

Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютеров и операционные системы

Бизев Никита Владимирович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Реализация циклов в NASM	9
4.2	Обработка аргументов командной строки	14
5	Выводы	18
6	Список литературы	19

Список иллюстраций

4.1	Создание файлов для лабораторной работы	9
4.2	Ввод текста из листинга 8.1	10
4.3	Запуск исполняемого файла	10
4.4	Изменение текста программы	11
4.5	Запуск обновленной программы	12
4.6	Изменение текста программы	13
4.7	Запуск исполняемого файла	13
4.8	Ввод текста программы из листинга 8.2	14
4.9	Запуск исполняемого файла	15
4.10	Ввод текста программы из листинга 8.3	16
4.11	Запуск исполняемого файла	16
4.12	Изменение текста программы	17
4.13	Запуск исполняемого файла	17

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. Задание для самостоятельной 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. 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.

Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек.

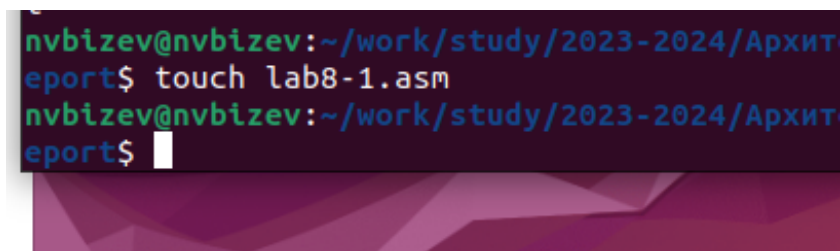
Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

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

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

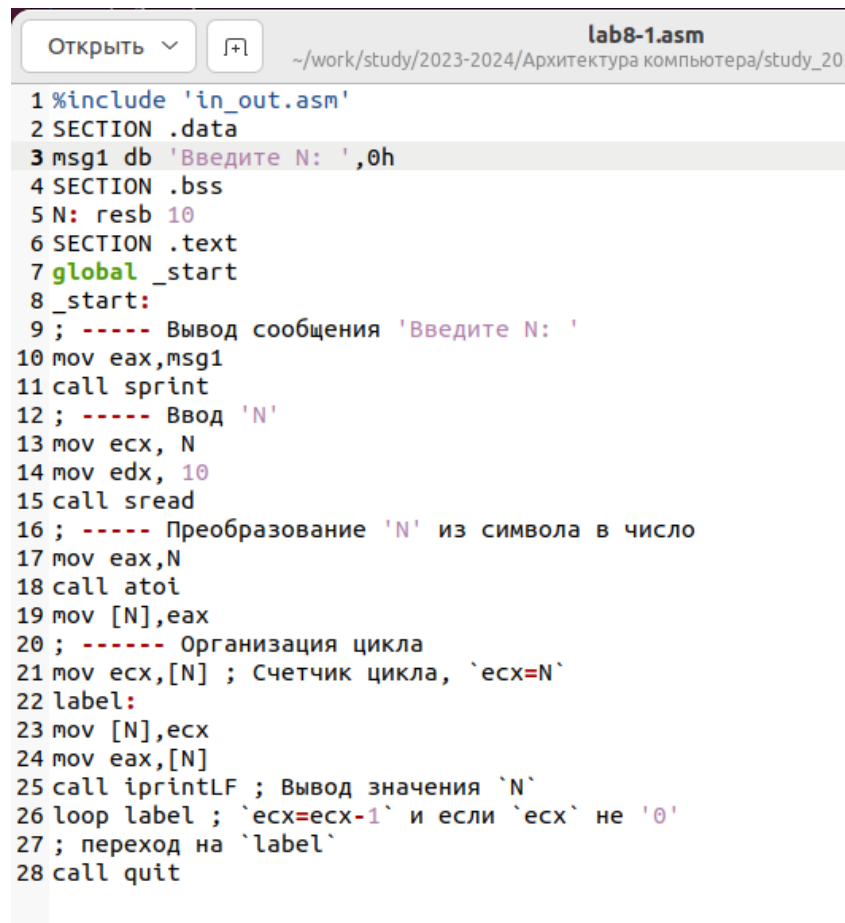
Создаю файл lab8-1.asm. (рис. 4.1).

A screenshot of a terminal window with a dark background and light-colored text. The prompt is 'nvbizev@nvbizev:~/work/study/2023-2024/Архитектура'. The command 'touch lab8-1.asm' has been entered and executed. The prompt is now 'nvbizev@nvbizev:~/work/study/2023-2024/Архитектура\$' with a cursor at the end.

```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура$ touch lab8-1.asm
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура$
```

Рис. 4.1: Создание файлов для лабораторной работы

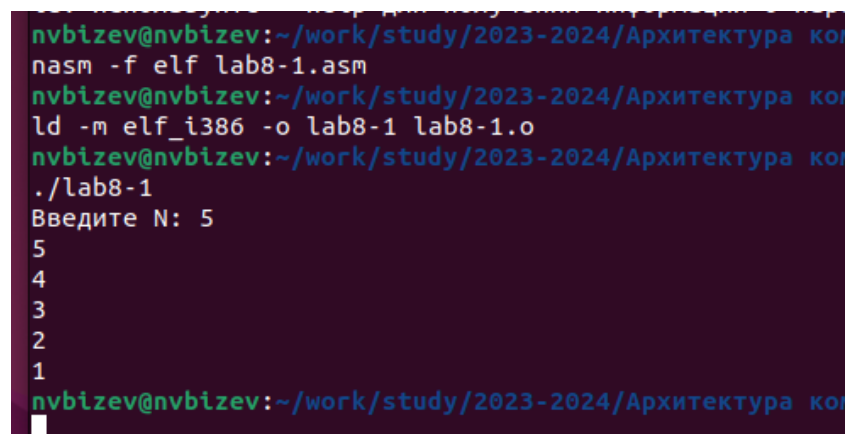
Ввожу в файл lab8-1.asm текст программы из листинга 8.1. (рис. 4.2).



```
lab8-1.asm
~/work/study/2023-2024/Архитектура компьютера/study_20
Открыть
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 mov [N],ecx
24 mov eax,[N]
25 call iprintLF ; Вывод значения `N`
26 loop label ; `ecx=ecx-1` и если `ecx` не '0'
27 ; переход на `label`
28 call quit
```

Рис. 4.2: Ввод текста из листинга 8.1

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

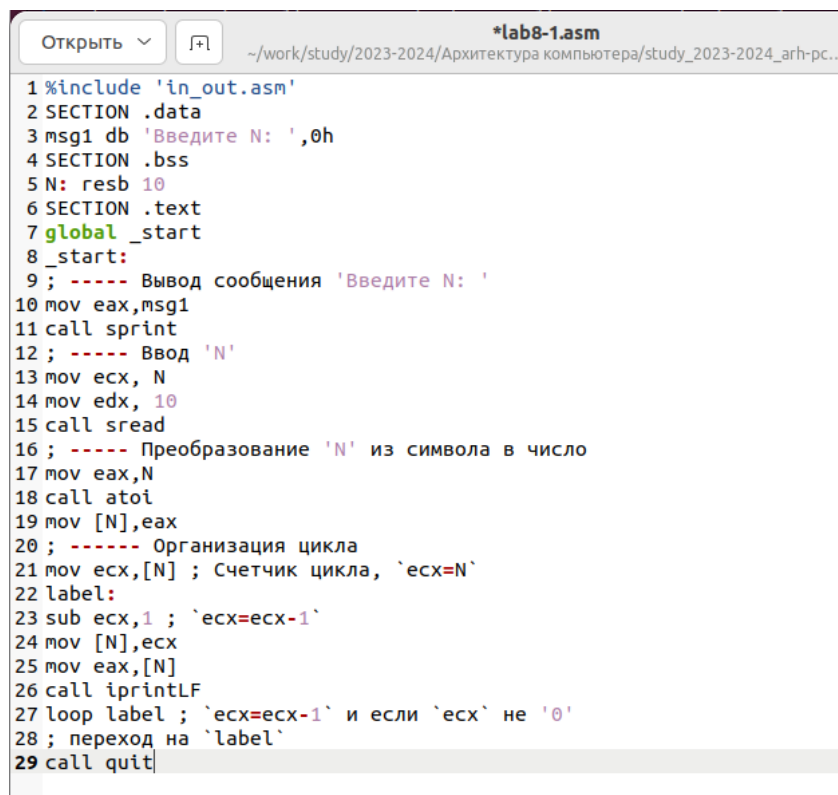


```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
nasm -f elf lab8-1.asm
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
ld -m elf_i386 -o lab8-1 lab8-1.o
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
./lab8-1
Введите N: 5
5
4
3
2
1
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
```

Рис. 4.3: Запуск исполняемого файла

Данная программа выводит числа от N до 1 включительно.

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



```
Открыть  [+]
```

~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-пс... *lab8-1.asm

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 sub ecx,1 ; `ecx=ecx-1`
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF
27 loop label ; `ecx=ecx-1` и если `ecx` не `0`
28 ; переход на `label`
29 call quit
```

Рис. 4.4: Изменение текста программы

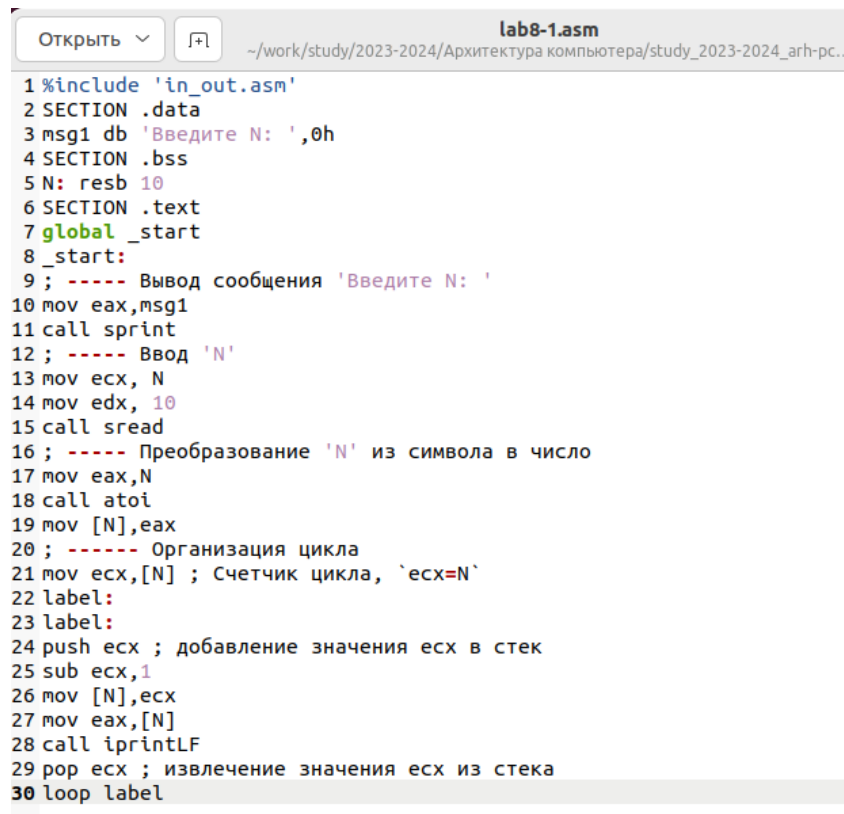
Создаю исполняемый файл и проверяю его работу. (рис. 4.5).

```
nvbizev@nvbizev: ~/work/study/2023-2024/Архитектура компьютера/s
4292494530
4292494528
4292494526
4292494524
4292494522
4292494520
4292494518
4292494516
4292494514
4292494512
4292494510
4292494508
4292494506
4292494504
4292494502
4292494500
4292494498
4292494496
4292494494
4292494492
4292494490
4292494488
```

Рис. 4.5: Запуск обновленной программы

В данном случае число проходов цикла не соответствует введенному с клавиатуры значению.

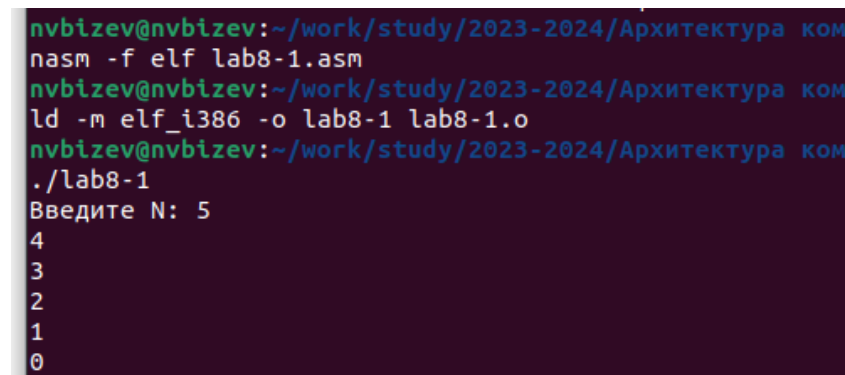
Вношу изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop. (рис. 4.6).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 label:
24 push ecx ; добавление значения ecx в стек
25 sub ecx,1
26 mov [N],ecx
27 mov eax,[N]
28 call iprintLF
29 pop ecx ; извлечение значения ecx из стека
30 loop label
```

Рис. 4.6: Изменение текста программы

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



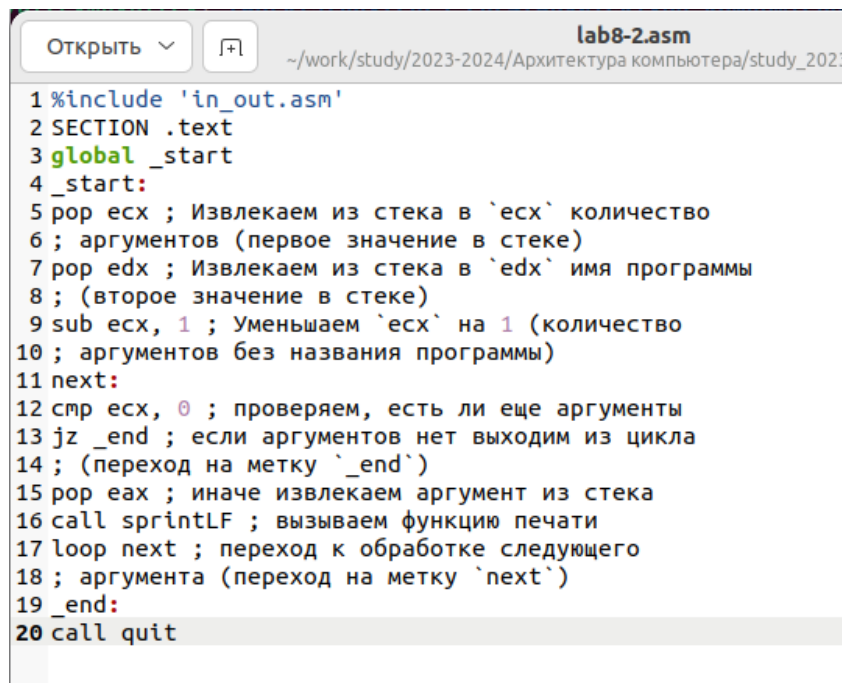
```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ком
nasm -f elf lab8-1.asm
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ком
ld -m elf_i386 -o lab8-1 lab8-1.o
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ком
./lab8-1
Введите N: 5
4
3
2
1
0
```

Рис. 4.7: Запуск исполняемого файла

В данном случае число проходов цикла соответствует введенному с клавиатуры значению и выводит числа от N-1 до 0 включительно.

4.2 Обработка аргументов командной строки

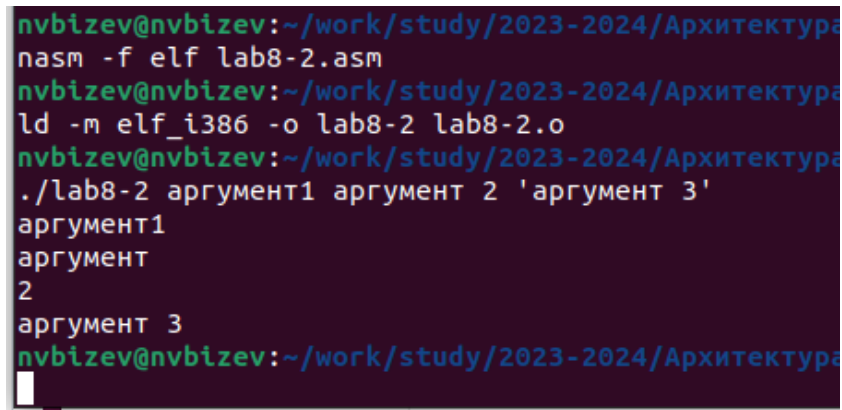
Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы из листинга 8.2. (рис. 4.8).



```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов без названия программы)
11 next:
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет выходим из цикла
14 ; (переход на метку `_end`)
15 pop eax ; иначе извлекаем аргумент из стека
16 call sprintLF ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit
```

Рис. 4.8: Ввод текста программы из листинга 8.2

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

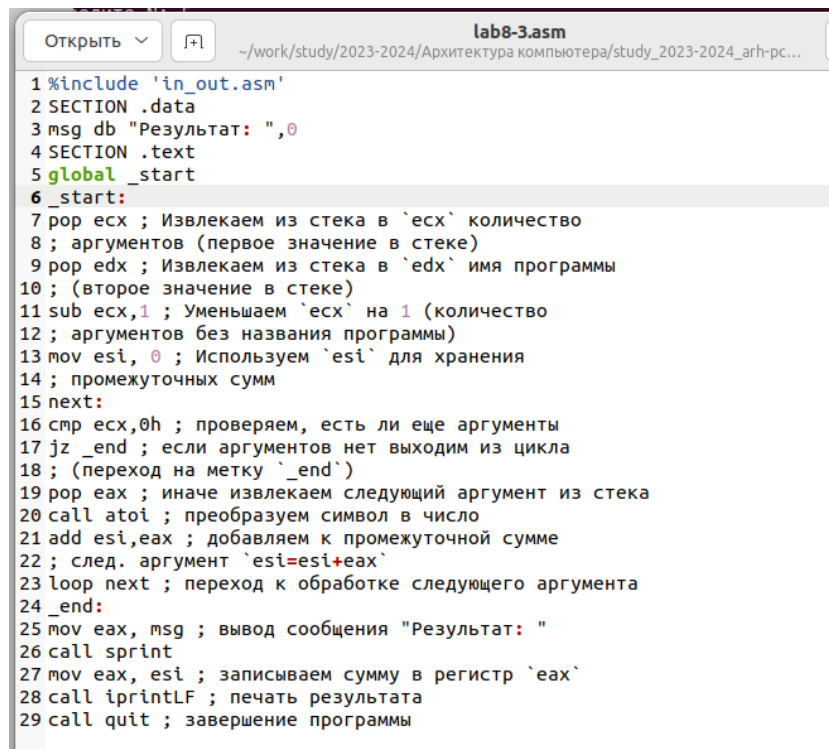
A terminal window with a dark background and light green text. The prompt is 'nvbizev@nvbizev:~/work/study/2023-2024/Архитектура'. The commands entered are: 'nasm -f elf lab8-2.asm', 'ld -m elf_i386 -o lab8-2 lab8-2.o', and './lab8-2 аргумент1 аргумент 2 'аргумент 3''. The output shows 'аргумент1', 'аргумент', '2', and 'аргумент 3' on separate lines.

```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура
nasm -f elf lab8-2.asm
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура
ld -m elf_i386 -o lab8-2 lab8-2.o
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура
./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура
```

Рис. 4.9: Запуск исполняемого файла

Программа вывела 4 аргумента, так как аргумент 2 не взят в кавычки, в отличие от аргумента 3, поэтому из-за пробела программа считывает “2” как отдельный аргумент.

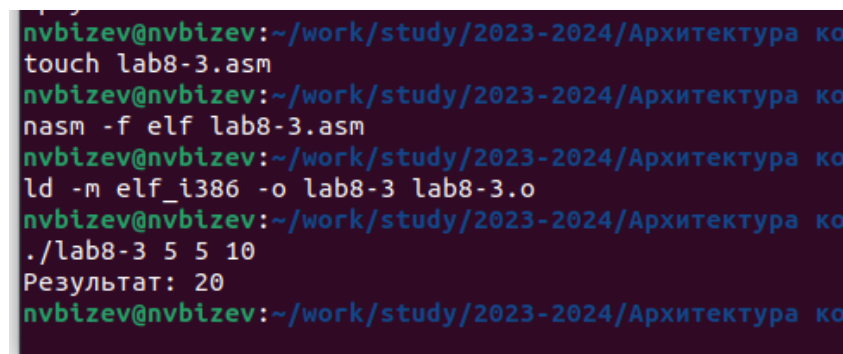
Рассмотрим пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создаю файл lab8-3.asm в каталоге ~/work/archpc/lab08 и ввожу в него текст программы из листинга 8.3. (рис. 4.10).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент `esi=esi+eax`
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр `eax`
28 call iprintLF ; печать результата
29 call quit ; завершение программы
```

Рис. 4.10: Ввод текста программы из листинга 8.3

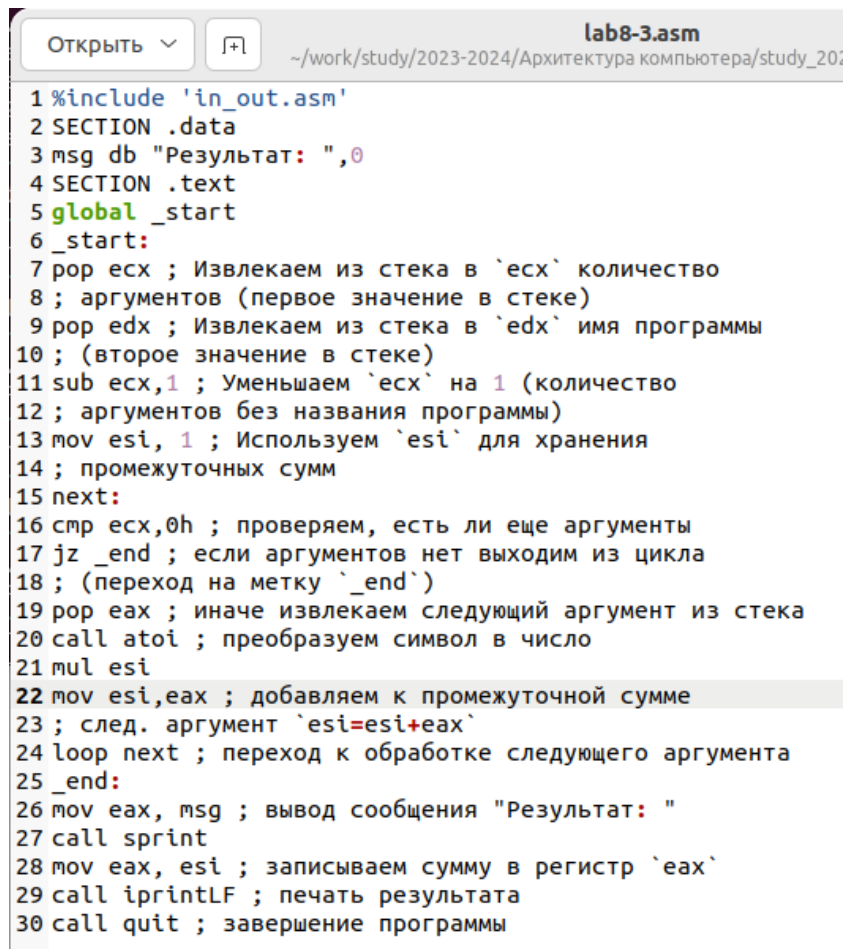
Создаю исполняемый файл и запускаю его, указав аргументы. (рис. 4.11).



```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
touch lab8-3.asm
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
nasm -f elf lab8-3.asm
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
ld -m elf_i386 -o lab8-3 lab8-3.o
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
./lab8-3 5 5 10
Результат: 20
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура ко
```

Рис. 4.11: Запуск исполняемого файла

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

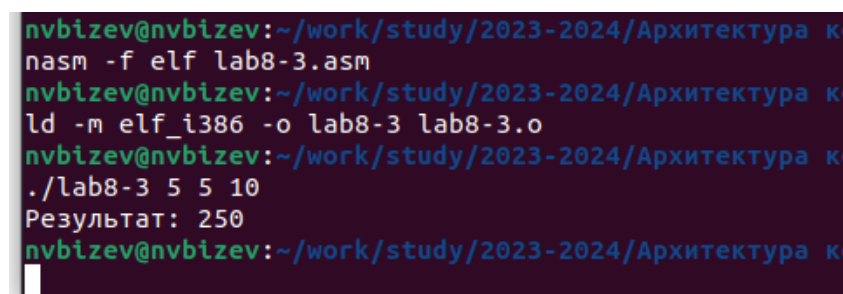


```
lab8-3.asm
~/work/study/2023-2024/Архитектура компьютера/study_2023-2024/

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 1 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 mul esi
22 mov esi,eax ; добавляем к промежуточной сумме
23 ; след. аргумент `esi=esi+eax`
24 loop next ; переход к обработке следующего аргумента
25 _end:
26 mov eax, msg ; вывод сообщения "Результат: "
27 call sprint
28 mov eax, esi ; записываем сумму в регистр `eax`
29 call iprintLF ; печать результата
30 call quit ; завершение программы
```

Рис. 4.12: Изменение текста программы

Создаю исполняемый файл и запускаю его, указав аргументы. (рис. 4.13).



```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура к
nasm -f elf lab8-3.asm
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура к
ld -m elf_i386 -o lab8-3 lab8-3.o
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура к
./lab8-3 5 5 10
Результат: 250
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура к
```

Рис. 4.13: Запуск исполняемого файла

5 Выводы

Благодаря данной лабораторной работе я приобрел навыки написания программ использованием циклов и обработкой аргументов командной строки, что поможет мне при выполнении последующих лабораторных работ.

6 Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/%D0%9B%D0%B0