

# **Отчёт по Лабораторной работе №8**

**Дисциплина: Архитектура компьютера**

Гозенко А.С.

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выполнение задания для самостоятельной работы	11
4	Выводы	13

## Список иллюстраций

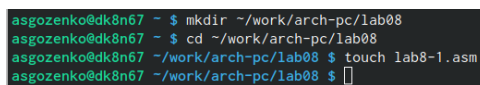
2.1	Создание файла . . . . .	5
2.2	Программа . . . . .	6
2.3	Работа программы . . . . .	6
2.4	Работа Программы . . . . .	6
2.5	Программа . . . . .	7
2.6	Работа программы . . . . .	7
2.7	Программа . . . . .	8
2.8	Работа программы . . . . .	8
2.9	Программа . . . . .	9
2.10	Работа программы . . . . .	9
2.11	Программа . . . . .	10
2.12	Работа программы . . . . .	10
3.1	Программа . . . . .	11
3.2	Работа программы . . . . .	12

# 1 Цель работы

Получение навыков по организации циклов и работе со стеком на языке NASM.

## 2 Выполнение лабораторной работы

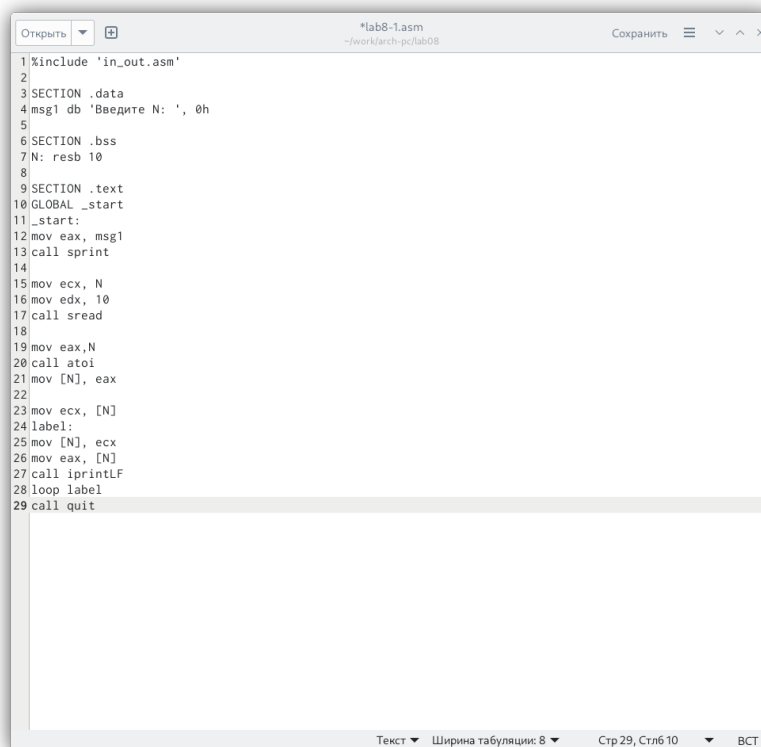
Создаю каталог lab08 и файл lab8-1.asm (рис. 2.1).

A terminal window showing four lines of commands and their outputs. The first line creates a directory, the second changes to it, the third creates a file, and the fourth shows the current directory.

```
asgozenko@dk8n67 ~ $ mkdir ~/work/arch-pc/lab08
asgozenko@dk8n67 ~ $ cd ~/work/arch-pc/lab08
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ touch lab8-1.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $
```

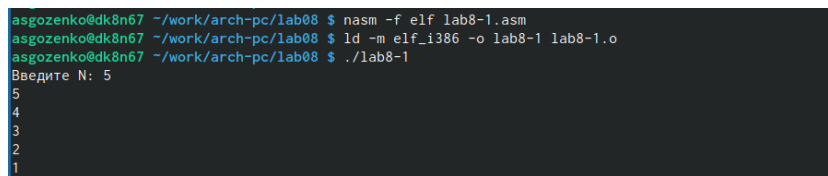
Рис. 2.1: Создание файла

Создаю файл с программой из листинга 1 (рис. 2.2) и проверяю её работу (рис. 2.3).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите N: ', 0h
5
6 SECTION .bss
7 N: resb 10
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg1
13 call sprint
14
15 mov ecx, N
16 mov edx, 10
17 call sread
18
19 mov eax, N
20 call atoi
21 mov [N], eax
22
23 mov ecx, [N]
24 label:
25 mov [N], ecx
26 mov eax, [N]
27 call iprintLF
28 loop label
29 call quit
```

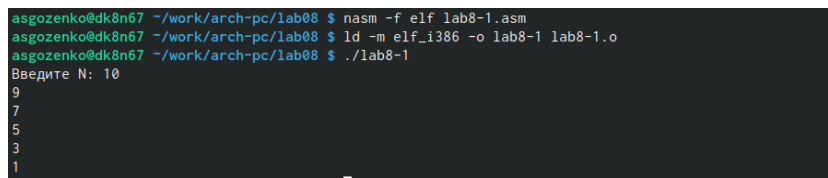
Рис. 2.2: Программа



```
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рис. 2.3: Работа программы

Вношу изменения в текст программы и проверяю её работу (рис. 2.4).



```
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
7
5
3
1
```

Рис. 2.4: Работа Программы

Внесу изменения в текст программы, чтобы выводились числа от 9 до 0 (рис. 2.5) и проверяю её работу (рис. 2.6).

```

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
23
24 label:
25 sub ecx,1 ; 'ecx=ecx-1'
26 mov [N],ecx
27 mov eax,[N]
28 call iprintlnLF
29 loop label
30 call quit
  
```

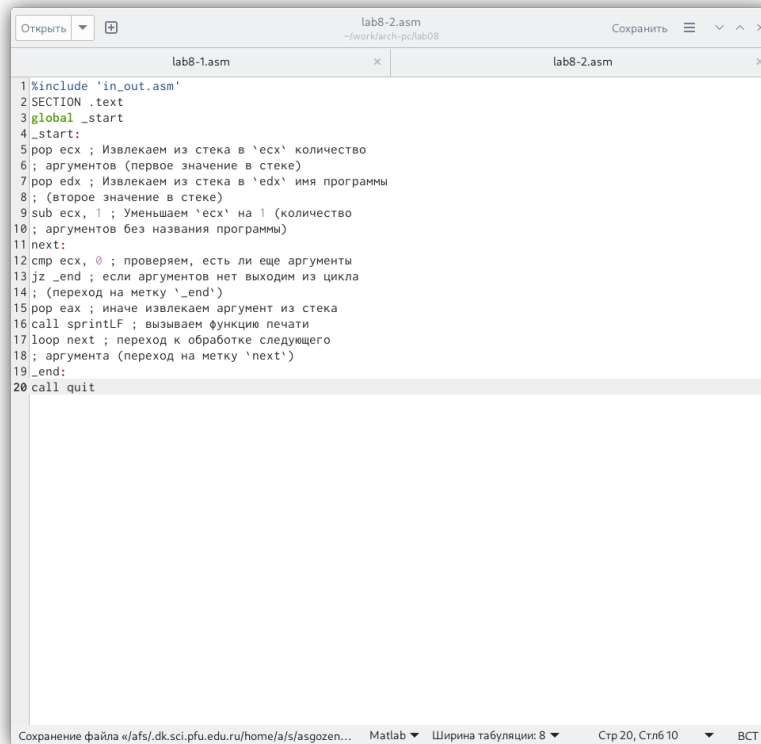
Рис. 2.5: Программа

```

asgozenko@dk8n67 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
  
```

Рис. 2.6: Работа программы

Создаю файл с программой из листинга 2 (рис. 2.7) и проверяю её работу (рис. 2.8). Было обработано 6 аргументов.

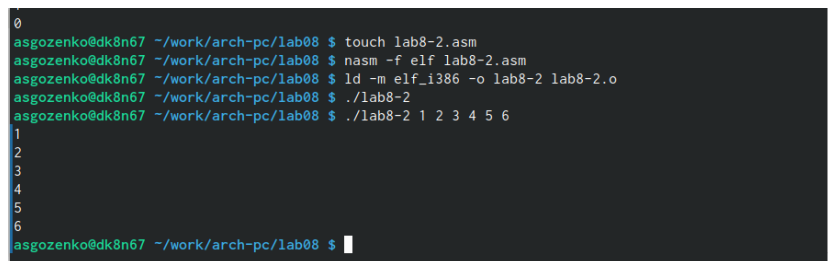


```
lab8-2.asm
~/work/arch-pc/lab08

lab8-1.asm x lab8-2.asm x

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 sprintf ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку 'next')
19 _end:
20 call quit
```

Рис. 2.7: Программа

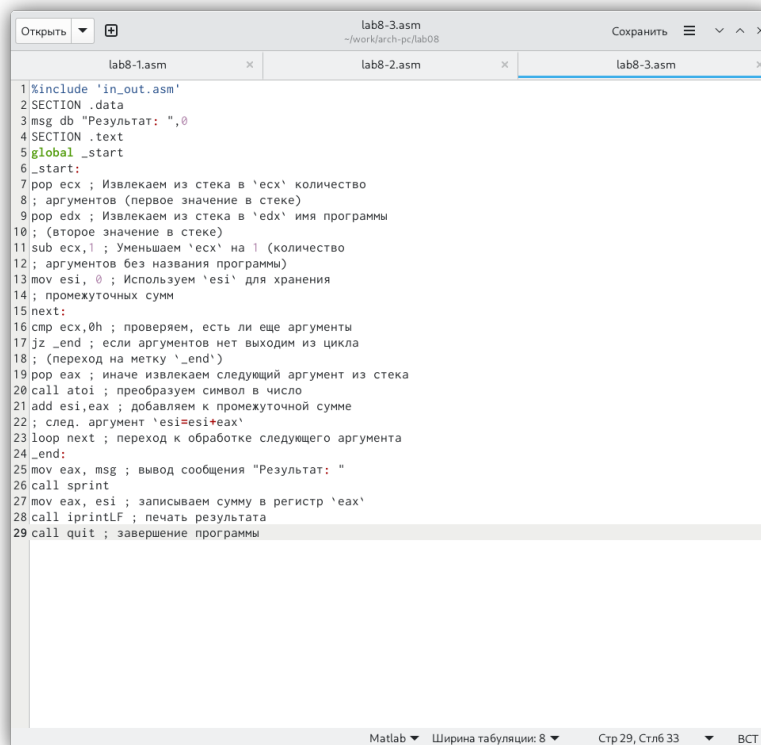


```
0
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ touch lab8-2.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-2
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-2 1 2 3 4 5 6
1
2
3
4
5
6
asgozenko@dk8n67 ~/work/arch-pc/lab08 $
```

Рис. 2.8: Работа программы

Создаю файл с программой из листинга 3 (рис. 2.9) и проверяю её работу (рис. 2.10).





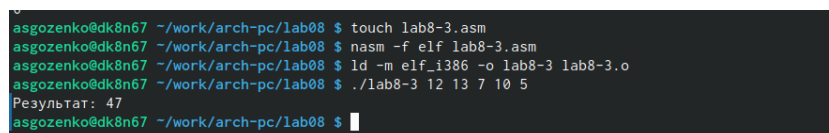
```
lab8-3.asm
~/work/arch-pc/lab08

lab8-1.asm x lab8-2.asm x lab8-3.asm x

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 iprintf ; печать результата
29 call quit ; завершение программы

Matlab Ширина табуляции: 8 Стр 29, Стлб 33 ВСТ
```

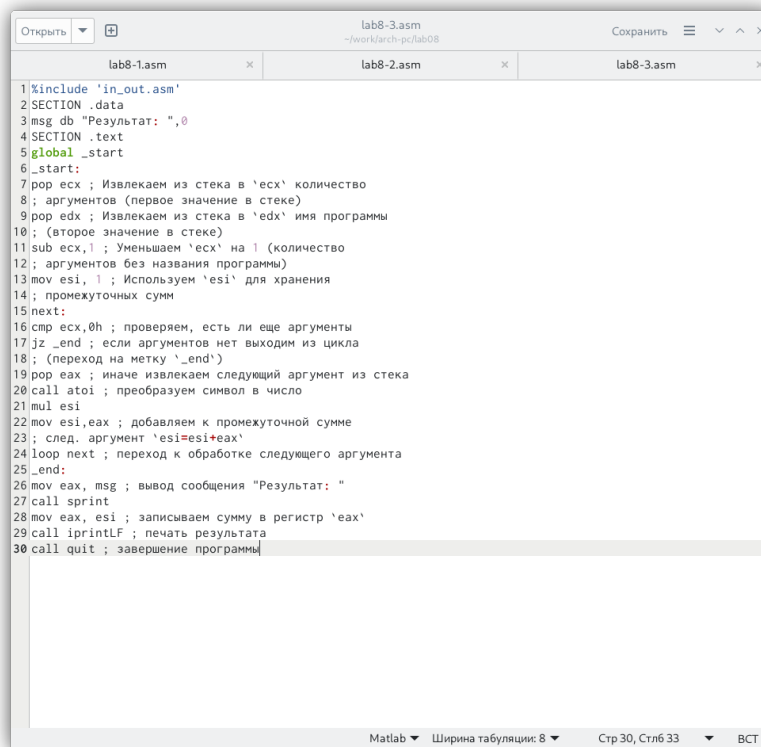
Рис. 2.9: Программа



```
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ touch lab8-3.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
asgozenko@dk8n67 ~/work/arch-pc/lab08 $
```

Рис. 2.10: Работа программы

Вношу изменения в текст программы для вычисления произведения аргументов командной строки (рис. 2.11) и проверяю её работу (рис. 2.12).

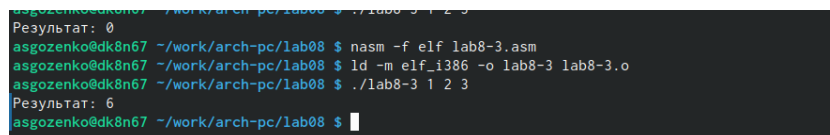


```
lab8-3.asm
~/work/arch-pc/lab08

lab8-1.asm x lab8-2.asm x lab8-3.asm x

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 iprintf ; печать результата
30 call quit ; завершение программы
```

Рис. 2.11: Программа



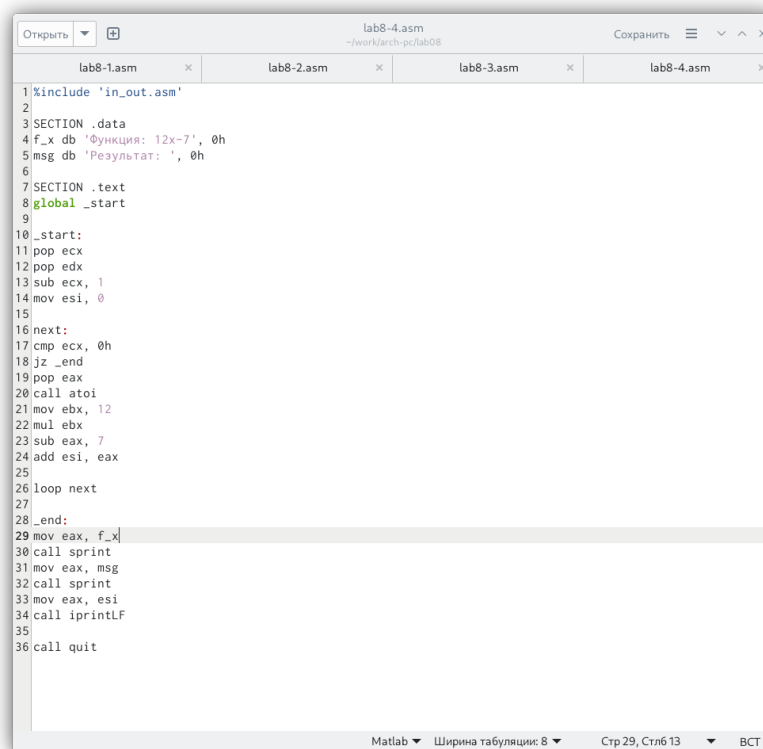
```
asgozenko@dk8n67: ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3
Результат: 0
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3
Результат: 6
asgozenko@dk8n67 ~/work/arch-pc/lab08 $
```

Рис. 2.12: Работа программы

## 3 Выполнение задания для самостоятельной работы

Вариант 13

Создаю файл с программой которая выполняет условие (рис. 3.1) и проверяю её работу (рис. 3.2).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 f_x db 'Функция: 12х-7', 0h
5 msg db 'Результат: ', 0h
6
7 SECTION .text
8 global _start
9
10 _start:
11 pop ecx
12 pop edx
13 sub ecx, 1
14 mov esi, 0
15
16 next:
17 cmp ecx, 0h
18 jz _end
19 pop eax
20 call atoi
21 mov ebx, 12
22 mul ebx
23 sub eax, 7
24 add esi, eax
25
26 loop next
27
28 _end:
29 mov eax, f_x
30 call sprint
31 mov eax, msg
32 call sprint
33 mov eax, esi
34 call iprintLF
35
36 call quit
```

Рис. 3.1: Программа

```
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3
Функция: 12х-7Результат: 51
asgozenko@dk8n67 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4 5 6
Функция: 12х-7Результат: 210
asgozenko@dk8n67 ~/work/arch-pc/lab08 $
```

Рис. 3.2: Работа программы

## 4 Выводы

Были получены навыки по организации циклов и работе со стеком на языке NASM.