

Отчёт по лабораторной работе 8

Архитектура компьютера

Довран Илиев

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Выводы	19

Список иллюстраций

2.1 Программа lab8-1.asm	7
2.2 Запуск программы lab8-1.asm	8
2.3 Программа lab8-1.asm	9
2.4 Запуск программы lab8-1.asm	10
2.5 Программа lab8-1.asm	11
2.6 Запуск программы lab8-1.asm	12
2.7 Программа lab8-2.asm	13
2.8 Запуск программы lab8-2.asm	13
2.9 Программа lab8-3.asm	14
2.10 Запуск программы lab8-3.asm	14
2.11 Программа lab8-3.asm	15
2.12 Запуск программы lab8-3.asm	16
2.13 Программа lab8-4.asm	17
2.14 Запуск программы lab8-4.asm	18

Список таблиц

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки..

2 Выполнение лабораторной работы

1. Создал папку для файлов лабораторной № 8, перешел в нее и инициировал создание файла с именем lab8-1.asm.
2. Внес программный код в файл lab8-1.asm, соответствующий листингу 8.1. Скомпилировал его в исполняемый файл и осуществил проверку функционирования.

The screenshot shows a text editor window with the following details:

- File menu: Open, Save.
- Title bar: lab8-1.asm, ~/work/arch-pc/lab08.
- Code area:

```
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. 2.1: Программа lab8-1.asm

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

Рис. 2.2: Запуск программы lab8-1.asm

3. Приведенный пример иллюстрирует, что манипулирование регистром `esx` внутри цикла `loop` может привести к ошибкам в работе программы. Модифицировал код, включив изменение значения регистра `esx` в процессе выполнения цикла:

При нечетном `N` программа запускается в бесконечный цикл, а при четном `N` выводит исключительно нечетные числа.

The screenshot shows a text editor window with the following details:

- File menu: Open, Save.
- Title bar: lab8-1.asm, ~/work/arch-pc/lab08.
- Code area:

```
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 iprintfLF
27 loop label
28 ; переход на `label`
29 call quit
```

Рис. 2.3: Программа lab8-1.asm

```
4294928672
4294928670
4294928668
42949^C
dovranivlev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
dovranivlev@VirtualBox:~/work/arch-pc/lab08$ █
```

Рис. 2.4: Запуск программы lab8-1.asm

4. Чтобы корректно использовать регистр `esx` внутри цикла, применим стек. Внес изменения в код программы, добавив команды `push` и `pop` для сохранения и восстановления значения счетчика цикла `loop`. Скомпилировал исполняемый файл и провел проверку его функционирования.

Программа отображает числа от $N-1$ до 0, количество итераций соответствует введенному числу N .

The screenshot shows a debugger interface with the following details:

- File Menu:** Open, Save.
- Title Bar:** lab8-1.asm, ~/work/arch-pc/lab08
- Code Area:** Assembly code for lab8-1.asm. The code is color-coded:
 - Directive: %include 'in_out.asm'
 - Section: SECTION .data
 - Variable: msg1 db 'Введите N: ',0h
 - Section: SECTION .bss
 - Variable: N: resb 10
 - Section: SECTION .text
 - Label: global _start
 - Label: _start:
 - Comment: ; ----- Вывод сообщения 'Введите N: '
 - Instruction: mov eax,msg1
 - Instruction: call sprint
 - Comment: ; ----- Ввод 'N'
 - Instruction: mov ecx, N
 - Instruction: mov edx, 10
 - Instruction: call sread
 - Comment: ; ----- Преобразование 'N' из символа в число
 - Instruction: mov eax,N
 - Instruction: call atoi
 - Instruction: mov [N],eax
 - Comment: ; ----- Организация цикла
 - Instruction: mov ecx,[N] ; Счетчик цикла, `ecx=N`
 - Label: label:
 - Instruction: push ecx ; добавление значения ecx в стек
 - Instruction: sub ecx,1
 - Instruction: mov [N],ecx
 - Instruction: mov eax,[N]
 - Instruction: call iprintLF
 - Instruction: pop ecx ; извлечение значения ecx из стека
 - Label: loop label
 - Instruction: call quit

Рис. 2.5: Программа lab8-1.asm

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

Рис. 2.6: Запуск программы lab8-1.asm

5. Сформировал файл lab8-2.asm в директории ~/work/arch-pc/lab08 и заполнил его кодом из листинга 8.2. Скомпилировал в исполняемый файл и выполнил его с аргументами.

Программа обработала пять аргументов.

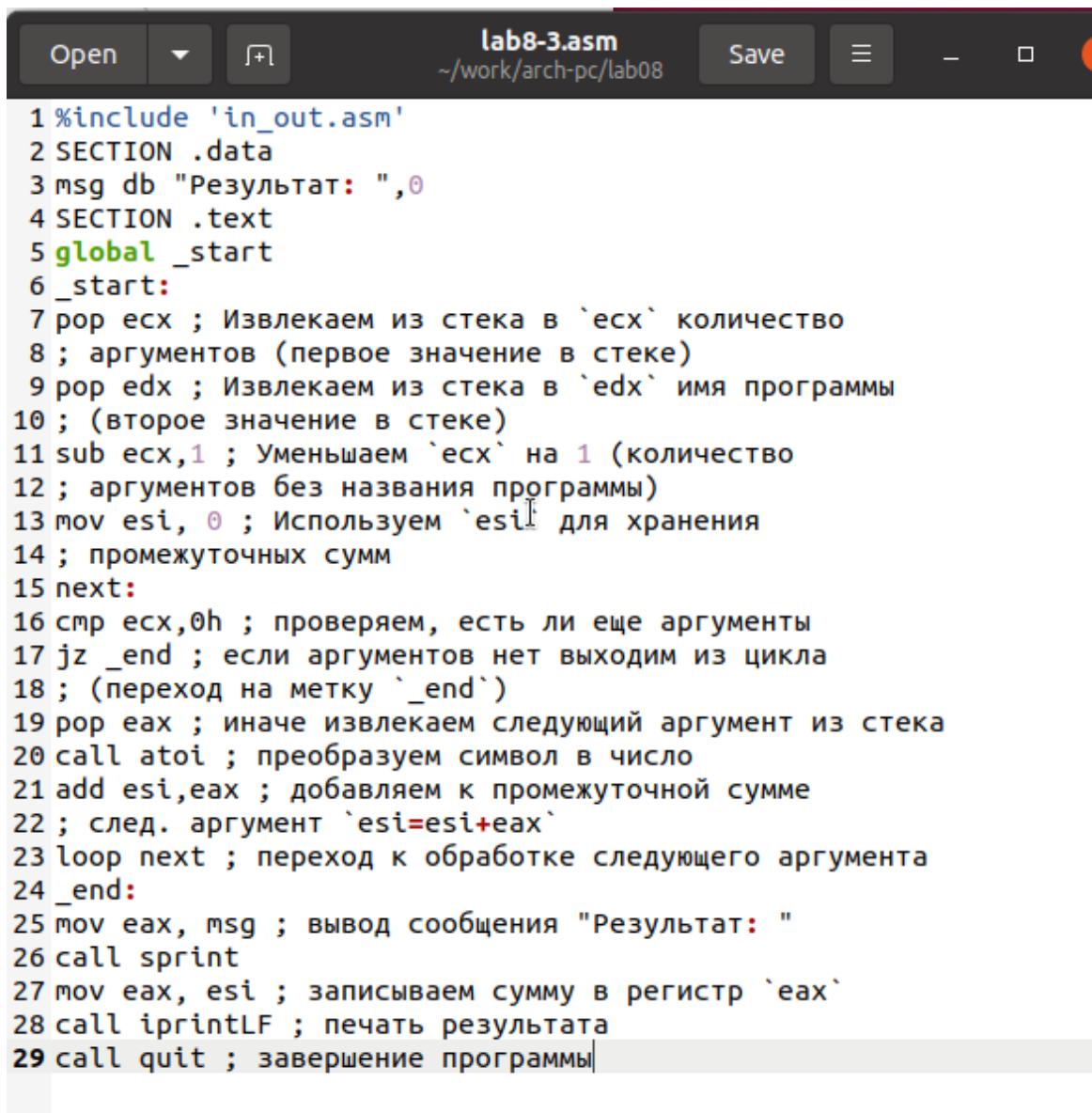
```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов 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. 2.7: Программа lab8-2.asm

```
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-2
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 argument 1 argument 2 'arg
ument 3'
argument
1
argument
2
argument 3
dovraniviev@VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.8: Запуск программы lab8-2.asm

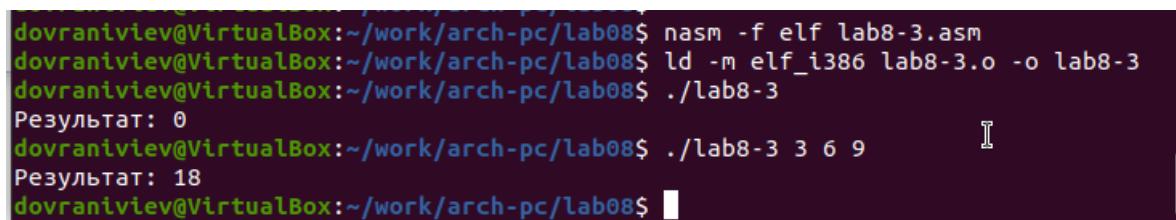
6. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.



The screenshot shows a text editor window with the file 'lab8-3.asm' open. The code is written in NASM assembly language. It includes sections for data and text, defines a global variable '_start', and implements a loop to sum up arguments passed on the stack. The code also handles the termination of the program.

```
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 iprintfLF ; печать результата
29 call quit ; завершение программы
```

Рис. 2.9: Программа lab8-3.asm

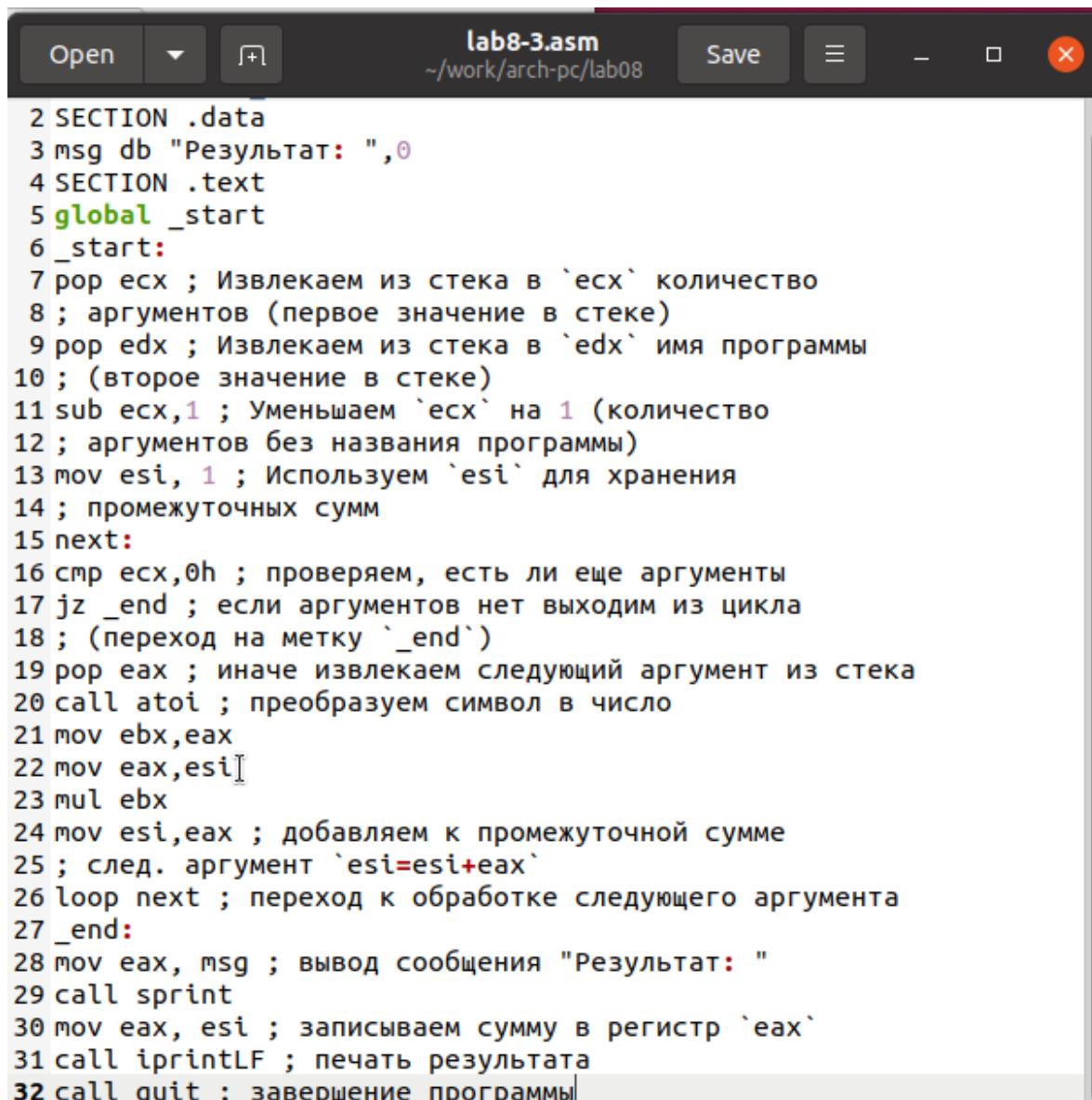


The terminal window shows the compilation of the assembly code into an ELF executable and its subsequent execution. The user enters the command 'nasm -f elf lab8-3.asm' to assemble the code. Then, 'ld -m elf_i386 lab8-3.o -o lab8-3' is run to link the object file into an executable. Finally, the program is executed with the command './lab8-3 3 6 9', which prints the result 'Результат: 18' to the console.

```
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 3 6 9
Результат: 0
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 3 6 9
Результат: 18
dovraniviev@VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.10: Запуск программы lab8-3.asm

7. Переписал код программы из листинга 8.3 так, чтобы она вычисляла произведение переданных через командную строку аргументов.



The screenshot shows a text editor window with the title bar "lab8-3.asm" and the path " ~/work/arch-pc/lab08". The code is written in assembly language:

```
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 mov ebx,eax
22 mov eax,esi]
23 mul ebx
24 mov esi,eax ; добавляем к промежуточной сумме
25 ; след. аргумент `esi=esi+eax`
26 loop next ; переход к обработке следующего аргумента
27 _end:
28 mov eax, msg ; вывод сообщения "Результат: "
29 call sprint
30 mov eax, esi ; записываем сумму в регистр `eax`
31 call iprintfLF ; печать результата
32 call quit ; завершение программы
```

Рис. 2.11: Программа lab8-3.asm

```
dovranivlev@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dovranivlev@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
dovranivlev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3
Результат: 1
dovranivlev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 3 6 9
Результат: 162
dovranivlev@VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.12: Запуск программы lab8-3.asm

8. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу на нескольких наборах x .

для варианта 5 $f(x) = 4x + 3$

The screenshot shows a text editor window with the following details:

- File menu: Open, Save, etc.
- Title bar: lab8-4.asm, ~work/arch-pc/lab08
- Code area:

```
3 msg db "Результат: ",0
4 fx: db 'f(x)= 4x + 3',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintLF
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,4
22 mul ebx
23 add eax,3
24 add esi,eax
25
26 loop next
27
28 _end:
29 mov eax, msg
30 call sprint
31 mov eax, esi
32 call iprintLF
33 call quit|
```

Рис. 2.13: Программа lab8-4.asm

```
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-4
f(x)= 4x + 3
Результат: 0
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 1
f(x)= 4x + 3
Результат: 7
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 1 3 6 9 8 7
f(x)= 4x + 3
Результат: 154
dovraniviev@VirtualBox:~/work/arch-pc/lab08$ █
```

Рис. 2.14: Запуск программы lab8-4.asm

3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере nasm.