Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Калашникова Ольга Сергеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Реализация циклов в NASM

Создаём каталог для программам лабораторной работы № 8 (при помощи команды mkdir ~/work/arch-pc/lab08), переходим в него (при помощи команды cd ~/work/arch-pc/lab08) и создаём файл lab8-1.asm (при помощи команды touch lab8-1.asm) (рис. 2.1),(рис. 2.2)

```
oskalashnikova@dk6n62:~$ mkdir ~/work/arch-pc/lab08
oskalashnikova@dk6n62:~$ cd ~/work/arch-pc/lab08
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ touch lab8-1.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.1: Создание папки, её открытие и создание файла

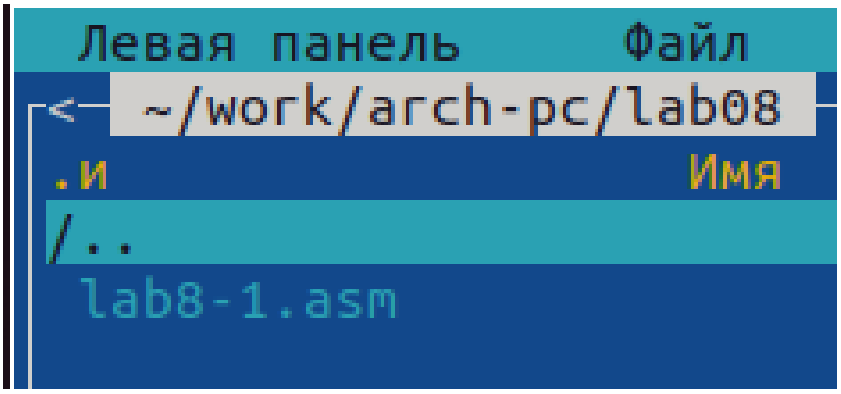
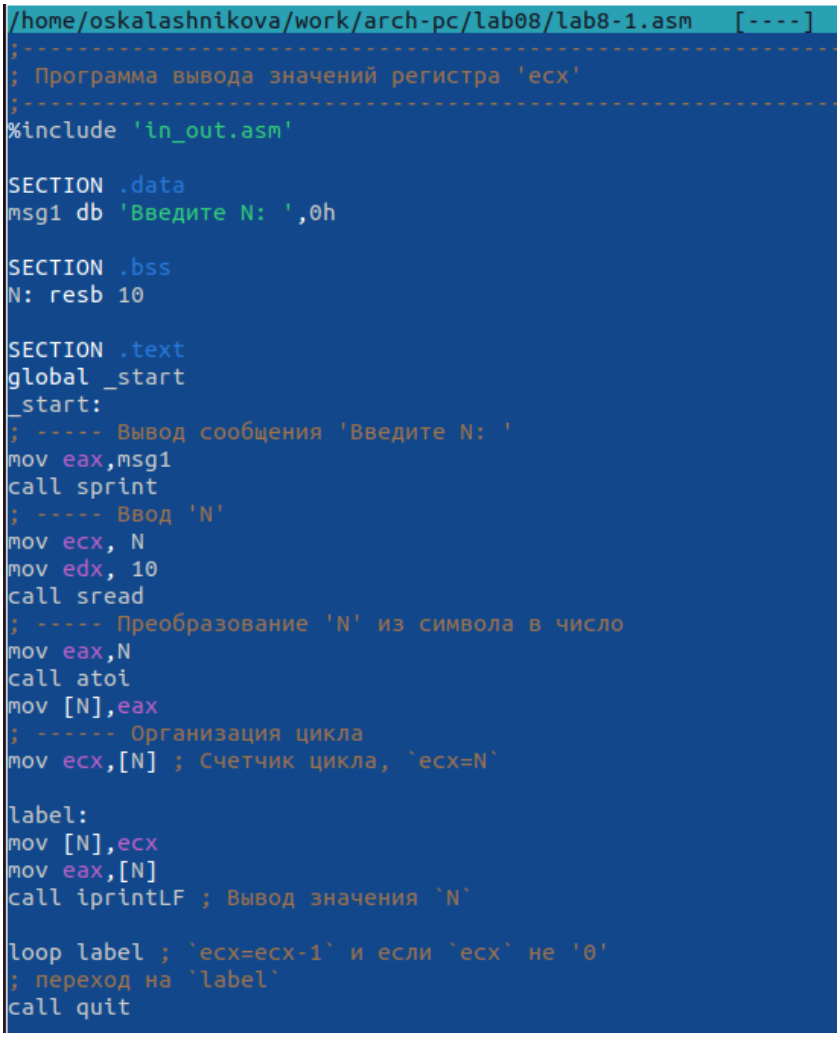


Рис. 2.2: Проверка наличия папки

Введём в файл lab8-1.asm текст программы из листинга 8.1., для корректной

работы нужно переместить файл “in_out.asm” в тот же каталог, где лежит и файл с текстом программы.(рис. 2.3),(рис. 2.4)



```
/home/oskalashnikova/work/arch-pc/lab08/lab8-1.asm   [----]
;-------------------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------------------
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`

label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`

loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2.3: Текст программы

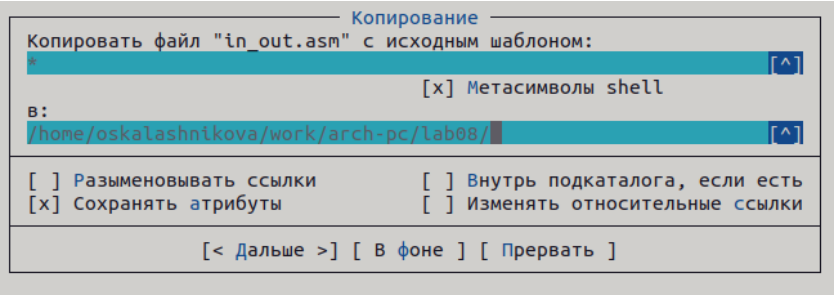


Рис. 2.4: Перемещение файла “in_out.asm”

Создаем исполняемый файл и запускаем его (компиляция: nasm -f elf lab8-1.asm , ld -m elf_i386 -o lab8-1 lab8-1.o , запуск: ./lab8-1) (рис. 2.5)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.5: Создание и запуск исполняемого файла

Меняем текст программы (Используем регистр ecx в теле цилка loop) (рис. 2.6)

```
/home/oskalashnikova/work/arch-pc/lab08/lab8-1-2.asm
;-------------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------------
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`

label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF

loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2.6: Изменённый текст программы

Создаём исполняемый файл и запускаем его (компиляция: nasm -f elf lab8-1-2.asm , ld -m elf_i386 -o lab8-1-2 lab8-1-2.o , запуск: ./lab8-1-2) (рис. 2.7), (рис. 2.8), (рис. 2.9)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ nasm -f elf lab8-1-2.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1-2 lab8-1-2.o
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.7: Создание исполняемого файла

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ./lab8-1-2
Введите N: 3
```

Рис. 2.8: Запуск исполняемого файла 1

```
4294912712
4294912710
4294912708
4294912706
4294912704
4294912702
4294912700
4294912698
4294912696
4294912694
4294912692
4294912690
4294912688
4294912686
4294912684
42949126^Z
[3]+  Остановлен    ./lab8-1-2
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.9: Запуск исполняемого файла 2

Использование регистра ecx в теле цилка loop может привести к некорректной работе программы, что мы и видим в данном примере.

Вопросы:

- Какие значения принимает регистр ecx в цикле? - Регистр ecx принимает некрректные значения.
- Соответствует ли число проходов цикла значению N введенному с клавиатуры? Число проходов цикла не соответствуют значению N введенному с клавиатуры.

Меняем текст программы (Используем стек. Добавим команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop) (рис. 2.10)

```
/home/oskalashnikova/work/arch-pc/lab08/lab8-1-3.asm
;-------------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------------
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`

label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека

loop label
call quit
```

Рис. 2.10: Изменённый текст программы

Создаём исполняемый файл и запускаем его (компиляция: nasm -f elf lab8-1-3.asm , ld -m elf_i386 -o lab8-1-3 lab8-1-3.o , запуск: ./lab8-1-3) (рис. 2.11)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ nasm -f elf lab8-1-3.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1-3 lab8-1-3.o
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ./lab8-1-3
Введите N: 5
4
3
2
1
0
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.11: Создание и запуск исполняемого файла

Соответствует ли в данном случае число проходов цикла значению N введенному с клавиатуры? - Число проходов цикла соответствуют значению N введенному с клавиатуры, но вывод идёт от 4 до 0 из-за того что ecx-1 теперь работает.

Создаём файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (при помощи команды touch lab8-2.asm) (рис. 2.12)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ touch lab8-2.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.12: Создание файла

Введём в файл lab8-2.asm текст программы из листинга 8.2. (рис. 2.13)

```
/home/oskalashnikova/work/arch-pc/lab08/lab8-2.asm
;-------------------------------------------------
; Обработка аргументов командной строки
;-------------------------------------------------
%include 'in_out.asm'

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)

next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем аргумент из стека
call sprintLF ; вызываем функцию печати

loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)

_end:
call quit
```

Рис. 2.13: Текст программы

Создаём исполняемый файл и запускаем его, указав аргументы: аргумент1 аргумент 2 ‘аргумент 3’ (компиляция: nasm -f elf lab8-1-3.asm , ld -m elf_i386 -o lab8-1-3 lab8-1-3.o , запуск: ./lab8-1-3) (рис. 2.14)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.14: Создание и запуск исполняемого файла

Сколько аргументов было обработано программой? - 4 аргумента, так как аргумент1 и ‘аргумент 3’ считаются по одному элементу, а аргумент и 2 это два

разных (стоят через пробел без кавычек)

Создаём файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (при помощи команды touch lab8-3.asm) (рис. 2.15)

Рис. 2.15: Создание файла

Введём в файл lab8-3.asm текст программы из листинга 8.3. (Программа вычисления суммы аргументов командной строки) (рис. 2.16)

```
/home/oskalashnikova/work/arch-pc/lab08/lab8-3.asm
%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
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=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
```

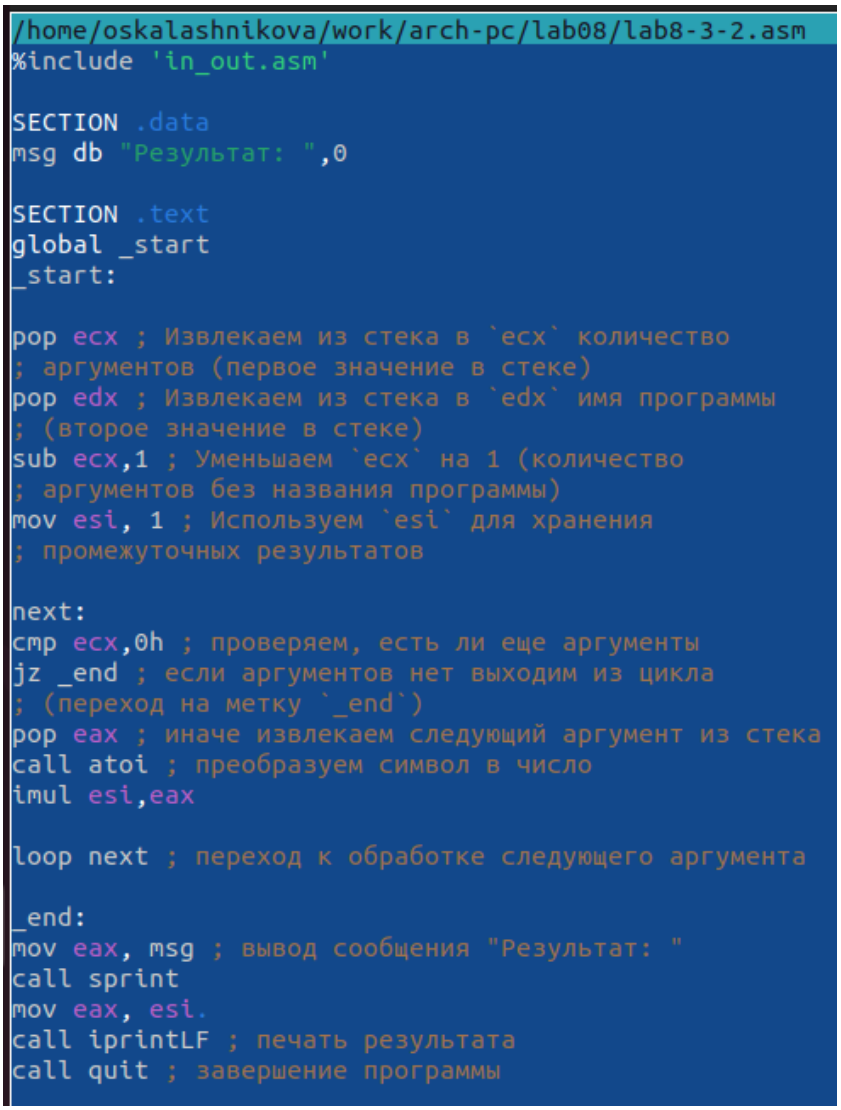
Рис. 2.16: Текст программы

Создаём исполняемый файл и запускаем его, указав аргументы: 12 13 7 10 5 (компиляция: nasm -f elf lab8-3.asm , ld -m elf_i386 -o lab8-3 lab8-3.o , запуск: ./lab8-3) (рис. 2.17)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.17: Создание и запуск исполняемого файла

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 2.18)



```
/home/oskalashnikova/work/arch-pc/lab08/lab8-3-2.asm
%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, 1 ; Используем `esi` для хранения
; промежуточных результатов

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 esi,eax

loop next ; переход к обработке следующего аргумента

_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi.
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис. 2.18: Изменённый текст программы

Создаём исполняемый файл и запускаем его, указав аргументы: 12 13 7 10 5 (компиляция: nasm -f elf lab8-3-2.asm , ld -m elf_i386 -o lab8-3-2 lab8-3-2.o , запуск: ./lab8-3-2) (рис. 2.19)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ nasm -f elf lab8-3-2.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3-2 lab8-3-2.o
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ./lab8-3-2 1 2 3 4
Результат: 24
oskalashnikova@dk6n62:~/work/arch-pc/lab08$
```

Рис. 2.19: Создание и запуск исполняемого файла

```
%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, 1 ; Используем `esi` для хранения
; промежуточных результатов

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 esi,eax

loop next ; переход к обработке следующего аргумента

_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi
call iprintLF ; печать результата
call quit ; завершение программы
```

##Задание для самостоятельной работы

Задание 1: Напишем программу, которая находит сумму значений функци f(x) для x = x1, x2, … , xn (т.е. программа должна выводить значение f(x1) + f(x2) + … + f(xn)) Значения xi передаются как аргументы. Вид функции f(x) возьмём из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. (рис. 2.20), (рис. 2.21)

7 $3(x+2)$

Рис. 2.20: Вариант 7

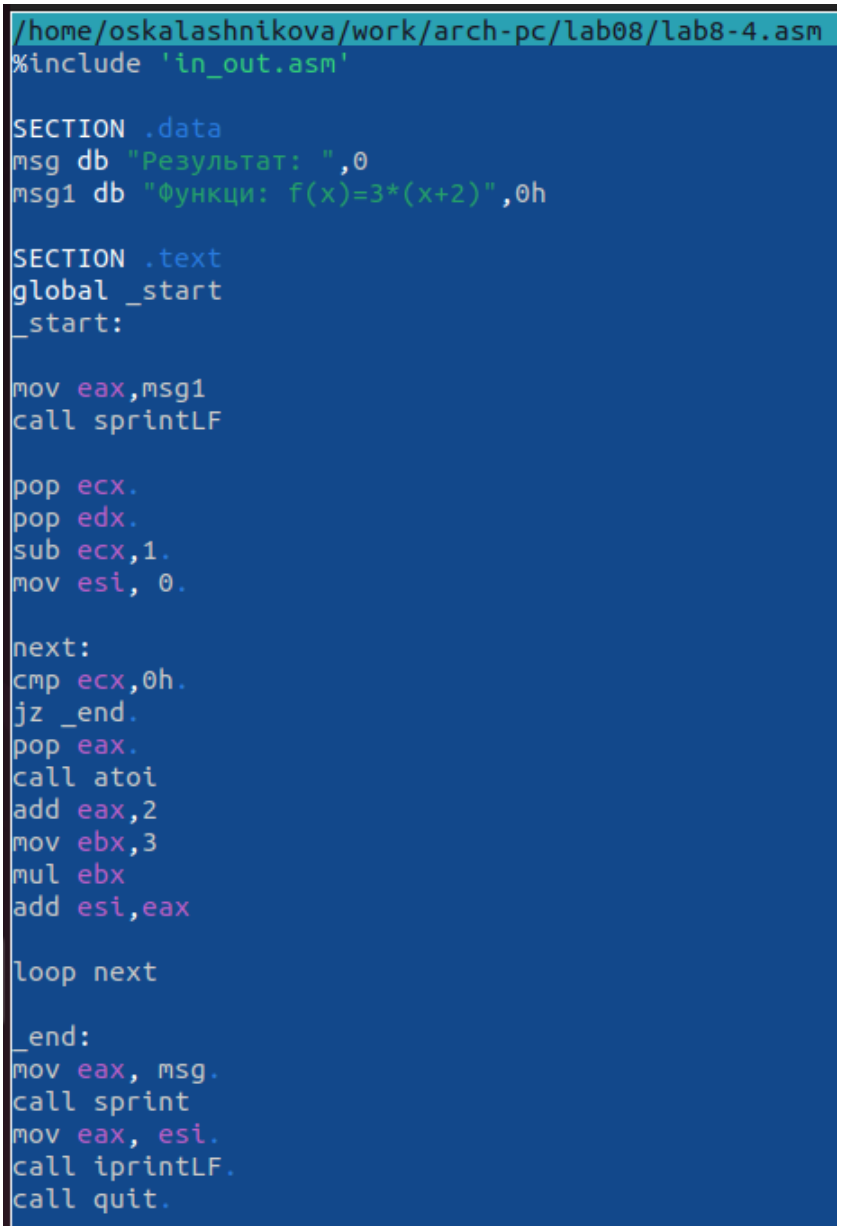


Рис. 2.21: Текст программы

Текст программы:

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0
```

```
msg1 db "Функци: f(x)=3*(x+2)",0h

SECTION .text
global _start
_start:

mov eax,msg1
call sprintLF

pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
add eax,2
mov ebx,3
mul ebx
add esi,eax

loop next

_end:
mov eax, msg
call sprint
```

```
mov eax, esi
call iprintLF
call quit
```

Создаём исполняемый файл и проверяем его работу на нескольких наборах (я использовала 1,2,3,4 и 10,11,12) (компиляция: nasm -f elf lab8-4.asm , ld -m elf_i386 -o lab8-4 lab8-4.o , запуск: ./lab8-4) (рис. 2.22)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функци: f(x)=3*(x+2)
Результат: 54
oskalashnikova@dk6n62:~/work/arch-pc/lab08$ ./lab8-4 10 11 12
Функци: f(x)=3*(x+2)
Результат: 117
```

Рис. 2.22: Создание и запуск исполняемого файла

Посчитав значения сами мы понимаем, что программа работает верно.

3 Выводы

В ходе выполнения лабораторной работы мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.