

# **Отчёт по лабораторной работе №8**

**Дисциплина: Архитектура компьютера**

Кузьмина Мария Константиновна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Реализация циклов в NASM . . . . .	6
3.2	Обработка аргументов командной строки . . . . .	9
3.3	Выполнение заданий для самостоятельной работы. . . . .	12
<b>4</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

3.1	снимок экрана . . . . .	6
3.2	снимок экрана . . . . .	7
3.3	снимок экрана . . . . .	7
3.4	снимок экрана . . . . .	8
3.5	снимок экрана . . . . .	8
3.6	снимок экрана . . . . .	9
3.7	снимок экрана . . . . .	9
3.8	снимок экрана . . . . .	10
3.9	снимок экрана . . . . .	10
3.10	снимок экрана . . . . .	11
3.11	снимок экрана . . . . .	11
3.12	снимок экрана . . . . .	12
3.13	снимок экрана . . . . .	12
3.14	снимок экрана . . . . .	13
3.15	снимок экрана . . . . .	15

# 1 Цель работы

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

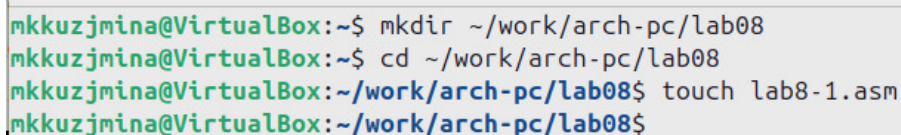
## 2 Задание

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

## 3 Выполнение лабораторной работы

### 3.1 Реализация циклов в NASM

Создаем директорию с помощью `mkdir`, переходим в нее и создаем файл `lab8-1.asm` (рис. 3.1):



```
mkkuzjmina@VirtualBox:~$ mkdir ~/work/arch-pc/lab08
mkkuzjmina@VirtualBox:~$ cd ~/work/arch-pc/lab08
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ touch lab8-1.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$
```

Рис. 3.1: снимок экрана

Вводим в файл `lab8-1.asm` текст программы из листинга 8.1 (рис. 3.2):

```

; -----
; Программа вывода значений регистра 'ecx'
; -----
%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]

```

Рис. 3.2: снимок экрана

Создаем исполняемый файл и запускаем его. Результат работы данной программы будет следующим (рис. 3.3):

```

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

```

Рис. 3.3: снимок экрана

Изменяем текст программы, добавив изменение значение регистра `ecx` в цикле:  
(рис. 3.4):

```
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. 3.4: снимок экрана

Создаем исполняемый файл и запускаем его. Видим, что вывод чисел происходит с шагом 1, количество проходов цикла уменьшается в 2 раза. (рис. 3.5):

```
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
5
3
1
```

Рис. 3.5: снимок экрана

Изменяем текст программы добавив команды `push` и `pop` для сохранения значения счетчика цикла `loop`: (рис. 3.6):



```

mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx    ; добавление значения ecx в стек
sub ecx,1   ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
pop ecx
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit

```

Рис. 3.6: снимок экрана

Создаем исполняемый файл и проверяем его работу. Видим, что число проходов цикла соответствует введенному с клавиатуры числу. (рис. 3.7):

```

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

```

Рис. 3.7: снимок экрана

## 3.2 Обработка аргументов командной строки

Создаем файл lab8-2.asm в каталоге и вводим в него текст программы из листинга 8.2. (рис. 3.8):

```

;-----
; Обработка аргументов командной строки
;-----
%include 'in_out.asm'
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)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
por 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. 3.8: снимок экрана

Создаем исполняемый файл и запускаем его, указав аргументы (аргумент1 аргумент 2 'аргумент 3') притом количество аргументов оставалось таким же, как и было введено : (рис. 3.9):

```

mkkuzmina@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
mkkuzmina@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
mkkuzmina@VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3

```

Рис. 3.9: снимок экрана

Создаем файл lab8-3.asm в каталоге и вводим в него текст программы из листинга 8.3 (рис. 3.10) и запускаем его (рис. 3.11)

```

#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 idprintLF ; печать результата

```

Рис. 3.10: снимок экрана

```

mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47

```

Рис. 3.11: снимок экрана

Изменяем текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. 3.12) и запускаем его (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
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
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

```

Рис. 3.12: снимок экрана

```

mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 5 8
Результат: 40

```

Рис. 3.13: снимок экрана

### 3.3 Выполнение заданий для самостоятельной работы.

Пишем программу, которая находит сумму значений функции для нескольких  $x$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Вид функции  $f(x)$  выберем из таблицы 8.1 вариантов заданий в соответствии с вариантом (15),

полученным при выполнении лабораторной работы № 7 (рис. 3.14). Создаем исполняемый файл и проверяем его работу на нескольких наборах  $x$  (рис. 3.15):

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция:  $f(x) = 6x + 13$ ", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintf

pop ecx
pop edx
sub ecx, 1
mov esi, 0

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

mov ebx, 6
mul ebx
add eax, 13
add esi, eax

loop next
```

Рис. 3.14: снимок экрана

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция:  $f(x) = 6x + 13$ ", 0
msg_result db "Результат: ", 0

SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg_func
```

```
call sprintf
```

```
pop ecx
```

```
pop edx
```

```
sub ecx, 1
```

```
mov esi, 0
```

```
next:
```

```
cmp ecx, 0
```

```
jz _end
```

```
pop eax
```

```
call atoi
```

```
mov ebx, 6
```

```
mul ebx
```

```
add eax, 13
```

```
add esi, eax
```

```
loop next
```

```
_end:
```

```
mov eax, msg_result
```

```
call sprintf
```

```
mov eax, esi
```

```
call iprintf
```

call quit

```
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
mkkuzjmina@VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция:  $f(x) = 6x + 13$ 
Результат: 112
```

Рис. 3.15: снимок экрана

## **4 Выводы**

В результате выполнения лабораторной программы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.