

# **Отчёт по лабораторной работе №8**

**Дисциплина: Архитектура компьютера**

-04-24

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Обработка аргументов командной строки . . . . .	11
3.2	Задание для самостоятельной работы . . . . .	15
<b>4</b>	<b>Выводы</b>	<b>18</b>

# Список иллюстраций

3.1	Текст программы . . . . .	7
3.2	Работа программы . . . . .	8
3.3	Изменения в программе . . . . .	8
3.4	Работа программы . . . . .	10
3.5	Работа программы после изменений . . . . .	11
3.6	Текст программы . . . . .	12
3.7	Работа программы . . . . .	12
3.8	Текст программы . . . . .	13
3.9	Результат . . . . .	13
3.10	Текст программы . . . . .	14
3.11	Результаты . . . . .	14
3.12	Код моей программы . . . . .	16
3.13	Работает успешно! . . . . .	17

## **Список таблиц**

# 1 Цель работы

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

## **2 Задание**

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Задание для самостоятельной работы

### 3 Выполнение лабораторной работы

Для начала создаю каталог для выполнения лаб. работы и в нем создаю файл lab08-1.asm. Далее ввожу в этот файл текст из листинга 8.1: (рис. 3.1).

```
%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. 3.1: Текст программы

Вот как работает эта программа: (рис. 3.2).

```

aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ld -m elf_i386 -o lab8 lab8.o
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./lab8
Введите N: 6
6
5
4
3
2
1

```

Рис. 3.2: Работа программы

Добавляю изменения в текст программы согласно методическим материалам (рис. 3.3).

```

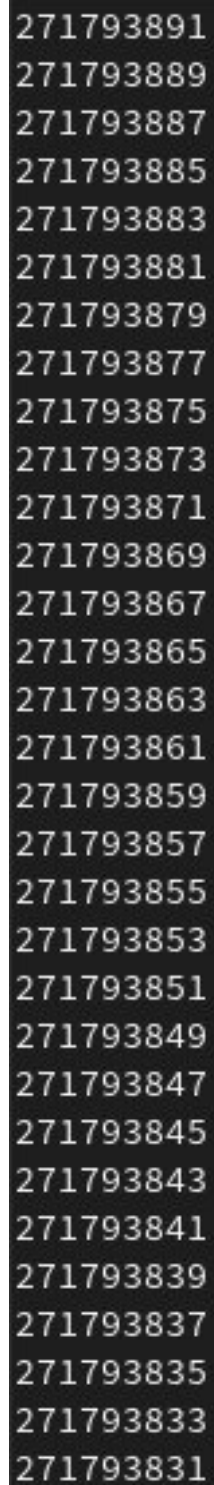
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; Вывод сообщения 'Введите N:'
mov eax,msg1
call sprint
; Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N], eax
;Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'.
label:
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения N
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'.
call quit

```

Рис. 3.3: Изменения в программе



Вот такой результат работы программы(рис. 3.4).



271793891  
271793889  
271793887  
271793885  
271793883  
271793881  
271793879  
271793877  
271793875  
271793873  
271793871  
271793869  
271793867  
271793865  
271793863  
271793861  
271793859  
271793857  
271793855  
271793853  
271793851  
271793849  
271793847  
271793845  
271793843  
271793841  
271793839  
271793837  
271793835  
271793833  
271793831

Рис. 3.4: Работа программы

Как мы видим, программа выводит огромное количество чисел, и, таким образом число проходов цикла гораздо больше чем введенное N.

Я вношу изменения в программу. добавляя команды push и pop. Запускаю программу и теперь все стало нормально. Количество проходов цикла соответствует введенному мной N(рис. 3.5).

```
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./lab8
Введите N: 8
7
5
3
1
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./lab8
Введите N: 6
5
3
1
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$
```

Рис. 3.5: Работа программы после изменений

### 3.1 Обработка аргументов командной строки

Создаю новый файл и ввожу в него текст листинга 8.2 (рис. 3.6).

```

#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. 3.6: Текст программы

Запускаю файл в работу, указав аргументы (рис. 3.7).

```

lab08-2 lab08-2.0
mesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./lab08-2 аргумент1 а
ргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3

```

Рис. 3.7: Работа программы

Можно сказать, что программой было обработано 4 аргумента.

Для рассмотрения следующего примера я создаю новый файл и ввожу туда текст листинга 8.3(рис. 3.8).

```

#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. 3.8: Текст программы

Результат работы получился соответствующим (рис. 3.9).

```

aesandane@fedora:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/labs/lab08/report/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aesandane@fedora:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/labs/lab08/report/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
aesandane@fedora:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/labs/lab08/report/lab08$

```

Рис. 3.9: Результат

Теперь нам нужно изменить текст программы под аргументы(рис. 3.10).

```

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    pop ecx        ; Извлекаем из стека в ecx количество аргументов
    pop edx        ; Извлекаем из стека в edx имя программы
    sub ecx, 1     ; Уменьшаем ecx на 1 (количество аргументов без названия программы)
    ....
    mov esi, 1     ; Используем esi для хранения произведения, начинаем с 1
next:
    cmp ecx, 0h    ; проверяем, есть ли еще аргументы
    jz _end        ; если аргументов нет, выходим из цикла

    pop eax        ; извлекаем следующий аргумент из стека
    call atoi      ; преобразуем символ в число
    imul esi, eax   ; умножаем промежуточное произведение на текущее значение

    loop next      ; переход к обработке следующего аргумента

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

```

Рис. 3.10: Текст программы

Как видим текст программы изменен успешно и у нас все работает (рис. 3.11).

```

aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ nasm -f elf lab8-3.asm
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ █

```

Рис. 3.11: Результаты

## 3.2 Задание для самостоятельной работы

Для выполнения самостоятельного задания я создам файл `poter.asm` в том же каталоге, где выполняла предыдущие задания. Суть задания заключается в том, что я должна самостоятельно написать такую программу, которая будет выводить сумму значений функций исходя из того, какие переменные я буду вводить в программу (как аргументы). При выполнении лабораторной работы №7, я выполняла задания для 2 варианта, соответственно при выполнении этой лабораторной работы, я тоже буду выполнять 2 вариант.

Самостоятельно пишу код для того, чтоб программа успешно работала:(рис. 3.12).

```

SECTION .text
GLOBAL _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi

mov ebx, eax
add eax, ebx.
add eax, ebx
sub eax,1

add esi, eax
loop next
_end:
mov eax, msg1
call sprintLF
mov eax, msg2
call sprint
mov eax, esi
call iprintLF.
call quit

```

Рис. 3.12: Код моей программы

Послек долгих попыток написать правильный код, я к этому пришла. Вот таким образом работает моя программа:(рис. 3.13).



```

aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ nasm -f elf test.asm
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ld -m elf_i386 -o test test.o
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./test 2 2 2 2
Функция: f(x)=3x-1
Результат: 20
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./test 3 3
Функция: f(x)=3x-1
Результат: 16
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./test 5 7
Функция: f(x)=3x-1
Результат: 34
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ ./test 3 7
Функция: f(x)=3x-1
Результат: 28
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report/lab08$ █

```

Рис. 3.13: Работает успешно!

## 4 Выводы

Я приобрела навыки написания программ с использованием циклов и навыки обработки аргументов командной строки.