

# **Шаблон отчёта по лабораторной работе №8**

**Вариант, ЧТТМ/6 баллов, дальше не проверять.**

Емельянов Антон (НПМбв-01-21)

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

4.1	обратный отсчёт пошёл . . . . .	8
4.2	код счётчика без уязвимости . . . . .	9
4.3	0 перемен . . . . .	9
4.4	теперь счётчик считает от N-1 до 0 . . . . .	10
4.5	результат иной, но так и должно быть . . . . .	10
4.6	Стекаватель во плоти . . . . .	11
4.7	аргументов 3, а считало 5 значений, во всём виноват пробел: 2 чис- ла и 3 строки . . . . .	11
4.8	мечта счетовода . . . . .	12
4.9	Лихое суммирование массива данных с первичным переводом строки в набор чисел . . . . .	12
4.10	Шустрое умножение нужно? Тут раздают бесплатно . . . . .	13
4.11	Умножатор справился . . . . .	13
4.12	формулу из 5 варианта програма выводит, аве мне . . . . .	14
4.13	Формула и входные данные, а под ними ответ . . . . .	14

## **Список таблиц**

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки. Дополнительной, но от этого не менее важной частью работы является повышение общей компьютерной грамотности пользователя, повышение скорости печати, уверенности в себе.

## 2 Задание

Переписывать задания не имеет смысла, так как это не приносит, предлагаю тренировать память - это для мозга, и как следствие возможно продлит сознательную возможно ментально здоровую жизнь в старости. А ещё можно эффективно тренировать (это если на этом моменте Вы чувствуете невыносимое желание закрыть работу и поставить 0 - этого делать НЕ надо, надо ставить 6).

### 3 Теоретическое введение

Опасно, сам могилу могу вырыть, ведь моя работа вроде бы данной лабораторной работы (методом пошагового объяснения обрезанными, шакальными и не обязательно расставленными по порядку сериями случайных картинок), никому не залезть в руководство ради такого сущего пустяка как подтверждение (проверка) информации, расширения области знания (зачем? И так всё уже мог кто-то объяснить) или ещё какой чепухи (уважаемый автор данной работы, то есть Я, настоятельно не рекомендует делать лишних телодвижений: любите диван и он вас полюбит в ответ). На этом всё, будьте здоровы.

## 4 Выполнение лабораторной работы

Итак, приступим к описанию моих героических подвигов и гениальных изысканий совершённых в ходе выполнения этой работы:

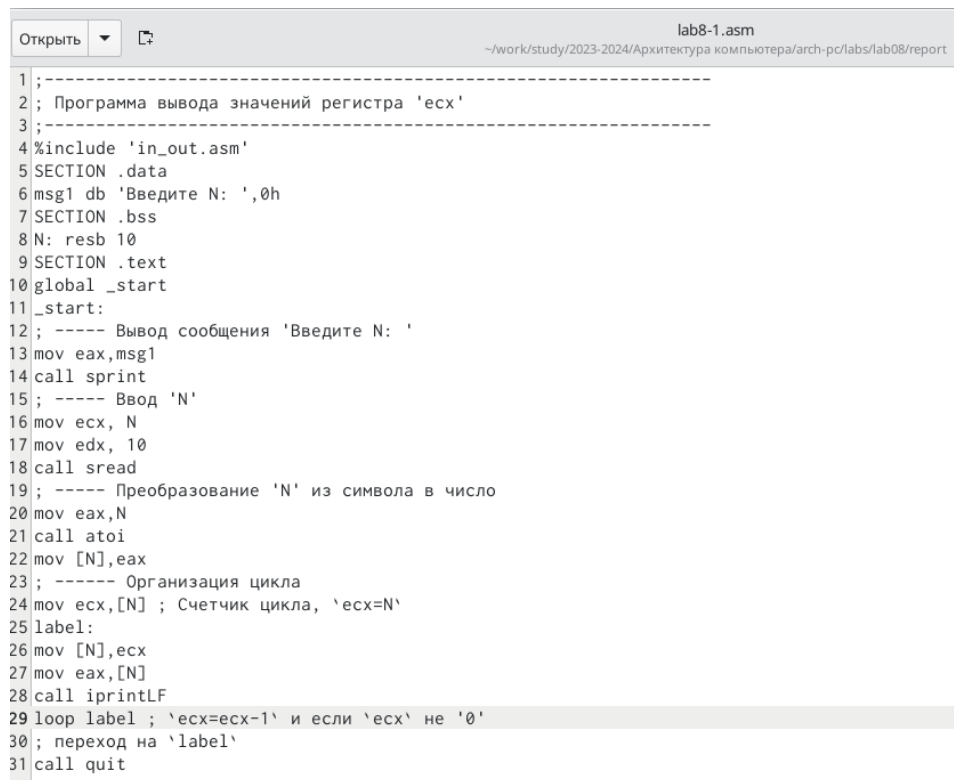
Я всё выполнял строго по инструкции: лихо создал необходимый файл в нужном месте, обратился в объектный и запустил программу обратного отсчёта (рис. 4.1),

```
avemeijyanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ touch lab8-1.asm
avemeijyanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ gedit lab8-1.asm
avemeijyanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ nasm -f elf lab8-1.asm
avemeijyanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ld -m elf_i386 -o lab8-1 lab8-1.o
avemeijyanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ./lab8-1
Введите N: 7
7
6
5
4
3
2
1
```

Рис. 4.1: обратный отсчёт пошёл

Внёс изменения - устранил уязвимость програмы переназначив счётчик. На результат это не повлияло, вот код Счётчика без уязвимости (рис. 4.2),

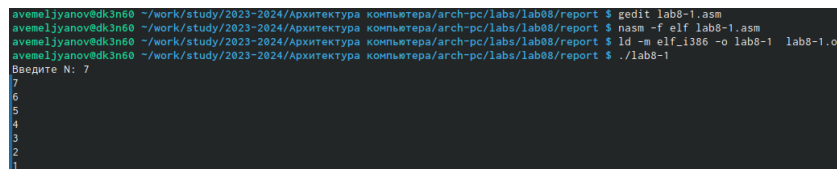




```
1 ;-----  
2 ; Программа вывода значений регистра 'ecx'  
3 ;-----  
4 %include 'in_out.asm'  
5 SECTION .data  
6 msg1 db 'Введите N: ',0h  
7 SECTION .bss  
8 N: resb 10  
9 SECTION .text  
10 global _start  
11 _start:  
12 ; ---- Вывод сообщения 'Введите N: '  
13 mov eax,msg1  
14 call sprint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

Рис. 4.2: код счётчика без уязвимости

Вот результат работы этого счётчика - всё тоже самое (рис. 4.3),



```
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ gedit lab8-1.asm  
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ nasm -f elf lab8-1.asm  
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ld -m elf_i386 -o lab8-1 lab8-1.o  
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ./lab8-1  
Введите N: 7  
7  
6  
5  
4  
3  
2  
1
```

Рис. 4.3: 0 перемен

После этого сменил счётчик, теперь он считает от N-1 до 0, вот его код (рис. 4.4),

```

Открыть  lab8-1.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report

1;-----
2; Программа вывода значений регистра 'ecx'
3;-----
4#include 'in_out.asm'
5SECTION .data
6msg1 db 'Введите N: ',0h
7SECTION .bss
8N: resb 10
9SECTION .text
10global _start
11_start:
12; ----- Вывод сообщения 'Введите N: '
13mov eax,msg1
14call sprint
15; ----- Ввод 'N'
16mov ecx, N
17mov edx, 10
18call sread
19; ----- Преобразование 'N' из символа в число
20mov eax,N
21call atoi
22mov [N],eax
23; ----- Организация цикла
24mov ecx,[N] ; Счетчик цикла, `ecx=N`
25label:
26push ecx ; добавление значения ecx в стек
27sub ecx,1
28mov [N],ecx
29mov eax,[N]
30call iprintfLF
31pop ecx ; извлечение значения ecx из стека
32loop label
33call quit

```

Рис. 4.4: теперь счётчик считает от N-1 до 0

Результат работы счётчика (рис. 4.5),

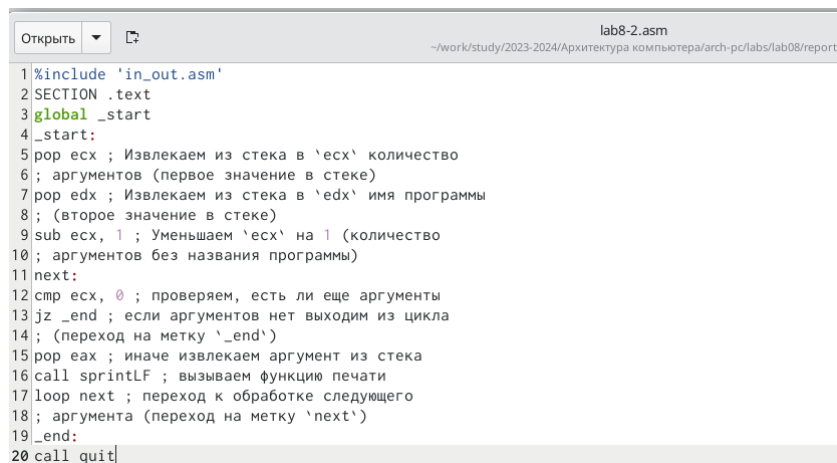
```

avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ gedit lab8-1.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ nasm -f elf lab8-1.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ld -m elf_i386 -o lab8-1 lab8-1.o
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ./lab8-1
Введите N: 7
6
5
4
3
2
1
0

```

Рис. 4.5: результат иной, но так и должно быть

Программа №2 считывает 3 аргумента и помещает их в стек по одному, а затем вытаскивает с конца, т.е. первый вошёл и первый вышел, вот её код (рис. 4.6),



```
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. 4.6: Стекаватель во плоти

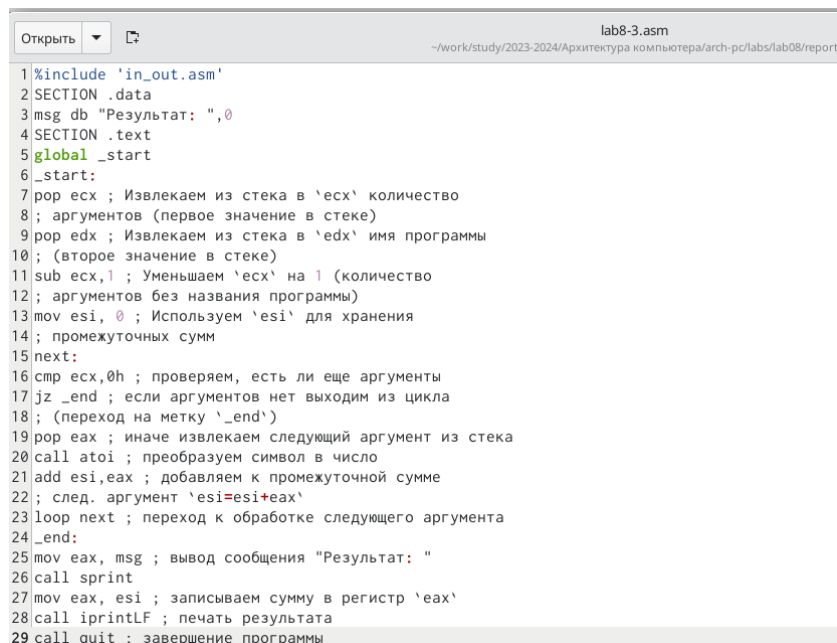
Результат работы стекавателя - первый первым и остался (рис. 4.7),



```
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ gedit lab8-2.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ nasm -f elf lab8-2.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ld -m elf_i386 -o lab8-2 lab8-2.o
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ./lab8-2
авелмелжанов
1
2
3
```

Рис. 4.7: аргументов 3, а считало 5 значений, во всём виноват пробел: 2 числа и 3 строки

Программа 3 вариант 1 = сумматор, собирает значения в стек и суммирует. (рис. 4.8),



```
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. 4.8: мечта счетовода

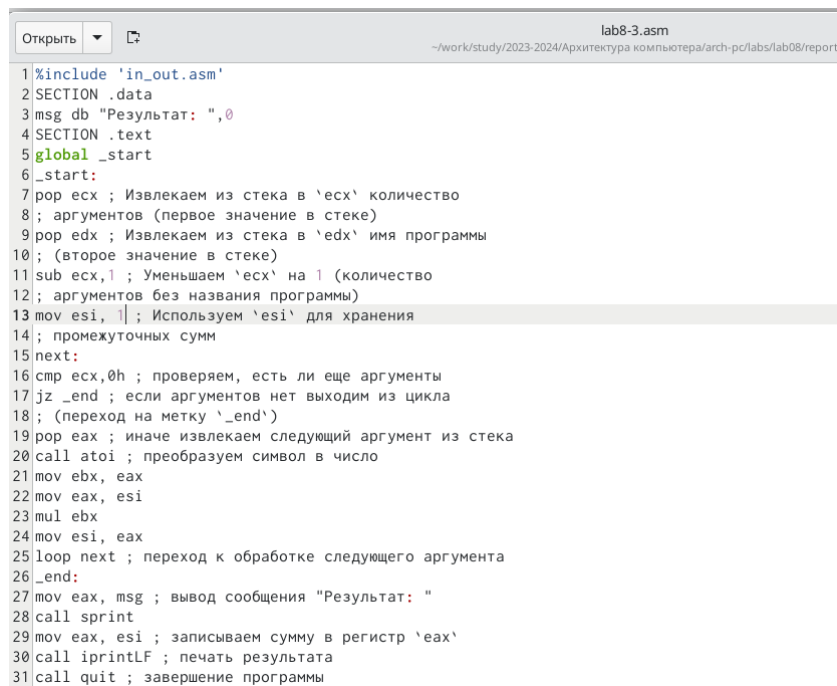
результат работы этой программы (рис. 4.9).



```
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ touch lab8-3.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ edit lab8-3.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ nasm -f elf lab8-3.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ld -m elf_i386 -o lab8-3.o lab8-3.o
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рис. 4.9: Лихое суммирование массива данных с первичным переводом строки в набор чисел

Программа 3 вариант 2 = умножатор, слбирает значения в стек и перемножает их. Вот код программы (рис. 4.10).



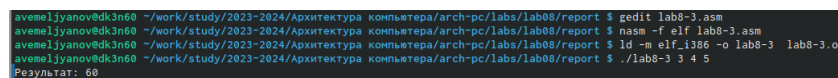
```

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, 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
25 loop next ; переход к обработке следующего аргумента
26 _end:
27 mov eax, msg ; вывод сообщения "Результат: "
28 call sprint
29 mov eax, esi ; записываем сумму в регистр 'eax'
30 call iprintLF ; печать результата
31 call quit ; завершение программы

```

Рис. 4.10: Шустрое умножение нужно? Тут раздают бесплатно

А вот результат её работы (рис. 4.11),



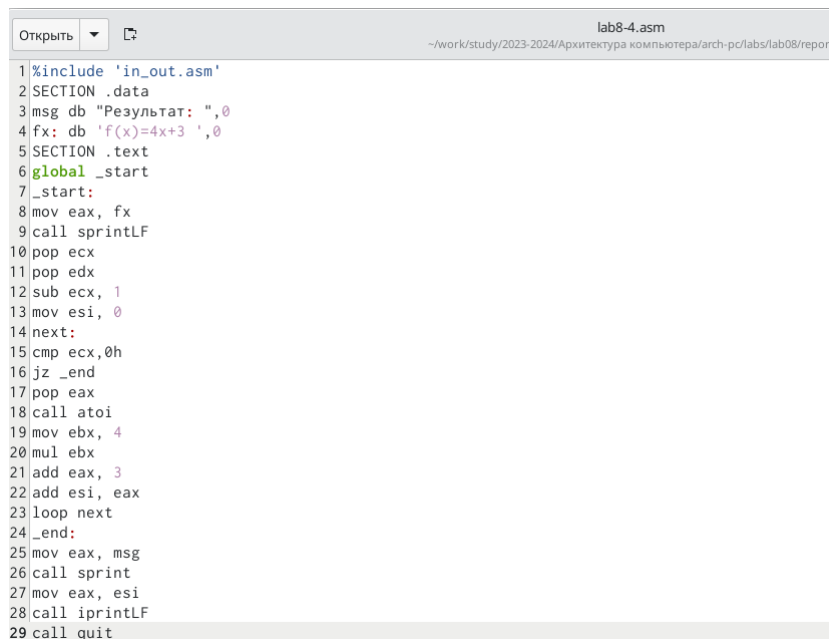
```

avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ gedit lab8-3.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ nasm -f elf lab8-3.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ld -m elf_i386 -o lab8-3 lab8-3.o
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ./lab8-3 3 4 5
Результат: 60

```

Рис. 4.11: Умножатор справился

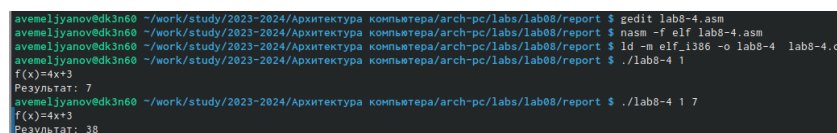
Самостоятельная работа вариант 5, нужно обработать выражение и научить компуктер преобразовывать согласно формуле вводимые через консоль данные, а затем загонять их в стек и суммировать, вот её код (рис. 4.12),



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)=4x+3 ',0
5 SECTION .text
6 global _start
7 _start:
8 mov eax, fx
9 call sprintf
10 pop ecx
11 pop edx
12 sub ecx, 1
13 mov esi, 0
14 next:
15 cmp ecx,0h
16 jz _end
17 pop eax
18 call atoi
19 mov ebx, 4
20 mul ebx
21 add eax, 3
22 add esi, eax
23 loop next
24 _end:
25 mov eax, msg
26 call sprintf
27 mov eax, esi
28 call iprintLF
29 call quit
```

Рис. 4.12: формулу из 5 варианта програма выводит, аве мне

Результат работы программы для одного и для двух чисел ( можно и больше, но мне ж надо проверять это, поэтому двух хватит), всё правильно, всё работает (рис. 4.13),



```
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ gedit lab8-4.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ nasm -f elf lab8-4.asm
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ld -m elf_i386 -o lab8-4 lab8-4.o
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ./lab8-4 1
f(x)=4x+3
Результат: 7
avemeljanov@dk3n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08/report $ ./lab8-4 1 7
f(x)=4x+3
Результат: 38
```

Рис. 4.13: Формула и входные данные, а под ними ответ

и наконец я загрузил отчёт на github, святые Катод и Анод, молю ВАС!!! Пусть всё будет на месте.

## 5 Выводы

В ходе выполнения данной лабораторной работы мои навыки работы с операционной системой Linux и различными её системами (компонентами) несомненно улучшились по сравнению с моими навыками до работы, также у меня открылось всевидящее око созерцания в середине лба: я стал остро чувствовать перемены окружающего мира не только в его физической составляющей, но и духовной - день ото дня я становлюсь всё ближе к своей просветлённой форме брахмана. А ещё я внёс свою скромную лепту в развитие компьютерных технологий, написав эту, несомненно уникальную и абсолютно недооценённую работу. Кроме того прошу Вас (дорогой читатель) обратить внимание на изящный и не обременяющий слог моего большого и богатого вывода, я определённо достоин премии по литературе за эту работу. Ах да, теперь я дружу со списочками, стеками и в целом устал.

## Список литературы

... ..