

# **Отчет по выполнению лабораторной работы №8**

**Дисциплина: архитектура компьютеров**

Чернятьева Олеся Олеговна

# 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	Реализация циклов в NASM . . . . .	7
3.2	Обработка аргументов командной строки . . . . .	11
<b>4</b>	<b>Выводы</b>	<b>18</b>

## Список иллюстраций

3.1	Создание каталога и файла . . . . .	7
3.2	Ввод программы из листинга 8.1 . . . . .	8
3.3	Проверка работы файла . . . . .	8
3.4	Изменение файла . . . . .	9
3.5	Проверка работы файла . . . . .	10
3.6	Внесение изменений в текст программы . . . . .	11
3.7	Запуск программы . . . . .	11
3.8	Создание файла lab8-2.asm . . . . .	11
3.9	Ввод программы из листинга 8.2 . . . . .	12
3.10	Запуск программы . . . . .	12
3.11	Ввод программы из листинга 8.3 . . . . .	13
3.12	Проверка работы файла . . . . .	13
3.13	Изменение текста листинга 8.3 . . . . .	14
3.14	Проверка работы программы . . . . .	14
3.15	Написание программы для самостоятельной работы . . . . .	15
3.16	Запуск программы . . . . .	16

## **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 Выполнение лабораторной работы

### 3.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm (рис. [3.1]).

```
oochernyatjeva@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/lab08/report $ mkdir ~/work/arch-pc/lab08
oochernyatjeva@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/lab08/report $ cd ~/work/arch-pc/lab08
oochernyatjeva@dk8n60 ~/work/arch-pc/lab08 $ touch lab8-1.asm
oochernyatjeva@dk8n60 ~/work/arch-pc/lab08 $ ls
lab8-1.asm
oochernyatjeva@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 3.1: Создание каталога и файла

Ввожу в файл lab8-1.asm текст программы из листинга 8.1.(рис. [3.2]).

```

1 ;-----
2 ; Программа вывода значений регистра 'ecx'
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
25 label:
26 mov [N],ecx
27 mov eax,[N]
28 call iprintLF ; Вывод значения 'N'
29 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
30 ; переход на 'label'
31 call quit

```

Рис. 3.2: Ввод программы из листинга 8.1

Созаю исполняемый файл и проверяю его работу.(рис. [3.3]).

```

oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 12
12
11
10
9
8
7
6
5
4
3
2
1
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $

```

Рис. 3.3: Проверка работы файла

Далее изменяю текст программы, добавив изменение значение регистра ecx в цикле.(рис. [3.4]).



```
4 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
5 label:
6 sub ecx,1 ; 'ecx=ecx-1'
7 mov [N],ecx
8 mov eax,[N]
9 call iprintLF
0 loop label
1 call quit
```

Рис. 3.4: Изменение файла

Создаю исполняемый файл и проверяю его работу.(рис. [3.5]).

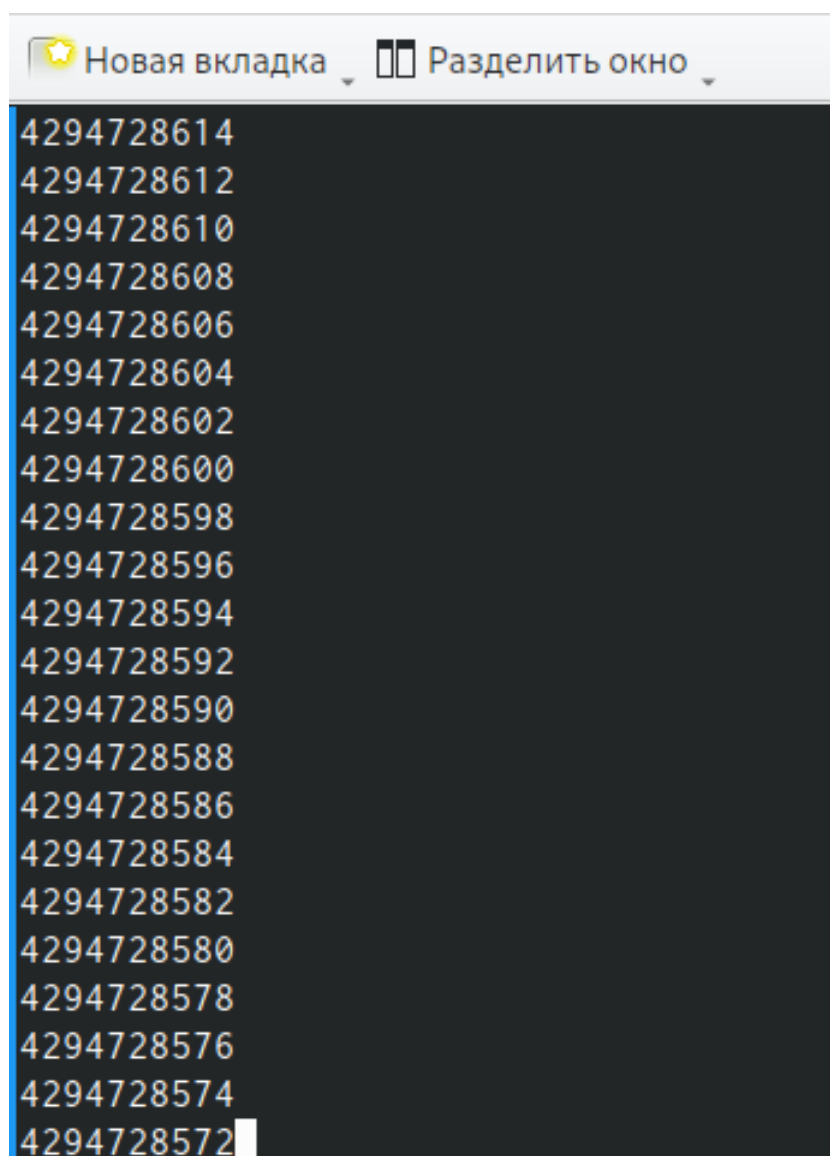


Рис. 3.5: Проверка работы файла

В этой программе число проходов цикла не соответствует значению N.

Вношу изменения в текст программы, добавив команды push и pop.(рис. [3.6]).

```

23 ,                организация цикла
24 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
25 label:
26 push ecx ; добавление значения ecx в стек
27 sub ecx,1
28 mov [N],ecx
29 mov eax,[N]
30 call iprintLF
31 pop ecx ; извлечение значения ecx из стека
32 loop label
33 call quit

```

Рис. 3.6: Внесение изменений в текст программы

Создаю исполняемый файл и запускаю его.(рис. [3.7]).

```

oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $

```

Рис. 3.7: Запуск программы

В данном случае число проходов цикла соответствует значению N.

## 3.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm.(рис. [3.8]).

```

oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ touch lab8-2.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $

```

Рис. 3.8: Создание файла lab8-2.asm

Ввожу в него программу из листинга 8.2.(рис. [3.9]).

```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в 'ecx' количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в 'edx' имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем 'ecx' на 1 (количество
10 ; аргументов без названия программы)
11 next:
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет выходим из цикла
14 ; (переход на метку '_end')
15 pop eax ; иначе извлекаем аргумент из стека
16 call sprintLF ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку 'next')
19 _end:
20 call quit
```

Рис. 3.9: Ввод программы из листинга 8.2

Создаю исполняемый файл и запускаю его, указав аргументы “12”, “39” и “1”.(рис. [3.10]).

```
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-2 12 39 1
12
39
1
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $
```

Рис. 3.10: Запуск программы

Программой было обработано 3 аргумента.

Создаю файл lab8-3.asm и ввожу в него программу из листинга 8.3.(рис. [3.11]).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в 'ecx' количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в 'edx' имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем 'ecx' на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем 'esi' для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку '_end')
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент 'esi=esi+eax'
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр 'eax'
28 call iprintLF ; печать результата
29 call quit ; завершение программы

```

Рис. 3.11: Ввод программы из листинга 8.3

Созаю исполняемый файл и запускаю его, указав аргументы.(рис. [3.2]).

```

oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-3
Результат: 0
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4
Результат: 10
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $

```

Рис. 3.12: Проверка работы файла

Программа работает.

Теперь изменяю текст программы из листинга 8.3 так, чтобы он вычислял произведение аргументов каждой строки.(рис. [3.13]).

```

%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
mul esi
mov 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. 3.13: Изменение текста листинга 8.3

Создаю исполняемый файл и запускаю его, указав аргументы.(рис. [3.14]).

```

oochernyatjeva@dk4n62 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4 5
Результат: 120
oochernyatjeva@dk4n62 ~/work/arch-pc/lab08 $ 

```

Рис. 3.14: Проверка работы программы

### #Задания для самостоятельной работы

Создаю файл lab8-4.asm и начинаю написание программы, которая находит сумму значений функции  $f(x)$  для своего варианта(вариант 16).(рис. [3.15]).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 msg2 db "Функция: f(x)=30x-11"
5 SECTION .text
6 global _start
7 _start:
8 mov eax,msg2
9 call sprintf
10 pop ecx
11 pop edx
12 sub ecx,1
13 mov esi, 0
14 mov edi, 30
15 next:
16 cmp ecx,0h
17 jz _end
18 pop eax
19 call atoi
20 mul edi
21 sub eax,11
22 add esi,eax
23 loop next
24 _end:
25 mov eax, msg
26 call sprintf
27 mov eax, esi
28 call iprintLF
29 call quit

```

Рис. 3.15: Написание программы для самостоятельной работы

Создаю исполняемый файл и запускаю его, указав аргументы.(рис. [3.16]).

```

oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4
Функция: f(x)=30x-11
Результат: 256
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $

```

Рис. 3.16: Запуск программы

Программы работает корректно.

Текст прогораммы:

`%include 'in_out.asm'`

`SECTION .data`

`msg db "Результат:",0`

`msg2 db "Функция: f(x)=30x-11"`

`SECTION .text`

`global _start`

`_start:`

`mov eax,msg2`

`call sprintLF`

`pop ecx`

`pop edx`

`sub ecx,1`

`mov esi, 0`

`mov edi, 30`

`next:`

`cmp ecx,0h`

`jz _end`

`pop eax`

`call atoi`

`mul edi`

`sub eax,11`

`add esi,eax`



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

## 4 Выводы

После выполнения данной лабораторной работы я научилась написанию программ с использованием циклов и обработкой аргументов командной строки.