----] 9 L: [ 1+30 31/ 31] *(675 / 675b) <EOF> [*] [X]
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- 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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 4.6: Изменения программы

Запускаю измененную программу (рис. 4.7).



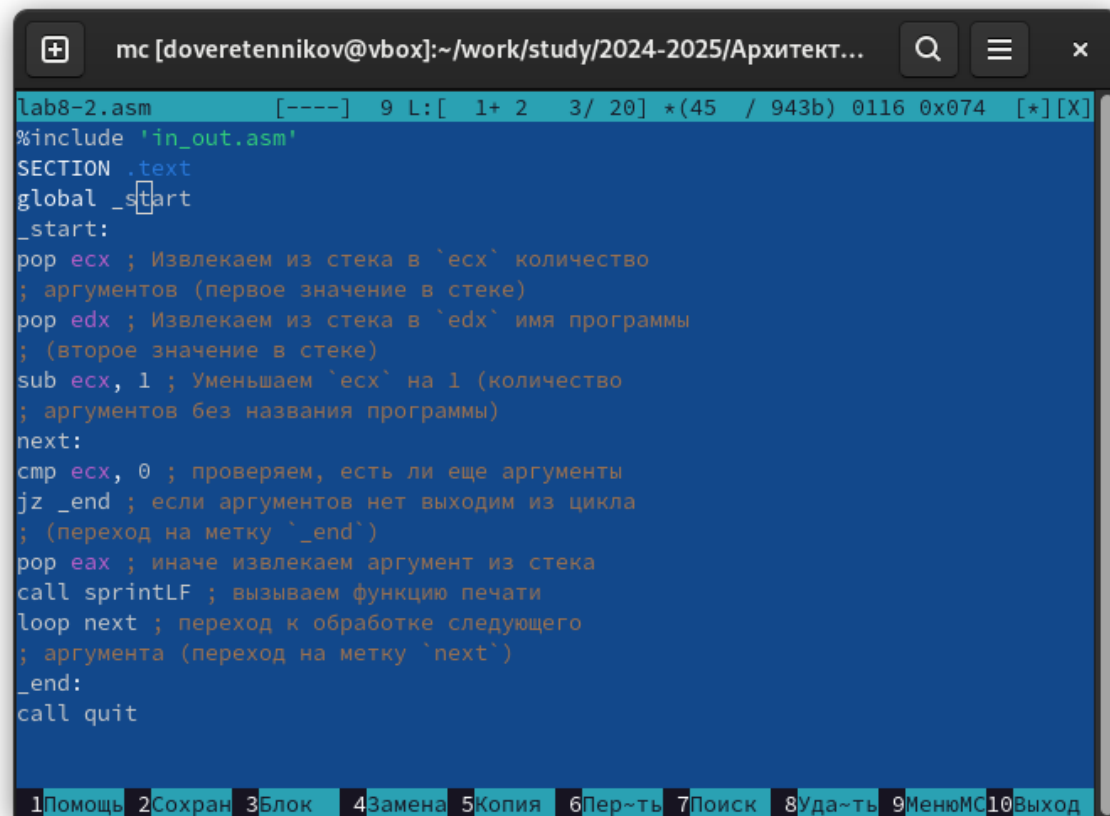
```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ nasm -f elf lab8-1.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$
```

Рис. 4.7: Запуск программы

В данном случае теперь число проходов цикла соответствует значению N введенному с клавиатуры, но произошло смещение чисел на -1.

4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm и ввожу в него текст программы из листинга 8.2 (рис. 4.8).

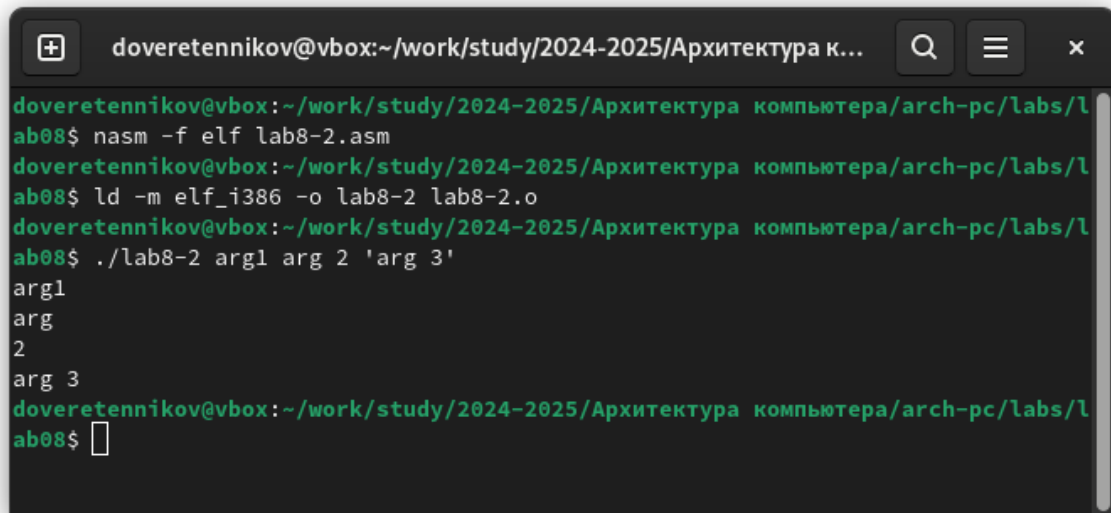


```
lab8-2.asm [----] 9 L: [ 1+ 2 3/ 20] *(45 / 943b) 0116 0x074 [*][X]
#include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx, 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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюMC10Выход

Рис. 4.8: Ввод программы из листинга

Создаю исполняемый файл и запускаю его, указав аргументы (рис. 4.9).

A terminal window with a dark background and light green text. The window title is "doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...". The terminal shows the following commands and output:

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ nasm -f elf lab8-2.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$
```

Рис. 4.9: Запуск программы

Программой было обработано столько же аргументов, сколько мы и ввели.

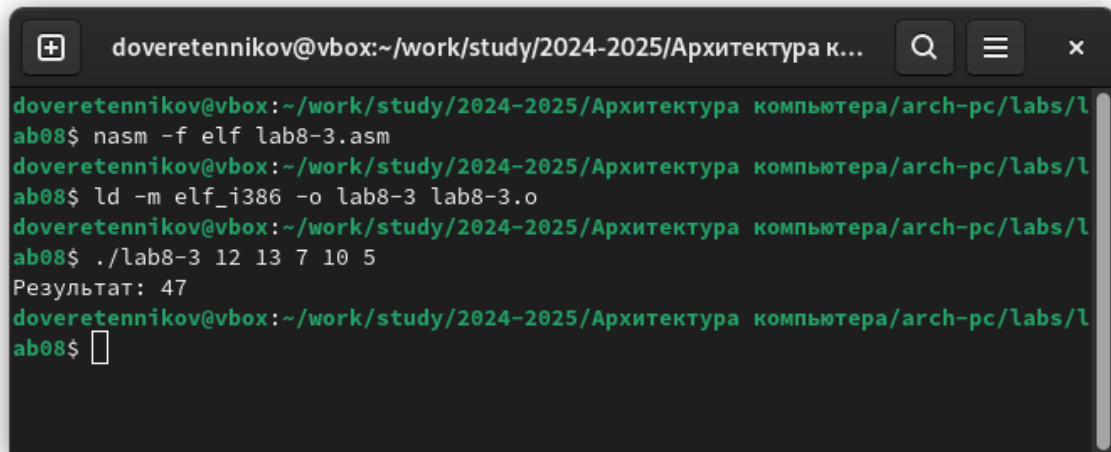
Создаю новый файл и ввожу в него текст программы из листинга 8.3 (рис. 4.10).


```
lab8-3.asm [-M--] 21 L: [ 1+ 0 1/ 29] *(21 /1428b) 0010 0x00A [*][X]
#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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС 10Выход

Рис. 4.10: Ввод программы в файл

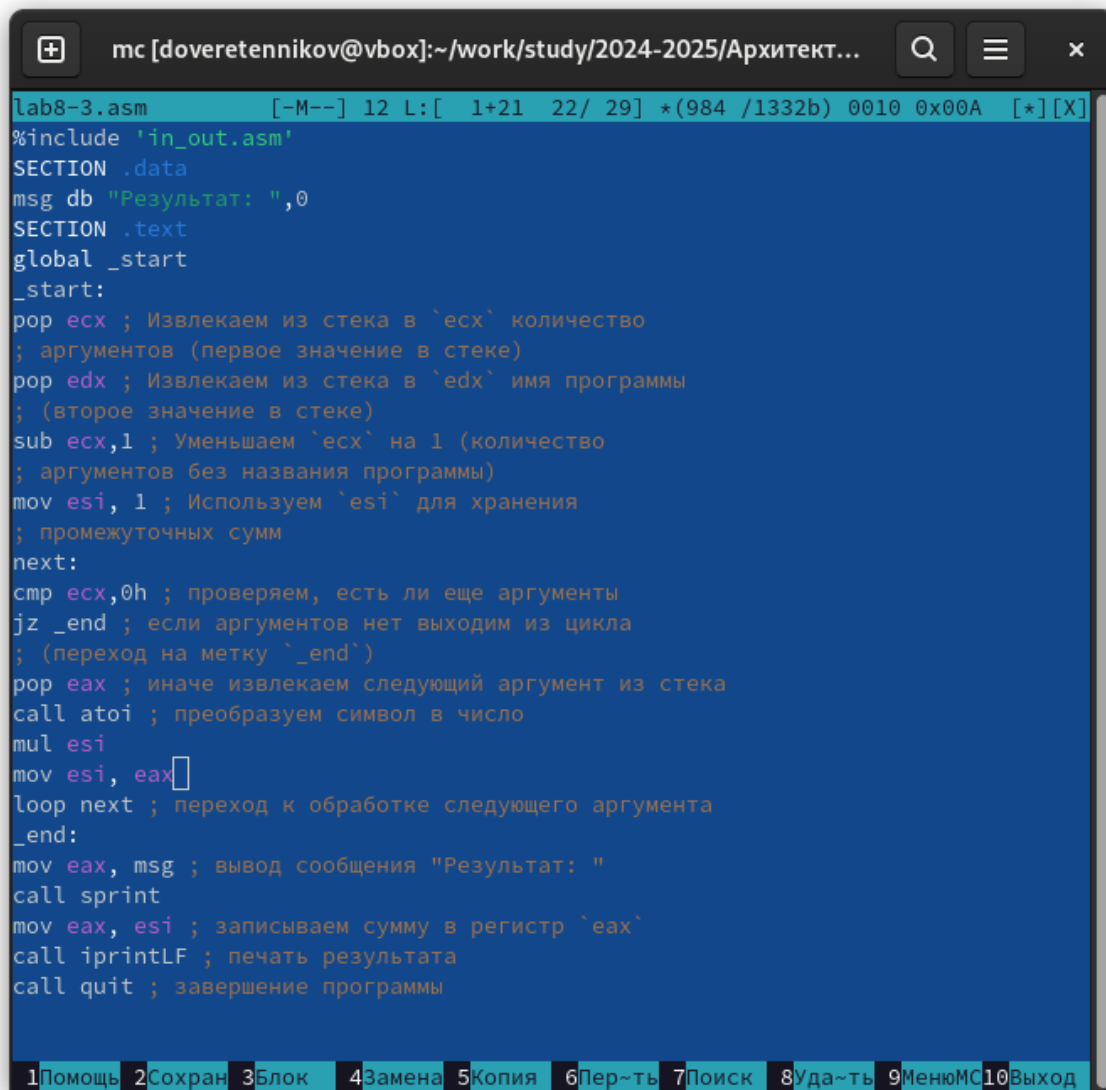
Создаю исполняемый файл и запускаю его, указав аргументы из примера (рис. 4.11).



```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ nasm -f elf lab8-3.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ./lab8-3 12 13 7 10 5
Результат: 47
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$
```

Рис. 4.11: Создание исполняемого файла и его запуск

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 4.12).

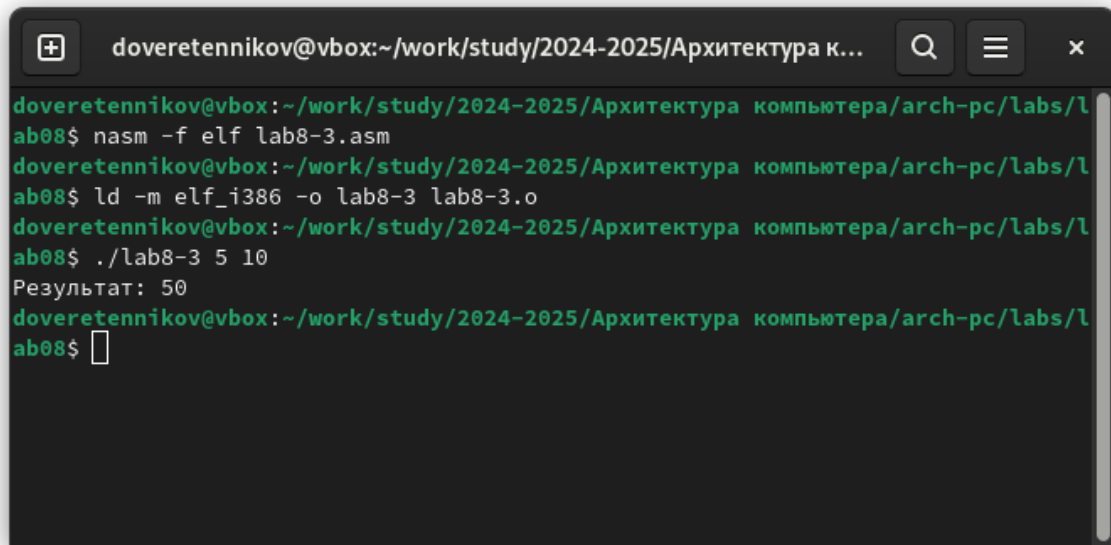


```
lab8-3.asm [-M--] 12 L: [ 1+21 22/ 29] *(984 /1332b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (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
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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 4.12: Изменение программы

Проверяю корректность работы программы (рис. 4.13).

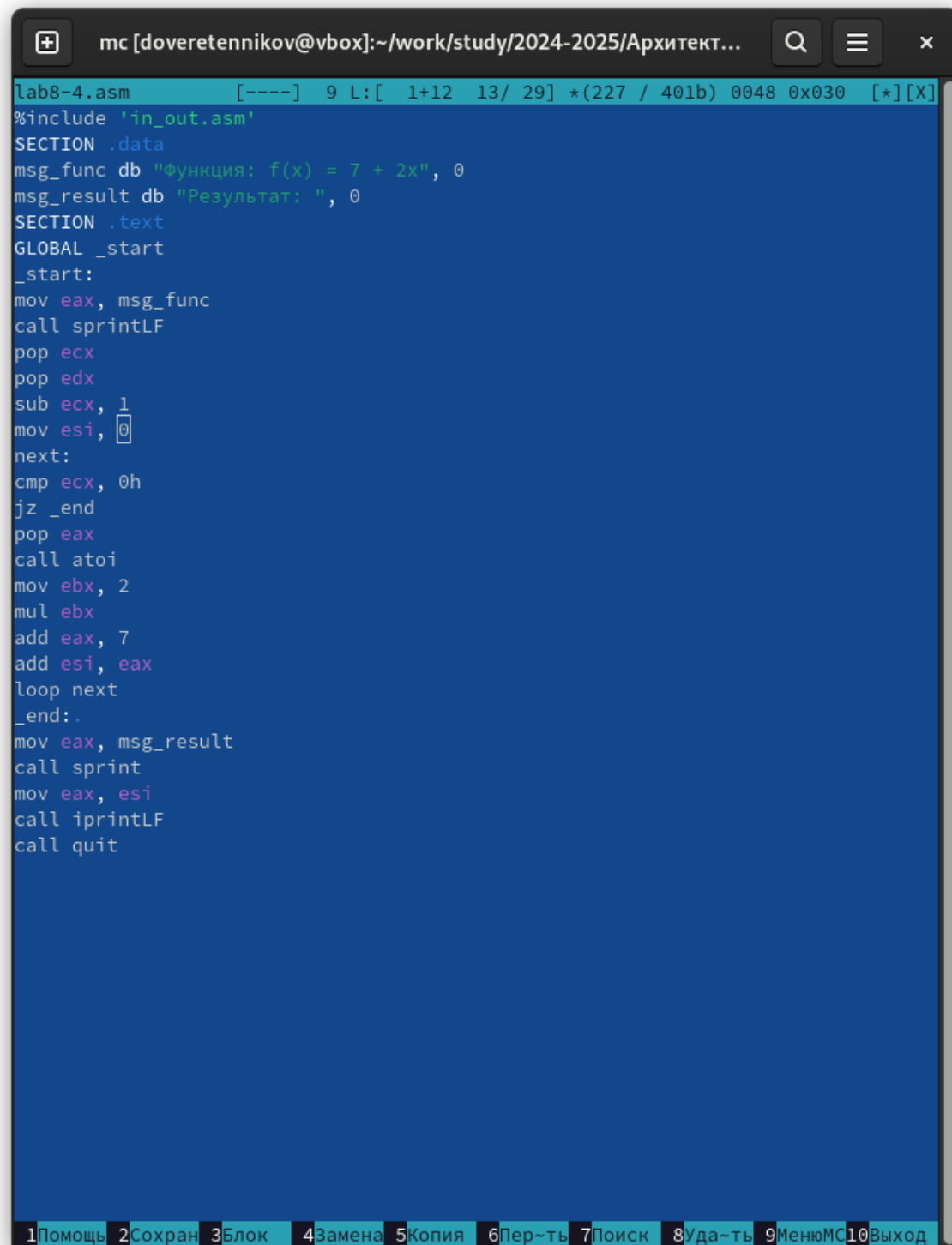


```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-3.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-3 5 10
Результат: 50
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 4.13: Проверка

4.3 Задание для самостоятельной работы

Напишите программу, которая находит сумму значений функции $f(x)$, так как у меня 8 вариант пишу программу для функции $f(x) = 7 + 2x$ (рис. 4.14).

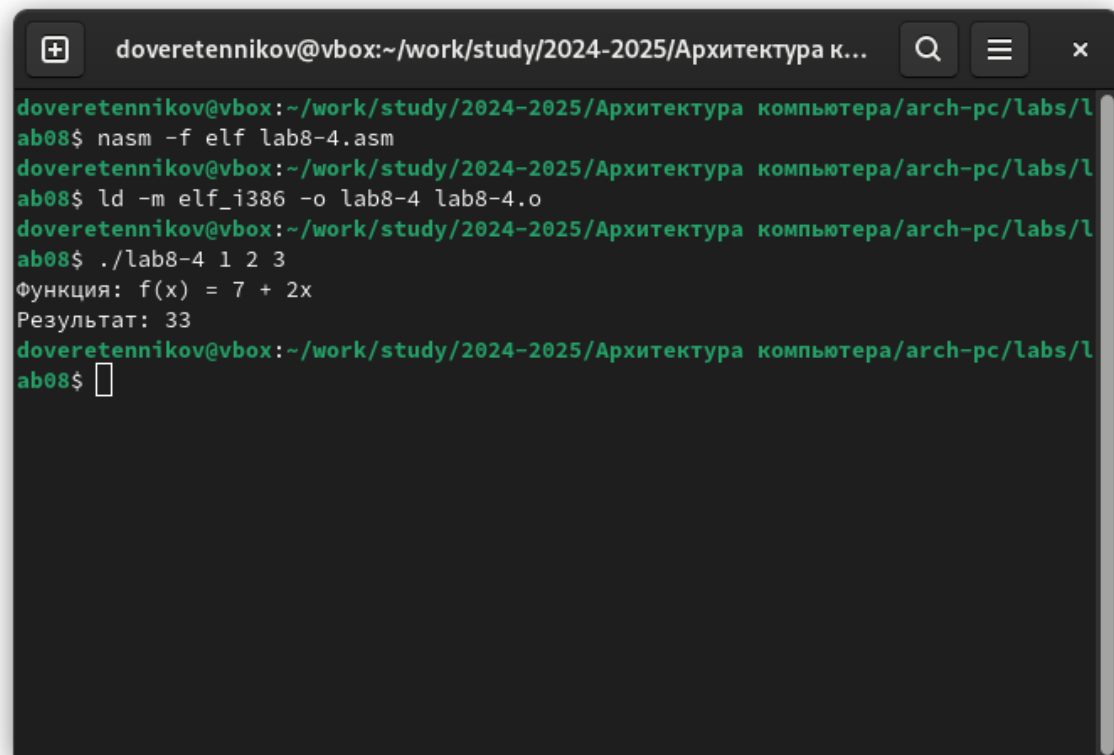


```
lab8-4.asm [----] 9 L: [ 1+12 13/ 29] *(227 / 401b) 0048 0x030 [*][X]
#include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 7 + 2x", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprintf
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
mov ebx, 2
mul ebx
add eax, 7
add esi, eax
loop next
_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintLF
call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 4.14: Текст программы

Далле проверяю корректность работы программы (рис. 4.15).

A terminal window with a dark background and green text. The window title is "doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...". The terminal shows the following commands and output:

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ nasm -f elf lab8-4.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$ ./lab8-4 1 2 3
Функция:  $f(x) = 7 + 2x$ 
Результат: 33
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab08$
```

Рис. 4.15: Проверка

После этого отправляю файлы на github

5 Выводы

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

Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-

- е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).