

Отчёт по лабораторной работе 8

Архитектура компьютера

Диденко Дмитрий Владимирович НПИбд-03-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация циклов в NASM	6
2.2	Обработка аргументов командной строки	12
2.3	Задание для самостоятельной работы	16
3	Выводы	19

Список иллюстраций

2.1	Код программы lab8-1.asm	7
2.2	Компиляция и запуск программы lab8-1.asm	8
2.3	Код программы lab8-1.asm	9
2.4	Компиляция и запуск программы lab8-1.asm	10
2.5	Код программы lab8-1.asm	11
2.6	Компиляция и запуск программы lab8-1.asm	12
2.7	Код программы lab8-2.asm	13
2.8	Компиляция и запуск программы lab8-2.asm	13
2.9	Код программы lab8-3.asm	14
2.10	Компиляция и запуск программы lab8-3.asm	14
2.11	Код программы lab8-3.asm	15
2.12	Компиляция и запуск программы lab8-3.asm	16
2.13	Код программы prog.asm	17
2.14	Компиляция и запуск программы prog.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

Был создан каталог для выполнения лабораторной работы № 8, а также создан файл с именем lab8-1.asm.

При использовании инструкции `loop` в NASM для реализации циклов, необходимо помнить о следующем: данная инструкция использует регистр `ecx` в качестве счетчика и на каждой итерации уменьшает его значение на единицу.

Для лучшего понимания этого процесса, рассмотрим пример программы, 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])

```
GNU nano 4.8 /home/dvddidenko/work/arch-pc/lab08/lab8-1.asm
%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: Код программы lab8-1.asm

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

Рис. 2.2: Компиляция и запуск программы lab8-1.asm

В данном примере демонстрируется, что использование регистра `ecx` в инструкции `loop` может привести к неправильной работе программы. В тексте программы были внесены изменения, которые включают изменение значения регистра `ecx` внутри цикла. (рис. [2.3])

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


```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab08/L
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab08/L
%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: Код программы lab8-1.asm

```
4294932392
4294932390
4294932388
4294932386
429493238^C
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$
```

Рис. 2.4: Компиляция и запуск программы lab8-1.asm

Для того чтобы использовать регистр `ecx` в цикле и обеспечить корректность работы программы, можно применить стек. Внесены изменения в текст программы, добавив команды `push` и `pop` для сохранения значения счетчика цикла `loop` в стеке. (рис. [2.5])

Был создан исполняемый файл и проверена его работа. Программа выводит числа от $N-1$ до 0, где количество проходов цикла соответствует значению N . (рис. [2.6])

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-p
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab
%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: Код программы lab8-1.asm

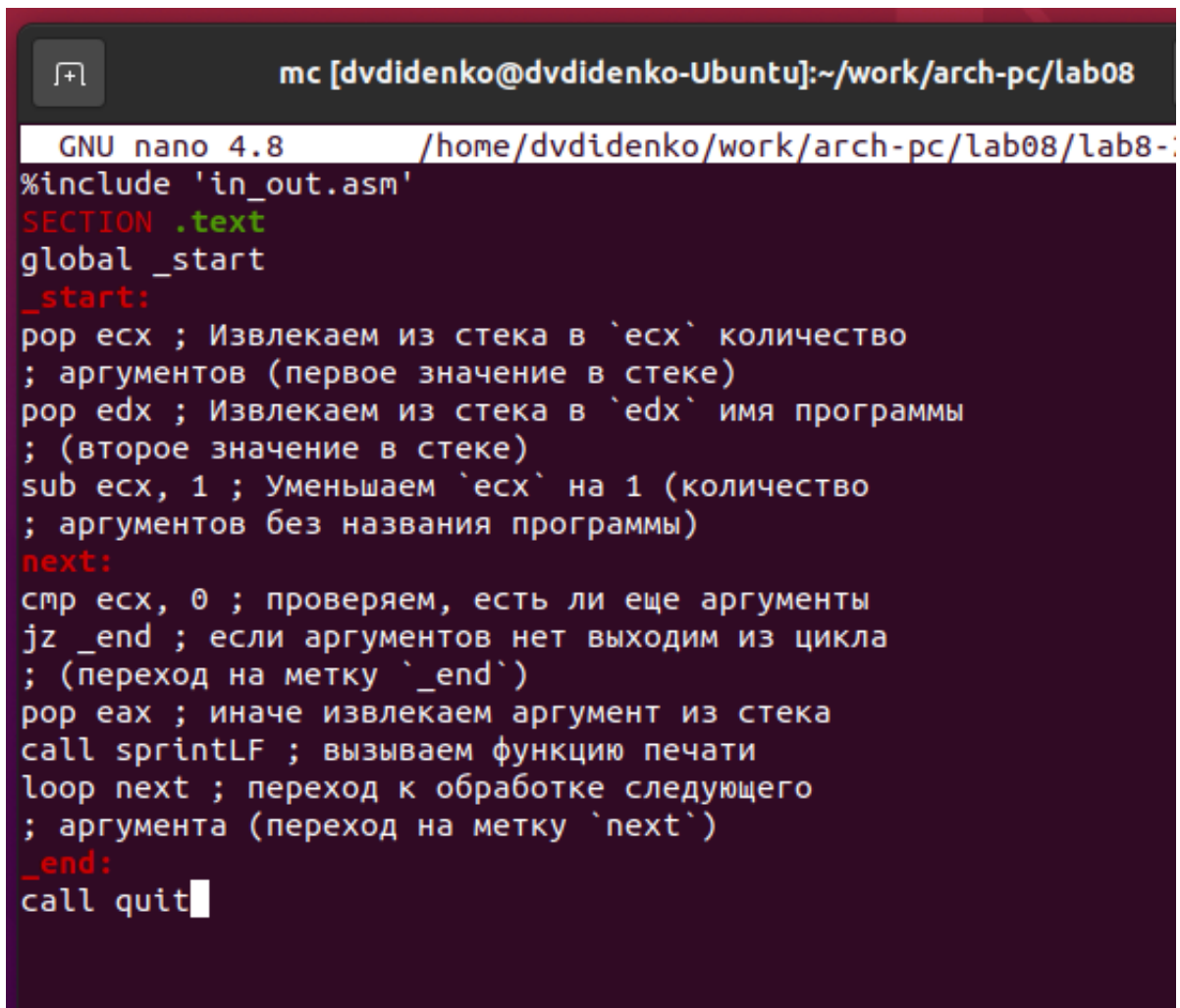
```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите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

Рис. 2.6: Компиляция и запуск программы lab8-1.asm

2.2 Обработка аргументов командной строки

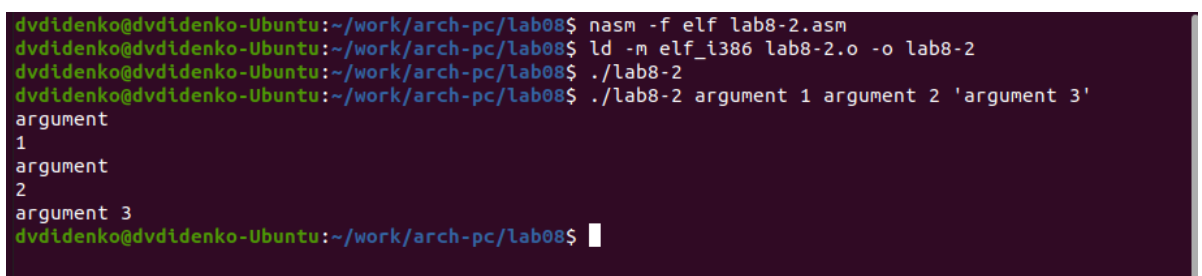
Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2. (рис. [2.7])

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



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab08
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab08/lab8-2.asm
#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 sprintf ; вызываем функцию печати
    loop next ; переход к обработке следующего
              ; аргумента (переход на метку `next`)
    _end:
    call quit
```

Рис. 2.7: Код программы lab8-2.asm

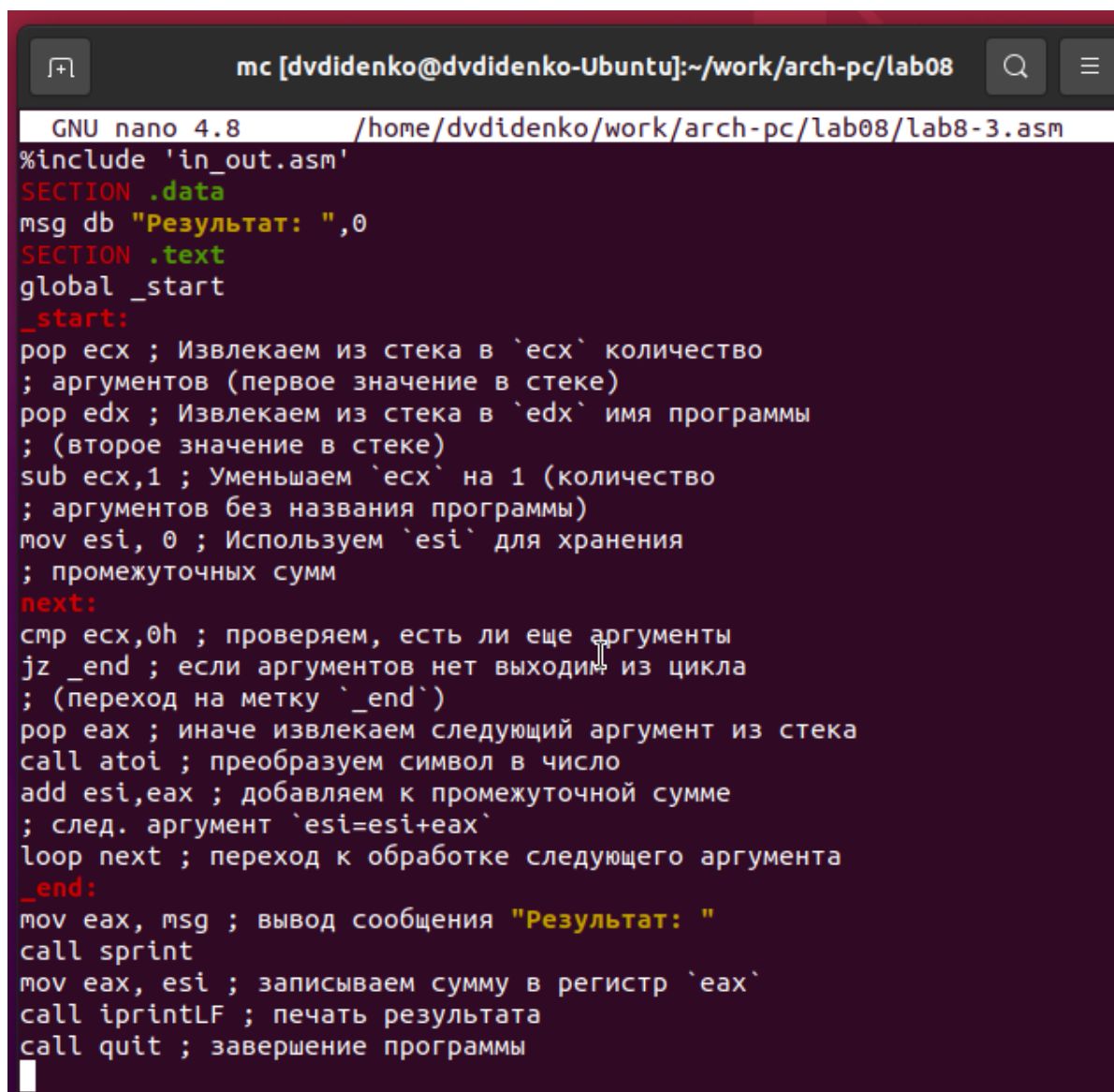


```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./lab8-2
argument 1 argument 2 'argument 3'
argument
1
argument
2
argument 3
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$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

Рис. 2.8: Компиляция и запуск программы lab8-2.asm

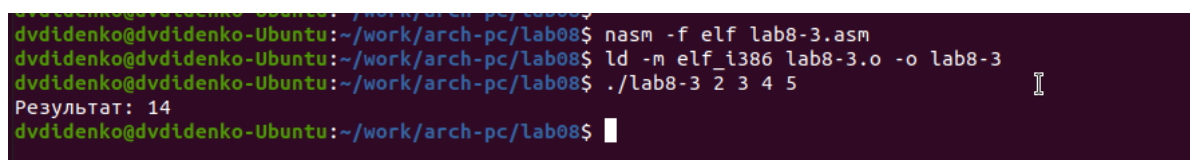
Рассмотрим еще один пример программы которая выводит сумму чисел, ко-

торые передаются в программу как аргументы. (рис. [2.9]) (рис. [2.10])



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab08
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab08/lab8-3.asm
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por 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`)
por 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
```

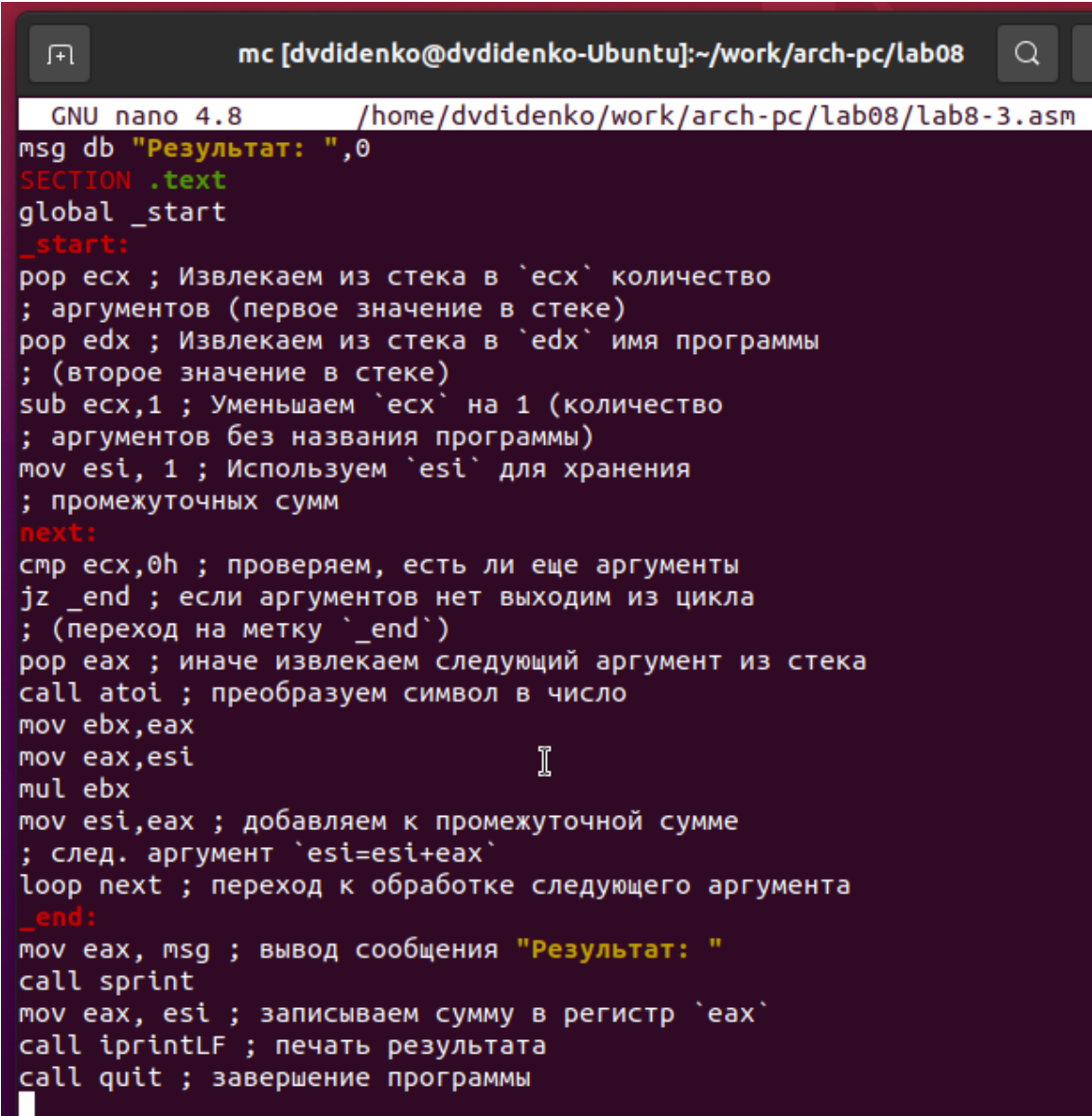
Рис. 2.9: Код программы lab8-3.asm



```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 2 3 4 5
Результат: 14
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$
```

Рис. 2.10: Компиляция и запуск программы lab8-3.asm

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



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab08
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab08/lab8-3.asm
msg db "Результат: ",0
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por 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`)
por 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: Код программы lab8-3.asm

```

dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./lab8-3 2 3 4 5
Результат: 120
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$

```

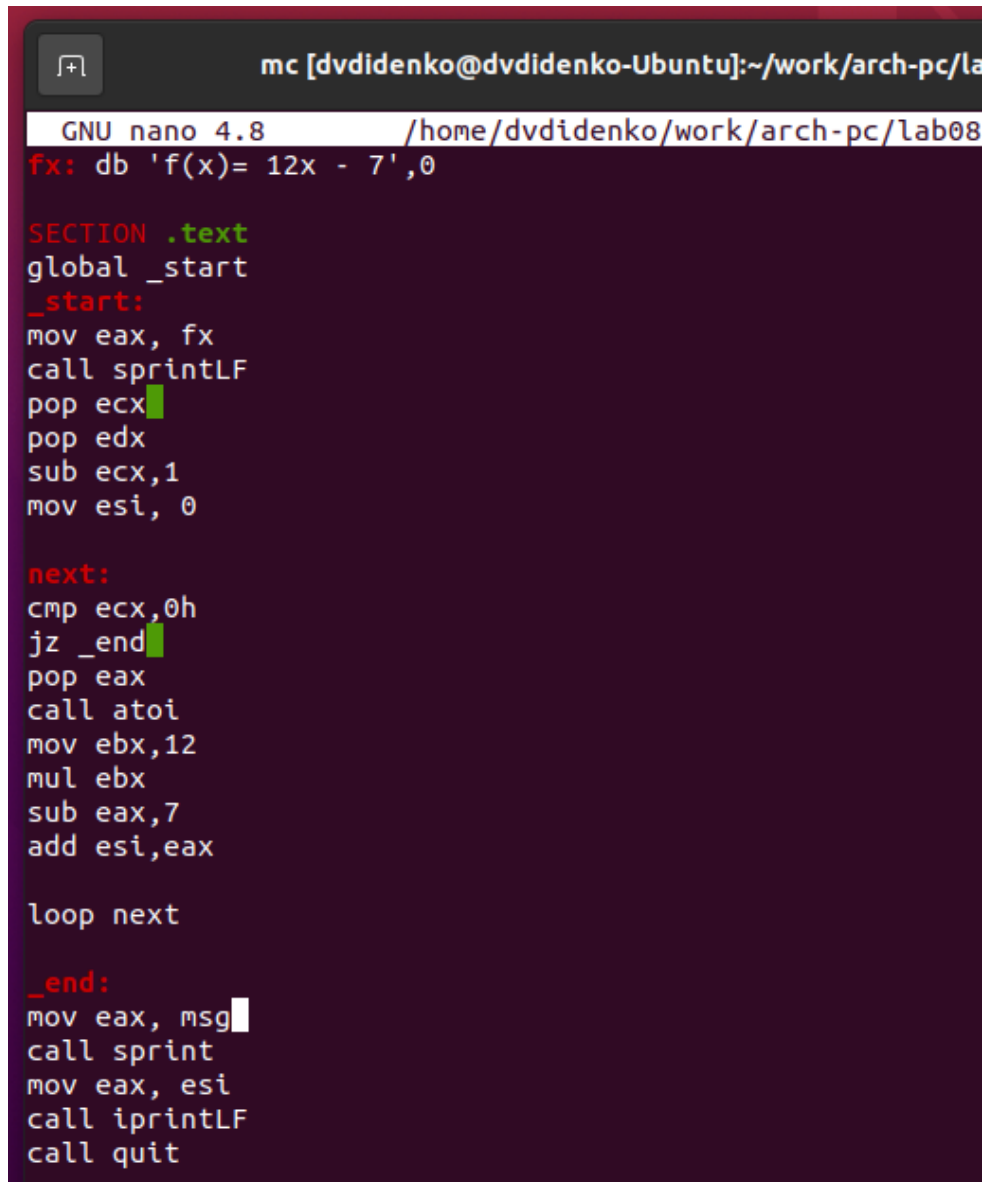
Рис. 2.12: Компиляция и запуск программы lab8-3.asm

2.3 Задание для самостоятельной работы

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])

Мой вариант 13:

$$f(x) = 12x - 7$$



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/la
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab08
fx: db 'f(x)= 12x - 7',0

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

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx,12
mul ebx
sub eax,7
add esi,eax

loop next

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

Рис. 2.13: Код программы prog.asm

Для проверки я запустил сначала с одним аргументом. Так, при подстановке $f(1) = 5, f(2) = 17$ Затем подал несколько аргументов и получил сумму значений функции.

```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ nasm -f elf prog.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 prog.o -o prog
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./prog
f(x)= 12x - 7
Результат: 0
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./prog 1
f(x)= 12x - 7
Результат: 5
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./prog 2
f(x)= 12x - 7
Результат: 17
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$ ./prog 6 2 6 4 6 1
f(x)= 12x - 7
Результат: 258
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab08$
```

Рис. 2.14: Компиляция и запуск программы prog.asm

3 Выводы

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