

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Будник Александра Олеговна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация циклов в NASM	6
2.2	Самостоятельное задание	16
3	Выводы	19

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab8-1.asm	7
2.3	Запуск программы lab8-1.asm	8
2.4	Программа lab8-1.asm	9
2.5	Запуск программы lab8-1.asm	9
2.6	Программа lab8-1.asm	10
2.7	Запуск программы lab8-1.asm	11
2.8	Программа lab8-2.asm	12
2.9	Запуск программы lab8-2.asm	13
2.10	Программа lab8-3.asm	14
2.11	Запуск программы lab8-3.asm	14
2.12	Программа lab8-3.asm	15
2.13	Запуск программы lab8-3.asm	15
2.14	Программа lab8-4.asm	17
2.15	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

2.1 Реализация циклов в NASM

Создала каталог для программ лабораторной работы № 8 и файл lab8-1.asm (рис. [2.1])

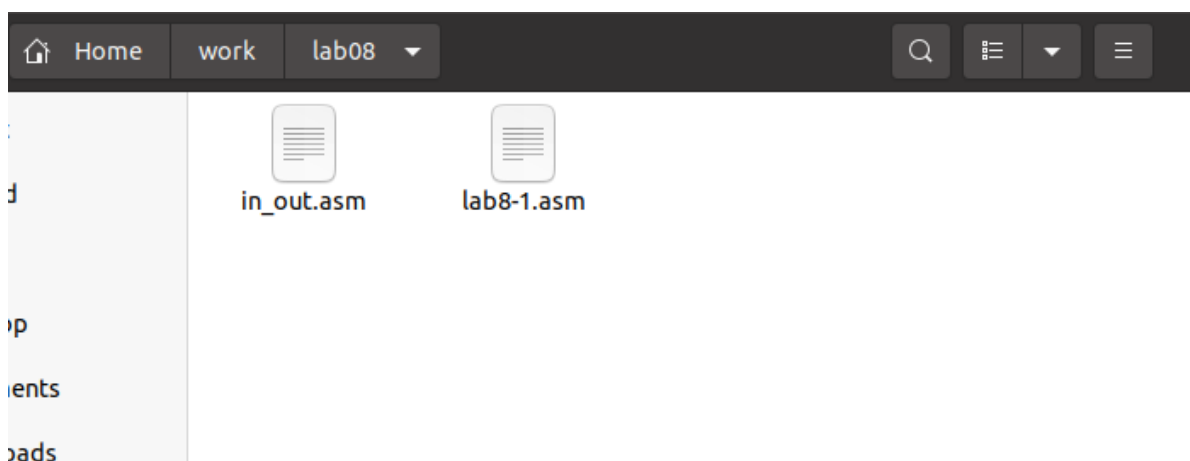


Рис. 2.1: Создан каталог

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`.

Написала в файл `lab8-1.asm` текст программы из листинга 8.1. (рис. [2.2])
Создала исполняемый файл и проверила его работу. (рис. [2.3])

```
aobudnik@VirtualBox: ~/work/lab08
/home/aob~ab8-1.asm [----] 0 L:[ 1+28 29/ 29] *(637 /
#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. 2.2: Программа lab8-1.asm

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

Рис. 2.3: Запуск программы lab8-1.asm

Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Изменяю текст программы, добавив изменение значение регистра `ecx` в цикле. (рис. [2.4]) Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N`. (рис. [2.5])


```
mc [aobudnik@VirtualBox]:~/work/lab08
/home/aob~ab8-1.asm [----] 20 L:[ 1+27 28/ 30] *(575 /
#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. 2.4: Программа lab8-1.asm

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

Рис. 2.5: 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. [2.6]) Создаю исполняемый файл и проверяю его работу. (рис. [2.7]) Программа выводит числа от $N-1$ до 0, число проходов цикла соответствует N .

```

mc [aobudnik@VirtualBox]:~/work/lab08
/home/aob~ab8-1.asm [----] 13 L:[ 1+26 27/ 31] *(585 /
#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

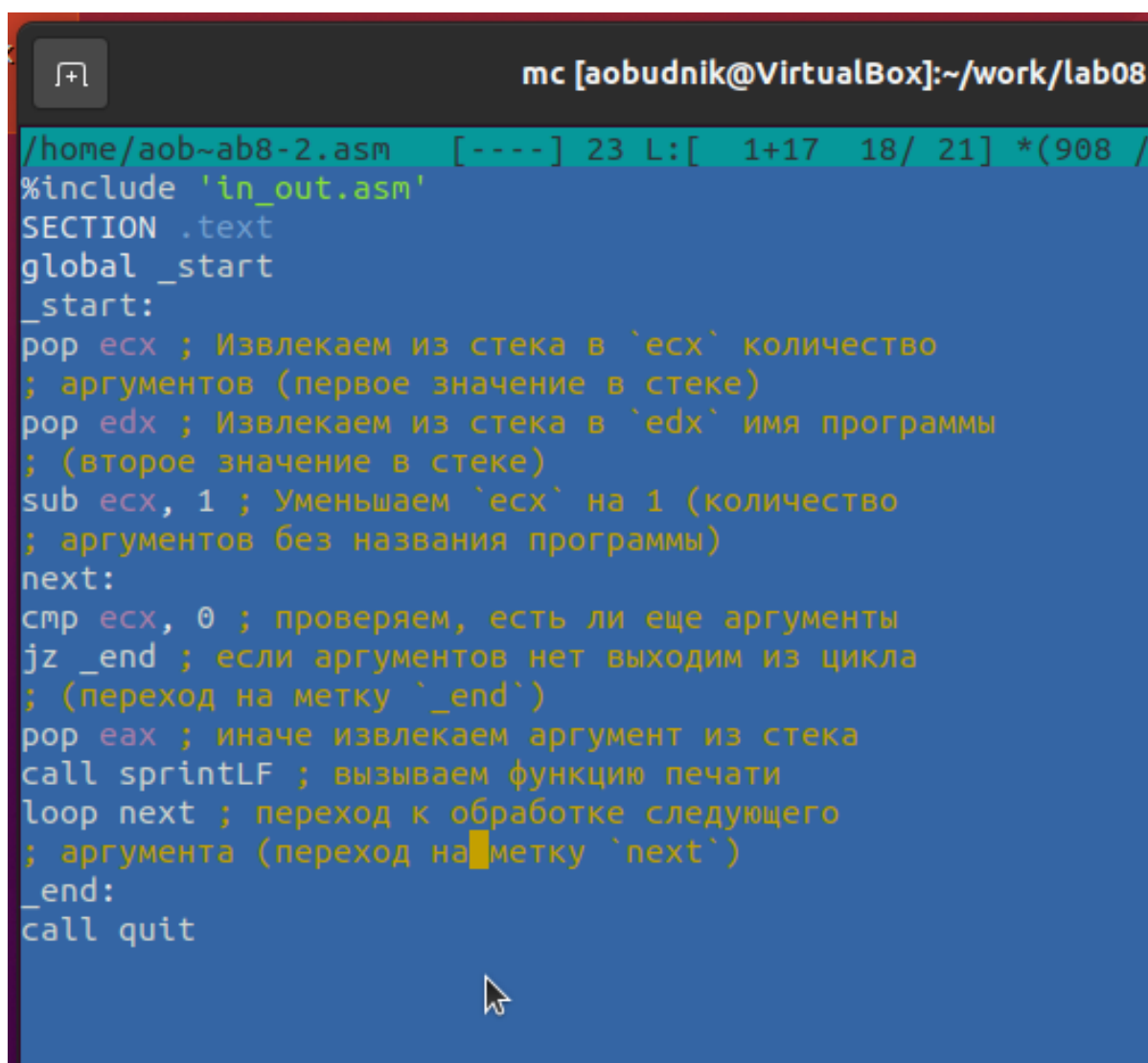
```

Рис. 2.6: Программа lab8-1.asm

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

Рис. 2.7: Запуск программы lab8-1.asm

Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и написала в него текст программы из листинга 8.2. (рис. [2.8]) Компилирую исполняемый файл и запускаю его, указав аргументы. Программа обработала 5 аргументов. Аргументами считаются слова/числа, разделенные пробелом. (рис. [2.9])



```
mc [aobudnik@VirtualBox]:~/work/lab08
/home/aob~ab8-2.asm  [ - - - - ] 23 L: [ 1+17 18/ 21 ] *(908 /
#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. 2.8: Программа lab8-2.asm

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

Рис. 2.9: Запуск программы lab8-2.asm

Рассмотрим еще один пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. (рис. [2.10]) (рис. [2.11])

```
mc [aobudnik@VirtualBox]:~/work/lab08
/home/aob~ab8-3.asm [----] 32 L:[ 1+28 29/ 30] *(1428/1429b) 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, 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
```

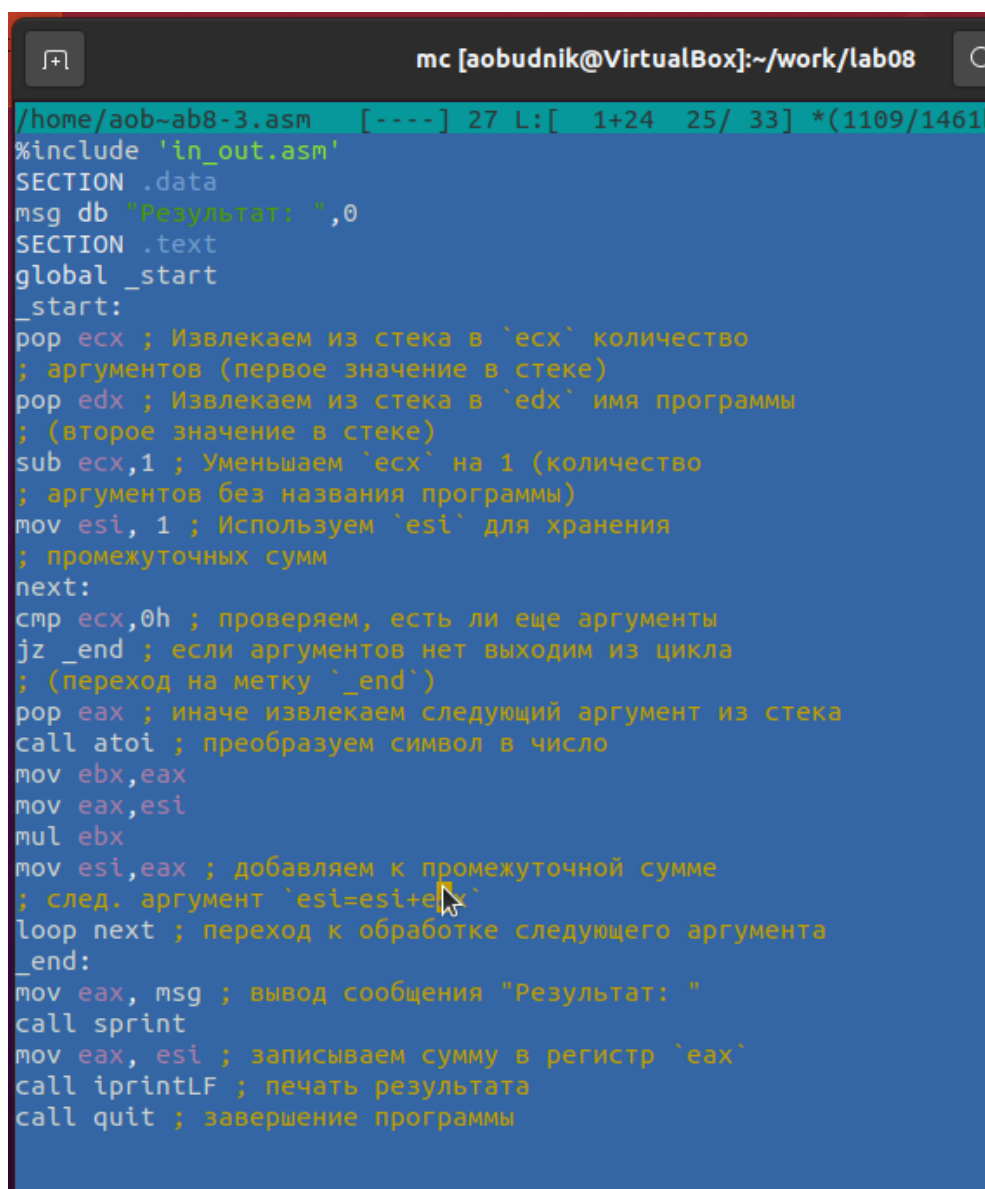
Рис. 2.10: Программа lab8-3.asm

```
aobudnik@VirtualBox:~/work/lab08$
aobudnik@VirtualBox:~/work/lab08$ nasm -f elf lab8-3.asm
aobudnik@VirtualBox:~/work/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
aobudnik@VirtualBox:~/work/lab08$ ./lab8-3 3 4 5
Результат: 12
aobudnik@VirtualBox:~/work/lab08$
aobudnik@VirtualBox:~/work/lab08$
```

Рис. 2.11: Запуск программы lab8-3.asm

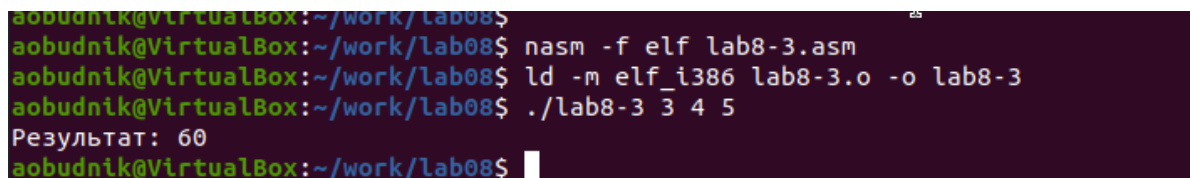
Изменяю текст программы из листинга 8.3 для вычисления произведения

аргументов командной строки. (рис. [2.12]) (рис. [2.13])



```
mc [aobudnik@VirtualBox]:~/work/lab08
/home/aob-ab8-3.asm [----] 27 L:[ 1+24 25/ 33] *(1109/1461
#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. 2.12: Программа lab8-3.asm



```
aobudnik@VirtualBox:~/work/lab08$
aobudnik@VirtualBox:~/work/lab08$ nasm -f elf lab8-3.asm
aobudnik@VirtualBox:~/work/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
aobudnik@VirtualBox:~/work/lab08$ ./lab8-3 3 4 5
Результат: 60
aobudnik@VirtualBox:~/work/lab08$
```

Рис. 2.13: Запуск программы lab8-3.asm

2.2 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. [2.14]) (рис. [2.15])

для варианта 17

$$f(x) = 10(x - 1)$$


```
mc [aobudnik@VirtualBox]:~/work/lab08
/home/aob~ab8-4.asm [----] 10 L:[ 1+21 22/ 34] *(261 / 3
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 10(x - 1)',0

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

next:
cmp ecx,0h
jz _end.
pop eax
call atoi
sub eax,1
mov ebx,10
mul ebx
add esi,eax

loop next

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

Рис. 2.14: Программа lab8-4.asm

```
aobudnik@VirtualBox:~/work/lab08$  
aobudnik@VirtualBox:~/work/lab08$ nasm -f elf lab8-4.asm  
aobudnik@VirtualBox:~/work/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4  
aobudnik@VirtualBox:~/work/lab08$ ./lab8-4 1  
f(x)= 10(x - 1)  
Результат: 0  
aobudnik@VirtualBox:~/work/lab08$ ./lab8-4 1 3 4 5  
f(x)= 10(x - 1)  
Результат: 90  
aobudnik@VirtualBox:~/work/lab08$
```

Рис. 2.15: Запуск программы lab8-4.asm

Убедились, что программа считает правильно $f(0) = 7$, $f(1) = 14$.

3 Выводы

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