

Отчет по лабораторной работе №8

дисциплина: Архитектура компьютера

Маньковская Дарья Станиславовна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель данной лабораторной работы - приобретение практического опыта в написании программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклов в NASM.
2. Обработка аргументов командной строки.
3. Выполнение заданий 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. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop). 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. Существует ещё две команды для добавления значений в стек. Это команда `pusha`, которая помещает в стек содержимое всех регистров общего назначения в следующем порядке: `ax`, `sx`, `dx`, `bx`, `sp`, `bp`, `si`, `di`. А также команда `pushf`, которая служит для перемещения в стек содержимого регистра флагов. Обе эти команды не имеют операндов. 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 `ecx`. Наиболее простой является инструкция `loop`. Она позволяет организовать безусловный цикл. Инструкция `loop` выполняется в два этапа. Сначала из регистра `ecx` 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`.

4 Выполнение лабораторной работы

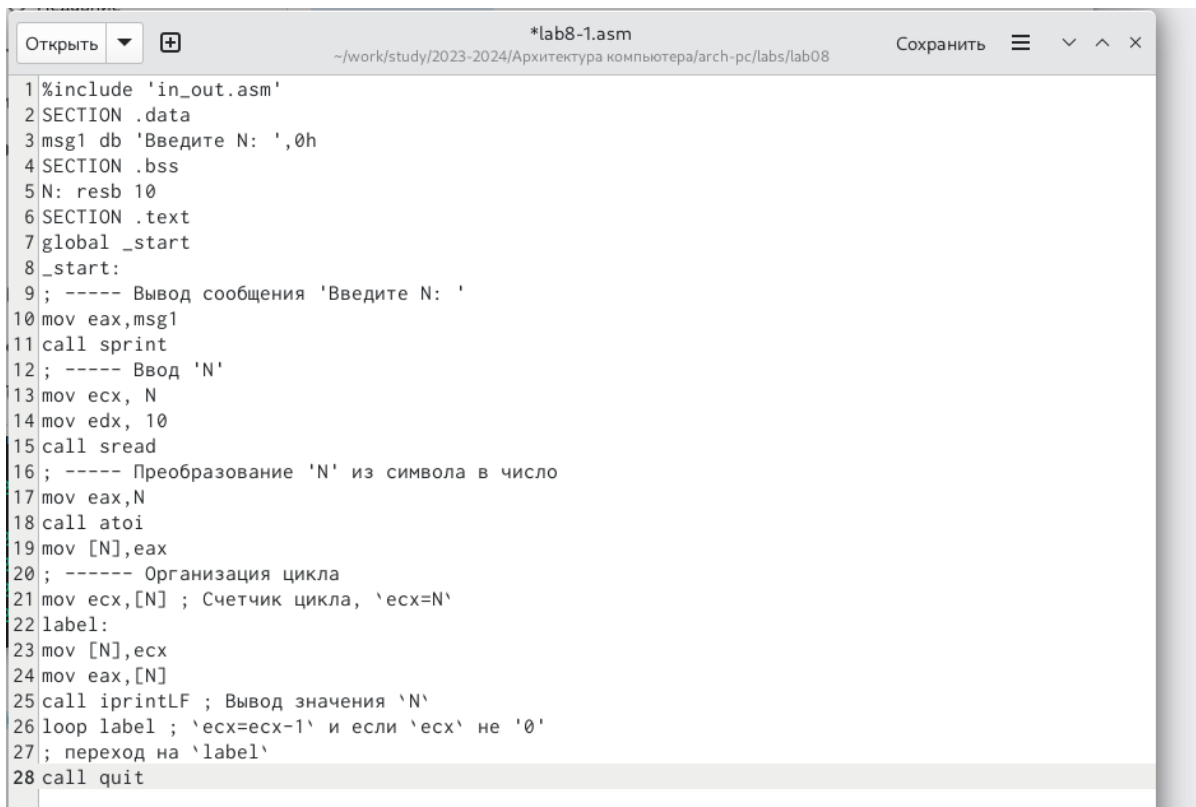
1) Реализация циклов в NASM

Перехожу в каталог, созданный для файлов с программами для лабораторной работы №8. С помощью `touch` создаю файл `lab8-1.asm`. Копирую в текущий каталог файл `in_out.asm`, так как он будет использоваться в дальнейшем (рис. 1).

```
dsmanjovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ cd
dsmanjovskaya@dk8n72 ~ $ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08
dsmanjovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ touch lab8-1.asm
dsmanjovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 1: Перемещение между директориями, создание файла

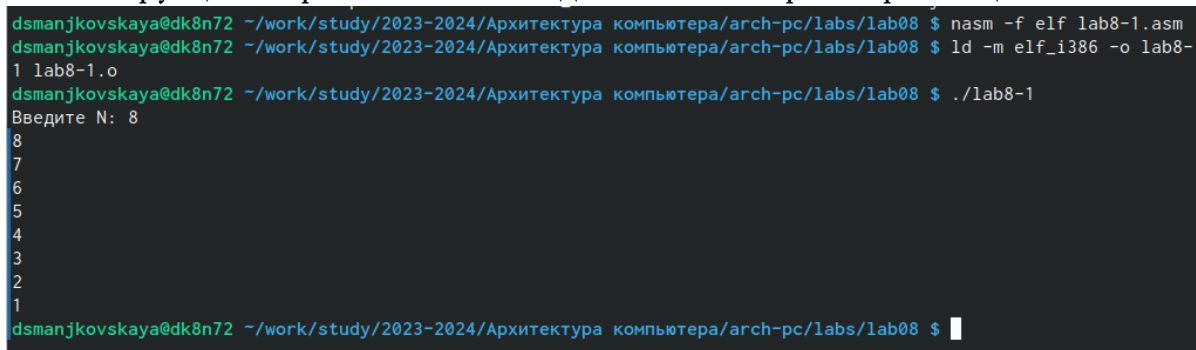
Открываю созданный файл `lab8-1.asm`, вставляю в него программу из листинга 8.1 (рис. 2)



```
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. 2: Редактирование файла

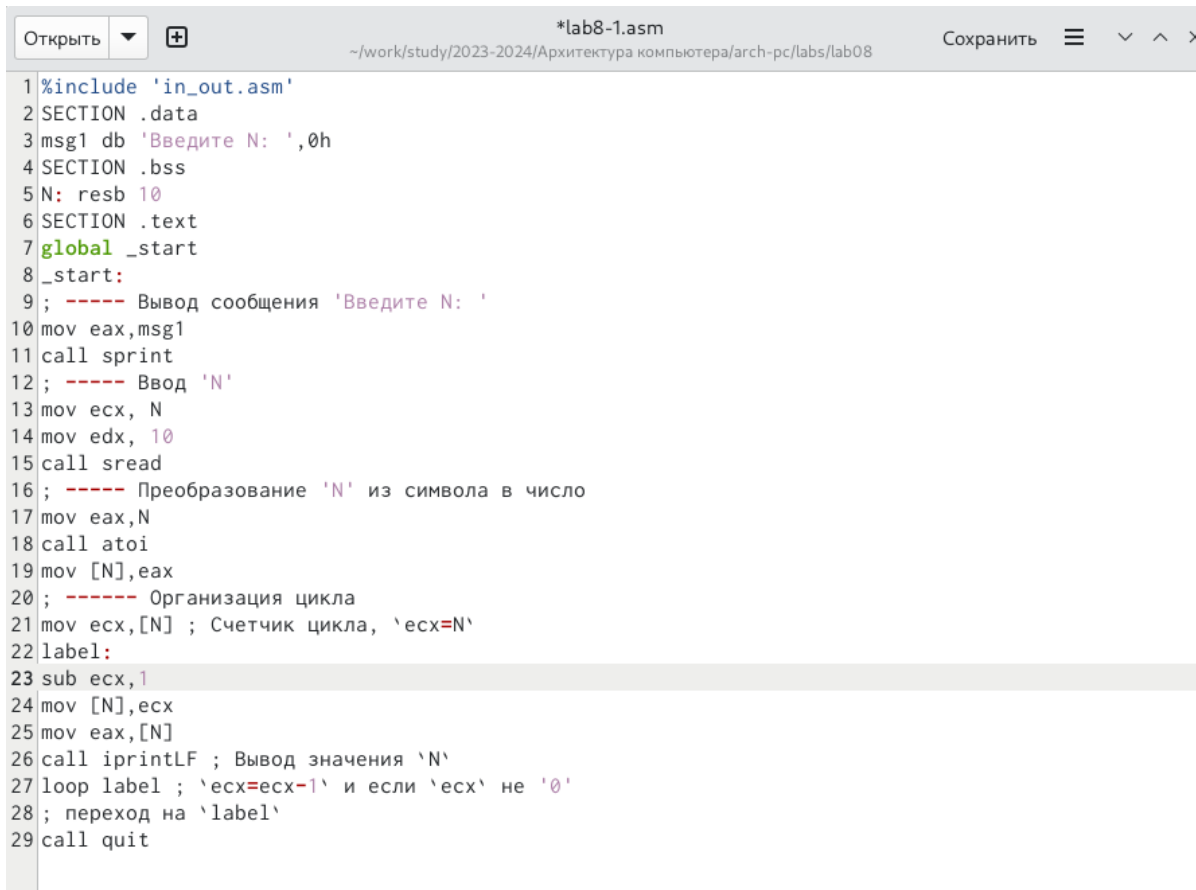
Создаю исполняемый файл и запускаю его (рис. 3). Мы видим, что использование инструкции `loop` позволяет выводить значения регистра `ecx` циклично.



```
dsmanjovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ nasm -f elf lab8-1.asm
dsmanjovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dsmanjovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-1
Введите N: 8
8
7
6
5
4
3
2
1
dsmanjovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 3: Запуск исполняемого файла

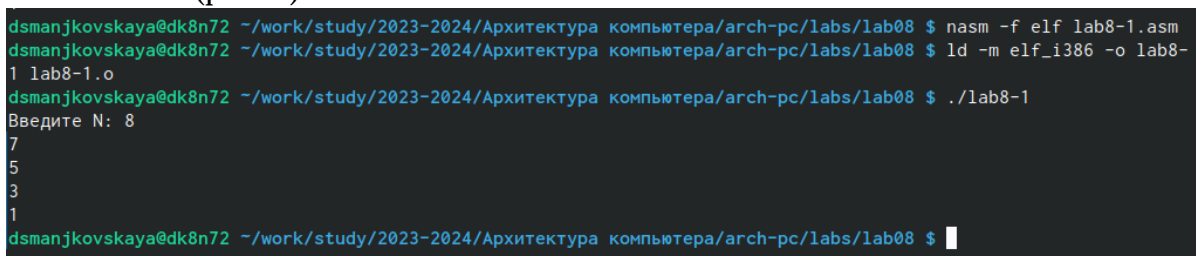
Изменяю значение `ecx` в цикле (рис. 4).



```
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
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF ; Вывод значения 'N'
27 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
28 ; переход на 'label'
29 call quit
```

Рис. 4: Редактирование файла

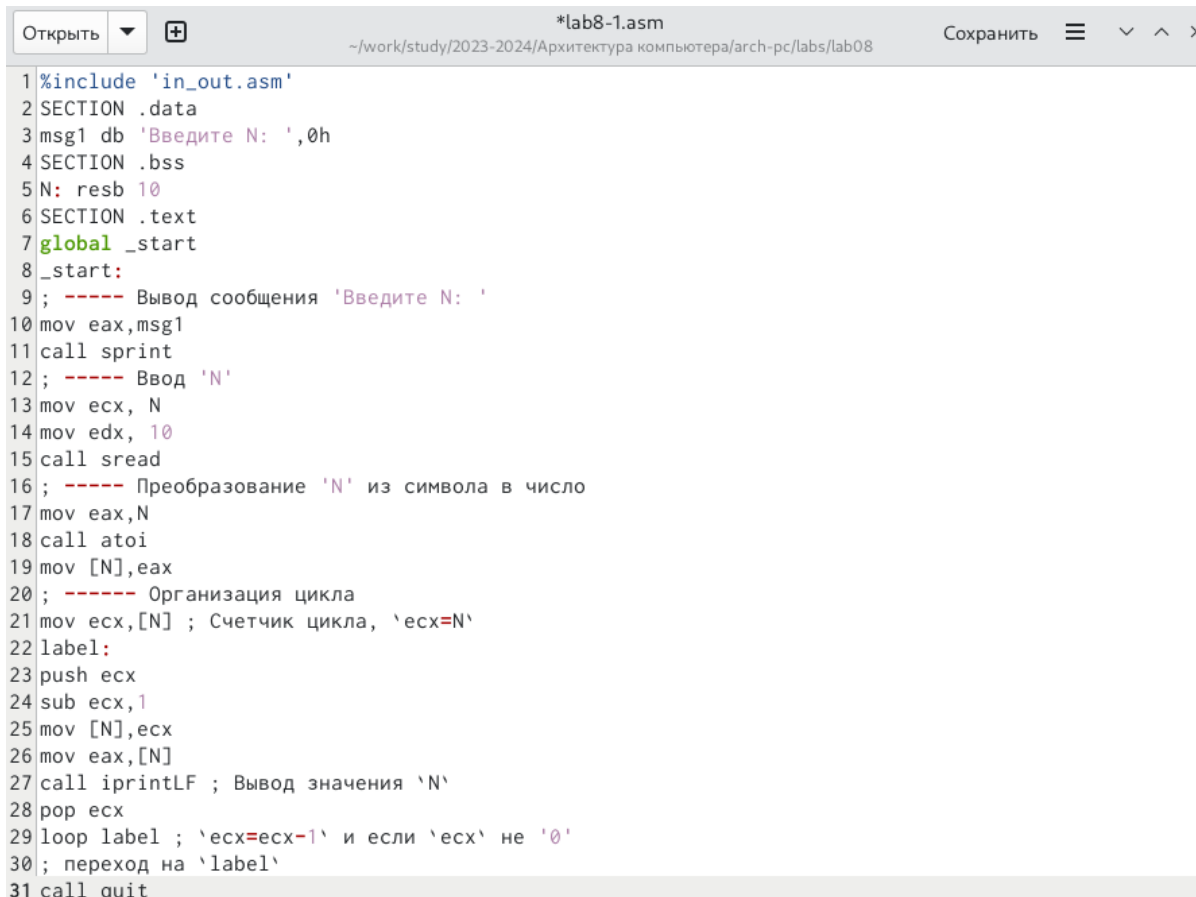
Создаю новый исполняемый файл и запускаю его. Видим, что регистр `ecx` в цикле принимает разные значения, а число проходов цикла не соответствует значению `N` (рис. 5).



```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ nasm -f elf lab8-1.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-1
Введите N: 8
7
5
3
1
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 5: Запуск исполняемого файла

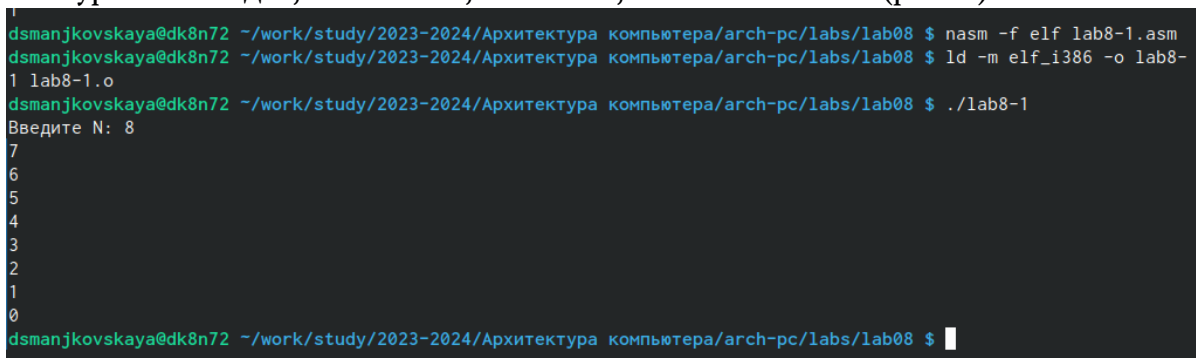
Вношу изменения в текст программы, добавив команды `push`, `pop` для сохранения значения счётчика цикла `loop` (рис. 6).



```
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 push ecx
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF ; Вывод значения 'N'
28 pop ecx
29 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
30 ; переход на 'label'
31 call quit
```

Рис. 6: Редактирование файла

Выполняю компиляцию и компоновку, и запускаю исполняемый файл. В данном случае число проходов цикла соответствует значению N, введенному с клавиатуры. Счёт идёт, не от 8-ми, а от 7-ми, но включается 0 (рис. 7)



```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ nasm -f elf lab8-1.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-1
Введите N: 8
7
6
5
4
3
2
1
0
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 7: Запуск исполняемого файла

2) Обработка аргументов командной строки

Создаю файл lab8-2.asm (рис. 8).


```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ touch lab8-2.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 8: Создание файла

Редактирую его, вводя программу из листинга 8.2 (рис. 9).

```
lab8-2.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08
Сохранить

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. 9: Редактирование файла

Создаю исполняемый файл и запускаю его. Вижу, что программа обработала 3 аргумента, указанные мною при запуске (рис. 10)

```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ nasm -f elf lab8-2.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-2
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-2 3 9 '1'
3
9
1
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

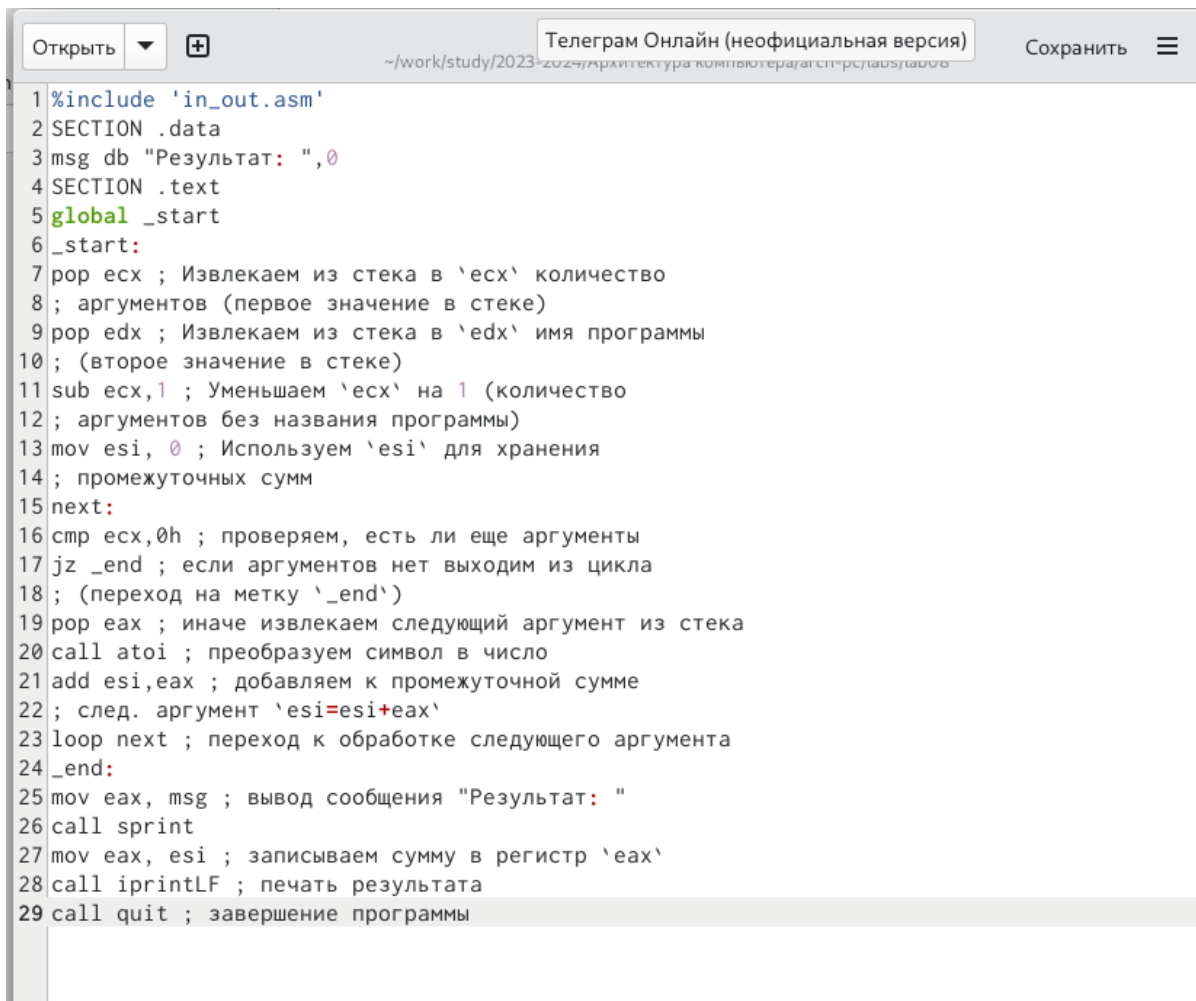
Рис. 10: Запуск исполняемого файла

Создаю файл lab8-3.asm (рис. 11).

```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ touch lab8-3.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 11: Создание файла

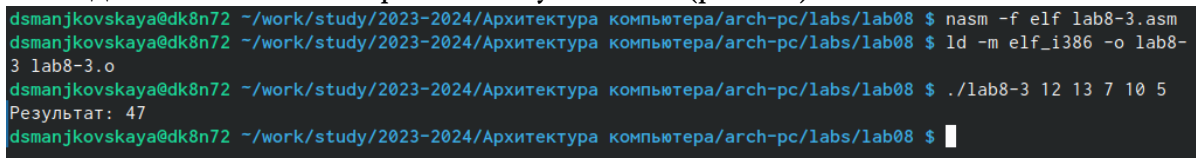
Ввожу в него программу из листинга 8.3 (рис. 12)



```
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. 12: Редактирование файла

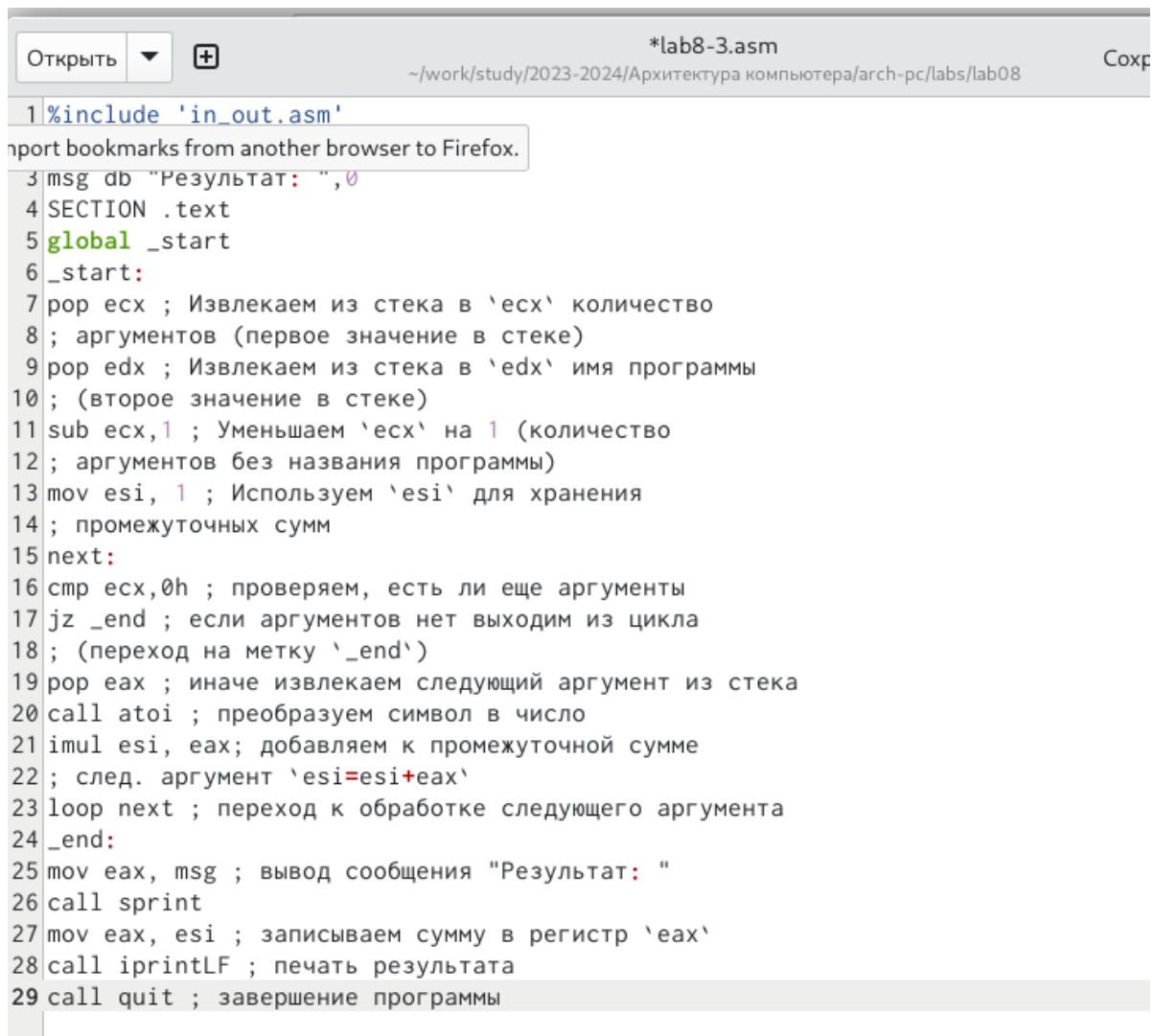
Создаю исполняемый файл и запускаю его (рис. 13)



```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ nasm -f elf lab8-3.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 13: Запуск исполняемого файла

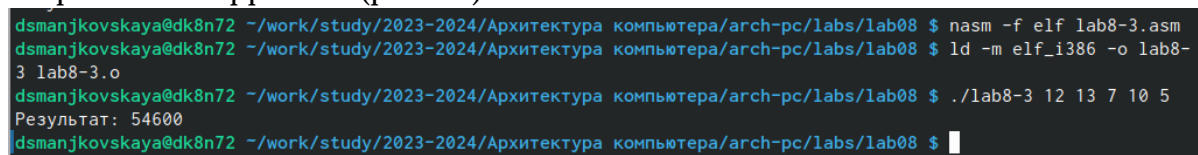
Изменяю текст программы для вычисления произведения аргументов командной строки (рис. 14)



```
1 %include 'in_out.asm'
2
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 imul 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. 14: Редактирование файла

Запускаю исполняемый файл, вижу, что выводится верное значение, программа работает корректно (рис. 15)

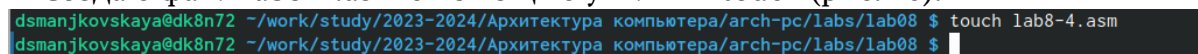


```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ nasm -f elf lab8-3.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 54600
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 15: Запуск исполняемого файла

3) Выполнение заданий для самостоятельной работы

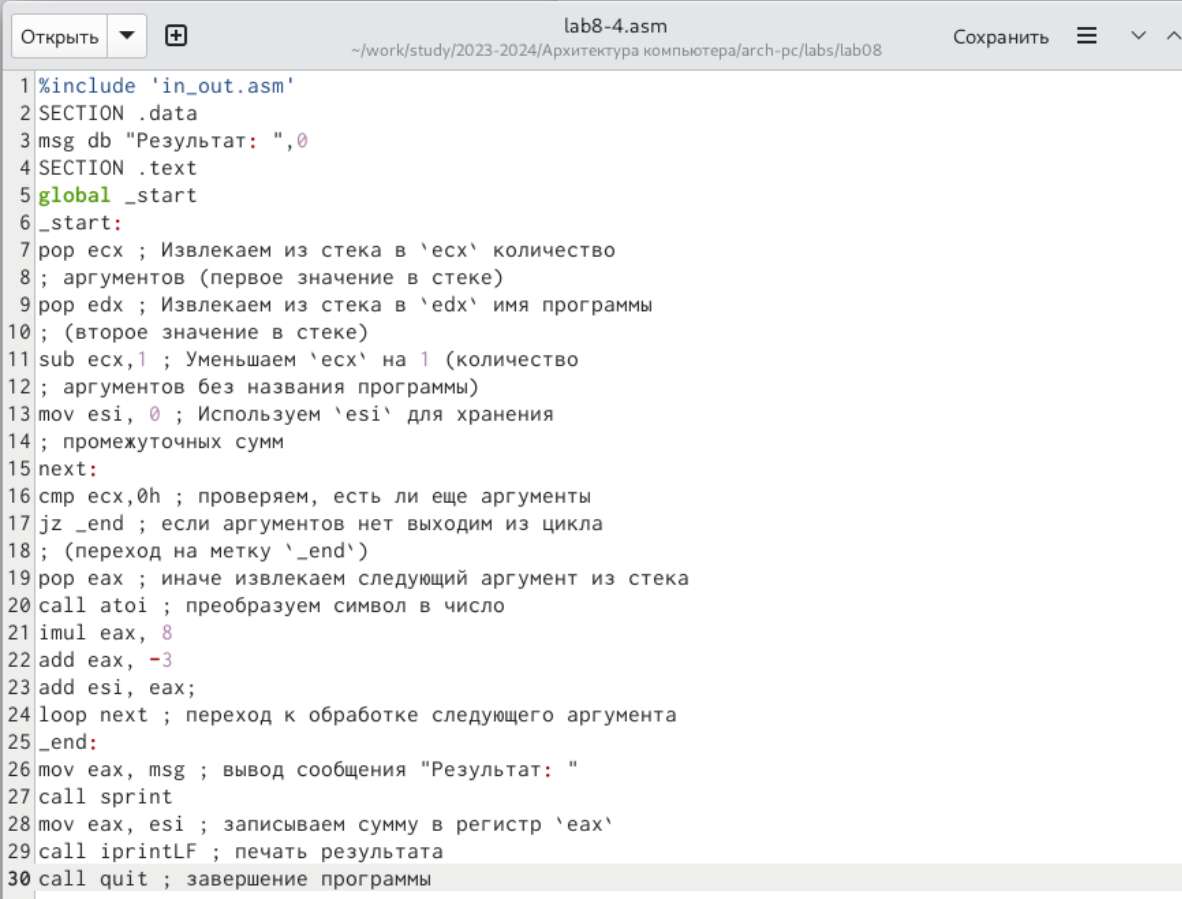
Создаю файл lab8-4.asm с помощью утилиты touch (рис. 16).



```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ touch lab8-4.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 16: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для суммирования значений функции, предложенной в варианте 19, полученным мною при выполнении лабораторной работы №7 (рис. 17)

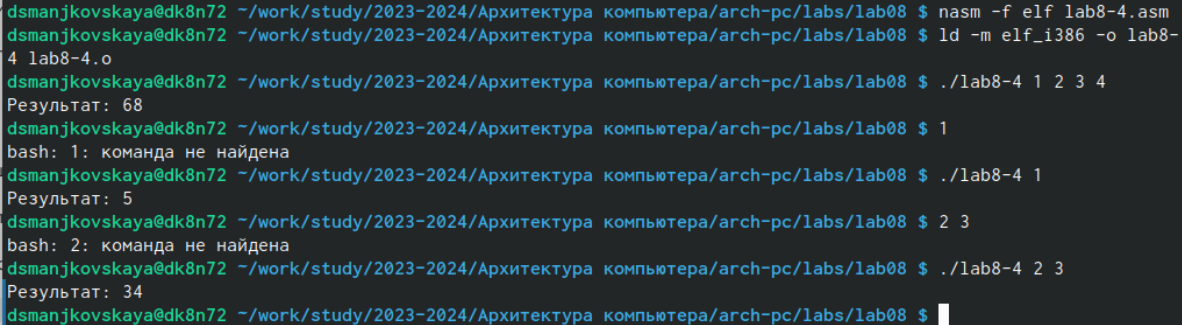


```
lab8-4.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08
Сохранить

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 imul eax, 8
22 add eax, -3
23 add 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. 17: Редактирование файла

Запускаю исполняемый файл, выполняю проверку и понимаю, что написанная мной программа работает верно (рис. 18).



```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ nasm -f elf lab8-4.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-4 1 2 3 4
Результат: 68
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ 1
bash: 1: команда не найдена
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-4 1
Результат: 5
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ 2 3
bash: 2: команда не найдена
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $ ./lab8-4 2 3
Результат: 34
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08 $
```

Рис. 18: Запуск исполняемого файла

Листинг 4.1 - Программа для суммирования нескольких значений функции, предложенной в варианте 19.

```
%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
imul eax, 5
add eax, 17
add 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

5 Выводы

При выполнении данной лабораторной работы я приобрела практический опыт в написании программ с использованием циклов и обработкой аргументов командной строки.

6 Список литературы

Архитектура ЭВМ