

Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Гаврилейко Алина Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки.....	13
4.3	Задание для самостоятельной работы.....	17
5	Выводы	20
6	Список литературы	21

Список иллюстраций

4.1	Создание каталога	8
4.2	Копирование программы из листинга	9
4.3	Запуск программы	9
4.4	Изменение программы.....	10
4.5	Запуск измененной программы.....	11
4.6	Добавление push и pop в цикл программы.....	12
4.7	Запуск измененной программы.....	13
4.8	Копирование программы из листинга.....	13
4.9	Запуск второй программы.....	14
4.10	Копирование программы из третьего листинга.....	15
4.11	Запуск третьей программы.....	15
4.12	Изменение третьей программы.....	16
4.13	Запуск измененной третьей программы.....	16
4.14	Написание программы для самостоятельной работы.....	17
4.15	Запуск программы для самостоятельной работы.....	19

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной ра- боты

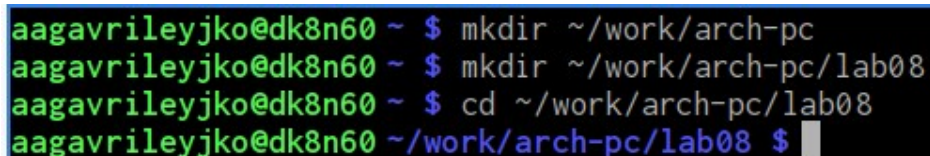
3 Теоретическое введение

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.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

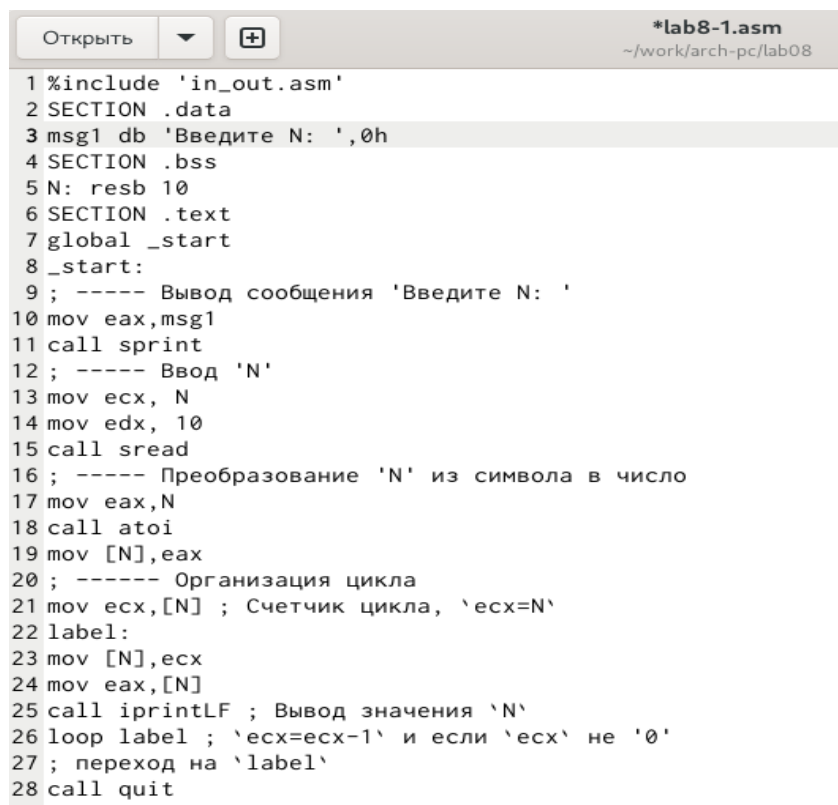
Создаю каталог для программ лабораторной работы №8 (рис. 4.1).



```
aagavrileyjko@dk8n60 ~ $ mkdir ~/work/arch-pc
aagavrileyjko@dk8n60 ~ $ mkdir ~/work/arch-pc/lab08
aagavrileyjko@dk8n60 ~ $ cd ~/work/arch-pc/lab08
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 4.1: Создание каталога

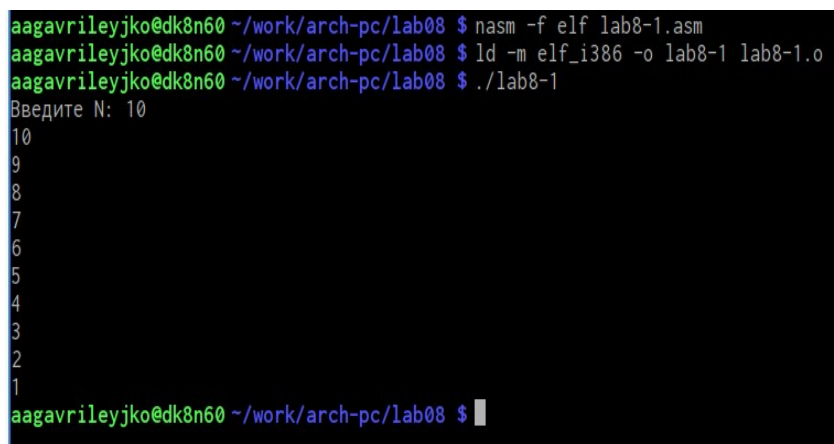
Копирую в созданный файл программу из листинга. (рис. 4.2).



```
Открыть  + *lab8-1.asm
~/work/arch-pc/lab08
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: Копирование программы из листинга

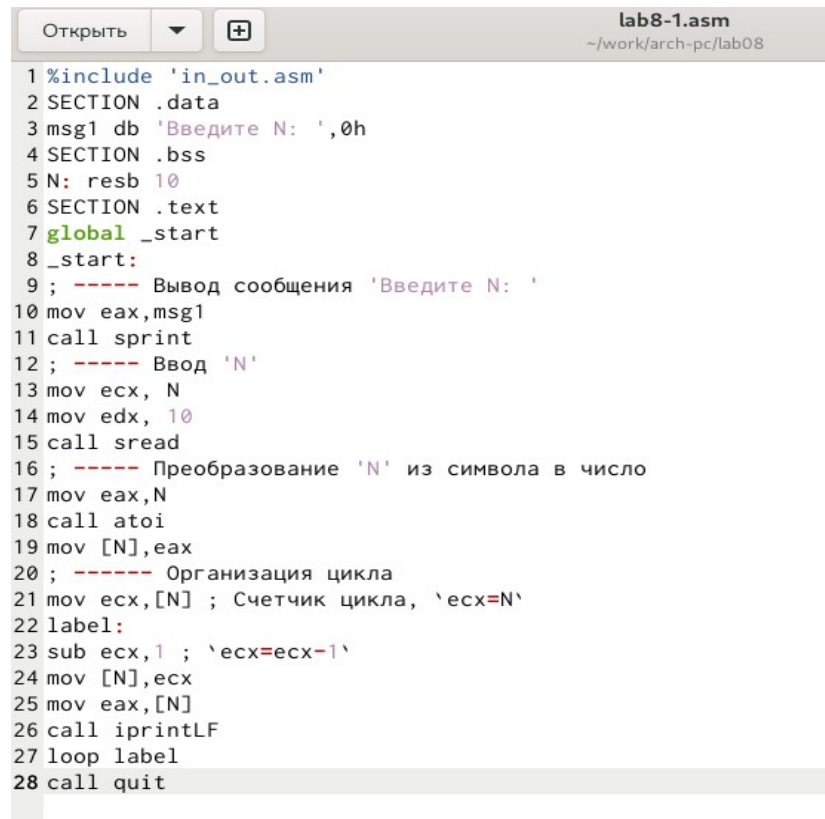
Запускаю программу, она показывает работу циклов в NASM (рис. 4.3).



```
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 4.3: Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра `ecx` (рис. 4.4).



```
lab8-1.asm
~/work/arch-pc/lab08

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
28 call quit
```

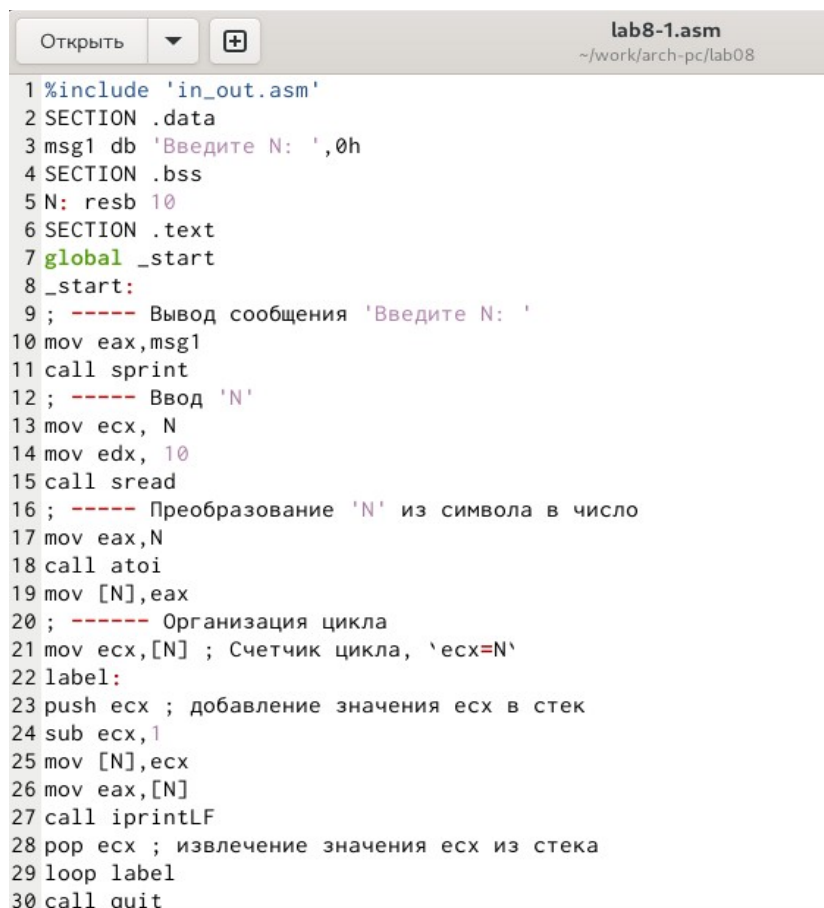
Рис. 4.4: Изменение программы

Из-за того, что теперь регистр `ecx` на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. 4.5).

```
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
7
5
3
1
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 4.5: Запуск измененной программы

Добавляю команды push и pop в программу (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 push ecx ; добавление значения ecx в стек
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF
28 pop ecx ; извлечение значения ecx из стека
29 loop label
30 call quit
```

Рис. 4.6: Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. 4.7).

```

aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $

```

Рис. 4.7: Запуск измененной программы

4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (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

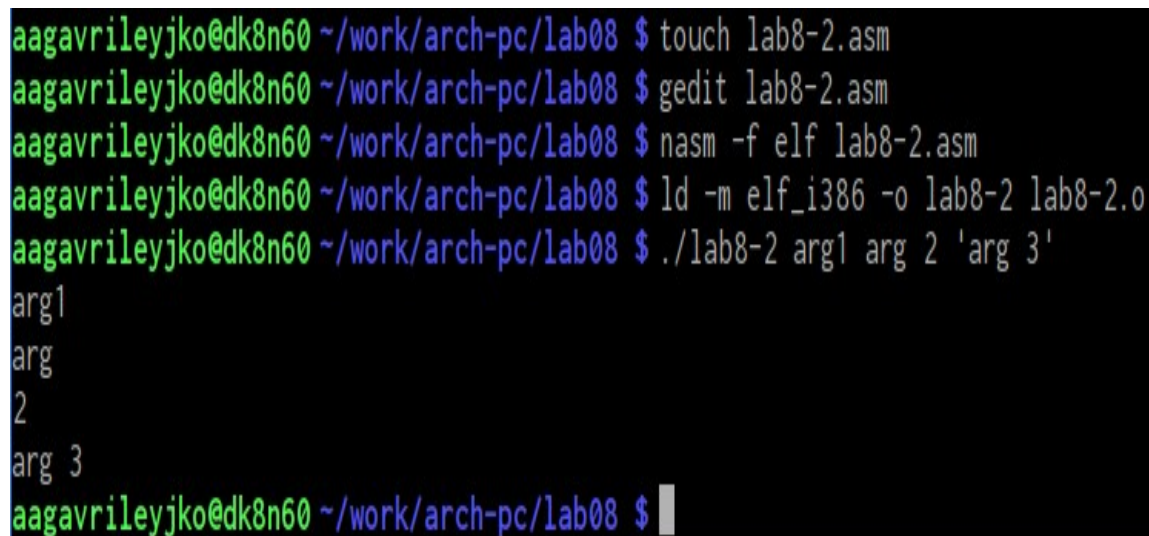
```

Рис. 4.8: Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было об-

ратоно то же количество аргументов, что и было введено (рис. 4.9).

Рис. 4.9: Запуск второй программы



```
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ touch lab8-2.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ gedit lab8-2.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $
```

Создаю новый файл для программы и копирую в него код из третьего листинга (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: Копирование программы из третьего листинга

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. 4.11).

```

aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $

```

Рис. 4.11: Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. 4.12).

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

Рис. 4.12: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. 4.13).

```
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-3 111 1 6
Результат: 666
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 4.13: Запуск измененной третьей программы

4.3 Задание для самостоятельной работы

Пишу программу, которая будет находить сумма значений для функции $f(x) = 10x - 4$, которая совпадает с моим десятым вариантом (рис. 4.14).

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg_func db "Функция: f(x) = 5(2 + x)", 0
5 msg_result db "Результат: ", 0
6
7 SECTION .text
8 GLOBAL _start
9
10 _start:
11     mov eax, msg_func      ; Выводим сообщение с описанием функции
12     call sprintf
13
14     pop ecx                ; Получаем количество аргументов
15     pop edx                ; Пропускаем имя программы
16     sub ecx, 1             ; Уменьшаем на 1, так как имя программы не считается
17     mov esi, 0             ; Переменная для суммы результата
18
19 next:
20     cmp ecx, 0h            ; Если все аргументы обработаны, переходим к завершению
21     jz _end
22
23     pop eax                ; Считываем следующий аргумент
24     call atoi              ; Преобразуем строку в число
25
26     add eax, 2             ; Вычисляем (2 + x)
27     mov ebx, 5             ; Умножаем на 5
28     mul ebx
29
30     add esi, eax           ; Добавляем результат к общей сумме
31
32     loop next
33
34 _end:
35 mov eax, msg_result
36 call sprintf
37 mov eax, esi
38 call iprintLF
39 call quit
```

Рис. 4.14: Написание программы для самостоятельной работы

Код программы:

Код программы:

```NASM

%include 'in\_out.asm'

SECTION .data

msg\_func db "Функция: f(x) = 5(2 + x)", 0

msg\_result db "Результат: ", 0

SECTION .text

GLOBAL \_start

\_start:

mov eax, msg\_func ; Выводим сообщение с описанием функции  
call sprintf

pop ecx ; Получаем количество аргументов  
pop edx ; Пропускаем имя программы  
sub ecx, 1 ; Уменьшаем на 1, так как имя программы не  
считается  
mov esi, 0 ; Переменная для суммы результата

next:

cmp ecx, 0h ; Если все аргументы обработаны, переходим

```

к завершению
 jz _end

 pop eax ; Считываем следующий аргумент
 call atoi ; Преобразуем строку в число

 add eax, 2 ; Вычисляем (2 + x)
 mov ebx, 5 ; Умножаем на 5
 mul ebx

 add esi, eax ; Добавляем результат к общей сумме

 loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

```

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. 4.15).

```
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3
Функция:  $f(x) = 5(2 + x)$ 
Результат: 60
aagavrileyjko@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 4.15: Запуск программы для самостоятельной работы

5 Выводы

В результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием циклов а также научилась обрабатывать аргументы командной строки.

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №8
3. Программирование на языке ассемблера NASM Столяров А. В.