

Отчёт по лабораторной работе 8

Архитектура компьютеров и операционные системы

Алексей Белов НПИбд-01-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Самостоятельное задание	17
3	Выводы	20

Список иллюстраций

2.1	Программа в файле lab8-1.asm	7
2.2	Запуск программы lab8-1.asm	8
2.3	Программа в файле lab8-1.asm	9
2.4	Запуск программы lab8-1.asm	10
2.5	Программа в файле lab8-1.asm	11
2.6	Запуск программы lab8-1.asm	12
2.7	Программа в файле lab8-2.asm	13
2.8	Запуск программы lab8-2.asm	13
2.9	Программа в файле lab8-3.asm	14
2.10	Запуск программы lab8-3.asm	15
2.11	Программа в файле lab8-3.asm	16
2.12	Запуск программы lab8-3.asm	17
2.13	Программа в файле lab8-4.asm	18
2.14	Запуск программы lab8-4.asm	19

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Создал каталог для программ лабораторной работы № 8 и файл lab8-1.asm

При реализации циклов в NASM с использованием инструкции `loop` необходимо помнить о том, что эта инструкция использует регистр `ecx` в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра `ecx`.

Написал в файл lab8-1.asm текст программы из листинга 8.1. (рис. [2.1]) Создал исполняемый файл и проверил его работу. (рис. [2.2])

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab08
/home/ad~8-1.asm [---] 20 L:[ 1+26 27/ 29] *(626 / 637b)
#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. 2.1: Программа в файле lab8-1.asm

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

Рис. 2.2: Запуск программы lab8-1.asm

Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Изменил текст программы добавив изменение значение регистра `ecx` в цикле. (рис. [2.3])

Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N`. (рис. [2.4])


```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/lab08
/home/ad~8-1.asm  [----] 10 L:[ 1+18 19/ 30] *(372 / 586b) 0120
#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
; переход на `label`
call quit
```

Рис. 2.3: Программа в файле lab8-1.asm

```

adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ █

```

Рис. 2.4: Запуск программы lab8-1.asm

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внес изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`. (рис. [2.5])

Создал исполняемый файл и проверьте его работу. Программа выводит числа от $N-1$ до 0, число проходов цикла соответствует N . (рис. [2.6])

```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/lab08
/home/ad~8-1.asm [----] 9 L:[ 1+17 18/ 31] *(361 / 675b) 001
%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. 2.5: Программа в файле lab8-1.asm

```

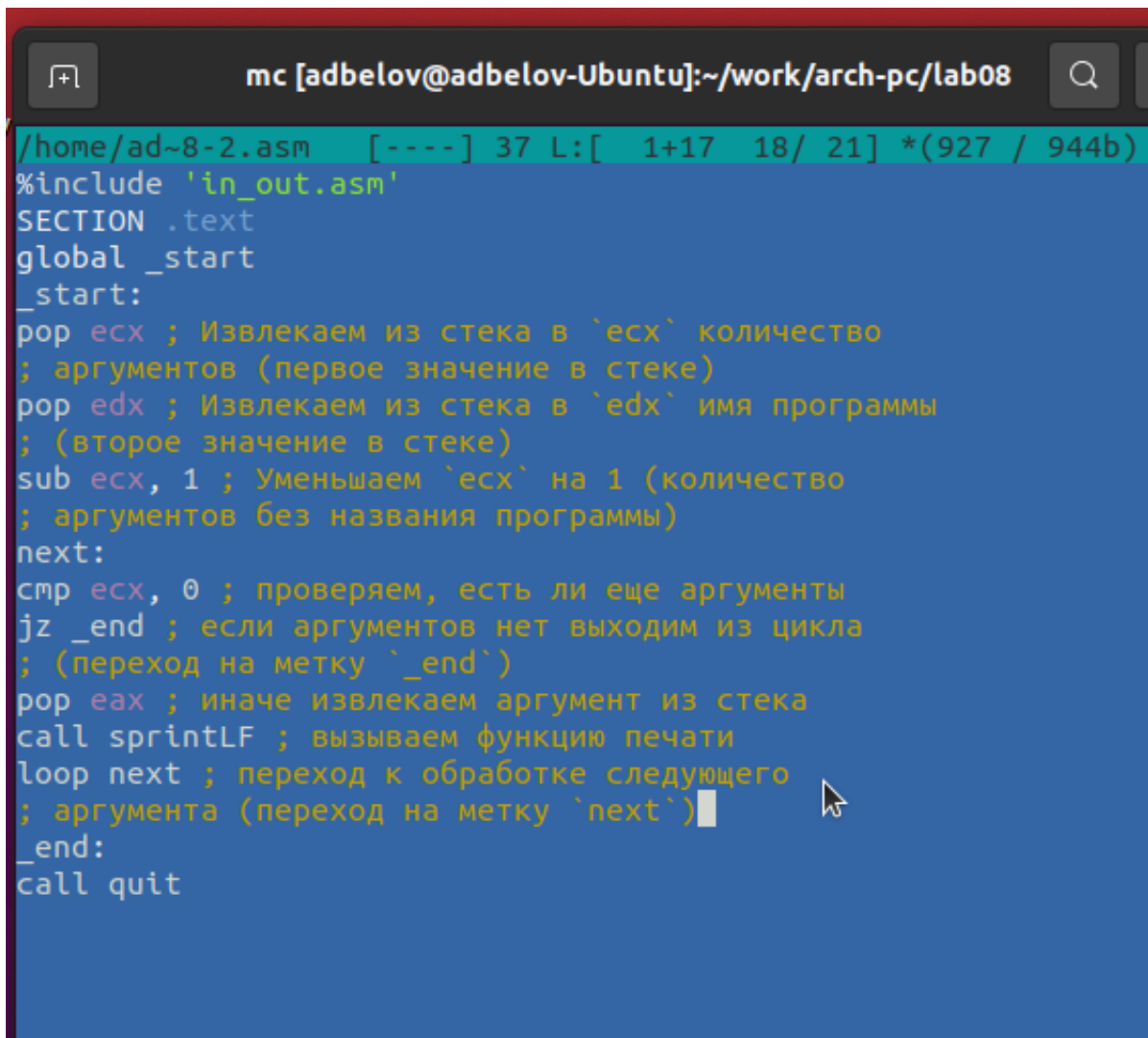
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$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

```

Рис. 2.6: Запуск программы lab8-1.asm

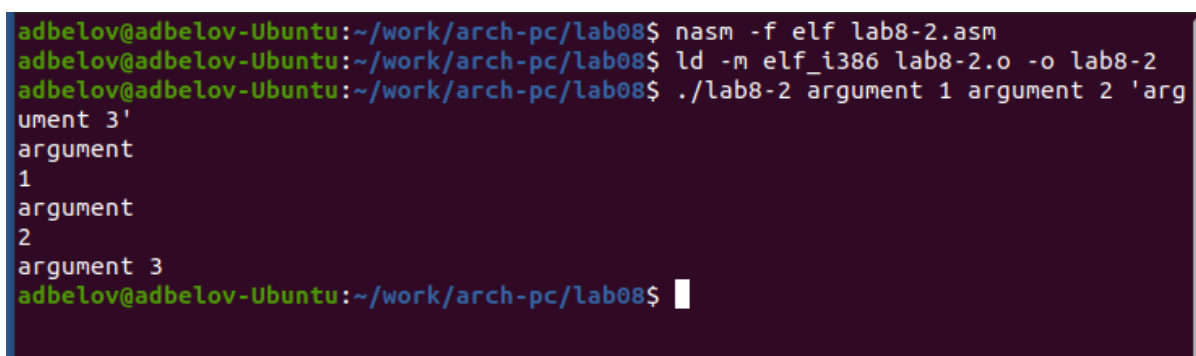
Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2. (рис. [2.7])

Создал исполняемый файл и запустил его, указав аргументы. Программа обработала 5 аргументов. Аргументами считаются слова/числа, разделенные пробелом. (рис. [2.8])



```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/lab08
/home/ad~8-2.asm [----] 37 L:[ 1+17 18/ 21] *(927 / 944b)
#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
```

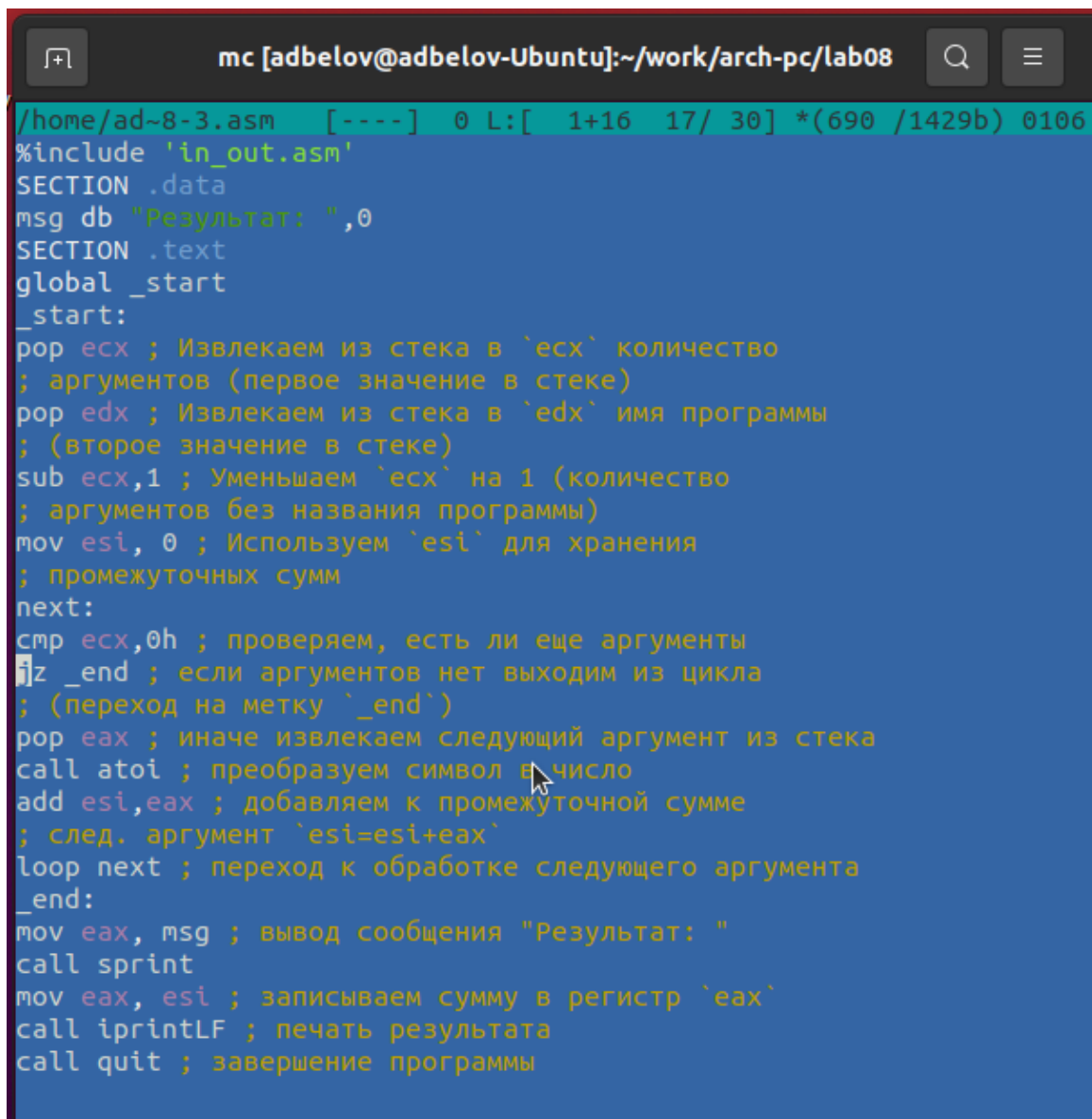
Рис. 2.7: Программа в файле lab8-2.asm



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

Рис. 2.8: Запуск программы lab8-2.asm

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. (рис. [2.9]) (рис. [2.10])



```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/lab08
/home/ad~8-3.asm [----] 0 L:[ 1+16 17/ 30] *(690 /1429b) 0106
%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 iprintfLF ; печать результата
call quit ; завершение программы
```

Рис. 2.9: Программа в файле lab8-3.asm

```
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 2 2 2
Результат: 6
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 6 8 7 1 3 4 6
Результат: 35
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$
```

Рис. 2.10: Запуск программы lab8-3.asm

Изменил текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. [2.11]) (рис. [2.12])

```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/lab08
/home/ad~8-3.asm [---] 7 L:[ 1+22 23/ 33] *(995 /1461b) 0010 0x00A[*]
%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 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
mov ebx,eax
mov eax,esi
mul ebx
mov 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. 2.11: Программа в файле lab8-3.asm


```

adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 2 2 2
Результат: 6
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 6 8 7 1 3 4 6
Результат: 35
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 2 2 2
Результат: 8
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 6 8 7 1 3 4 6
Результат: 24192
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ █

```

Рис. 2.12: Запуск программы lab8-3.asm

2.1 Самостоятельное задание

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 № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x . (рис. [2.13]) (рис. [2.14])

для варианта 8

$$f(x) = 7 + 2x$$

```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/lab08
/home/ad~8-4.asm [----] 0 L:[ 1+18 19/ 34] *(220 /
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 7 + 2x',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx,2
mul ebx
add eax,7
add esi,eax

loop next

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

Рис. 2.13: Программа в файле lab8-4.asm

Для проверки я запустил сначала с одним аргументом. Так, при подстановке $f(1) = 9, f(3) = 13$ Затем подал несколько аргументов и получил сумму значений функции.

```
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-4
f(x)= 7 + 2x
Результат: 0
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-4 1
f(x)= 7 + 2x
Результат: 9
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$ ./lab8-4 1 3 4 6 7 8 9 6
f(x)= 7 + 2x
Результат: 144
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab08$
```

Рис. 2.14: Запуск программы lab8-4.asm

3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере `naasm`.