

# **Отчёт по лабораторной работе №8**

**Дисциплина: Архитектура компьютера**

Пономарева Татьяна Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</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 . . . . .	10
<b>4</b>	<b>Задание для самостоятельной работы</b>	<b>13</b>
<b>5</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

3.1	Терминал. Создание каталога lab08. Создание файла lab8-1.asm . .	7
3.2	Окно Midnight Commander. Содержание файла lab8-1.asm . . . . .	7
3.3	Терминал. Создание исполняемого файла lab8-1. Проверка работы lab8-1 . . . . .	8
3.4	Окно Midnight Commander. Содержание измененного файла lab8-1.asm	8
3.5	Терминал. Создание исполняемого файла lab8-1. Проверка работы lab8-1 . . . . .	9
3.6	Окно Midnight Commander. Содержание измененного файла lab8-1.asm	9
3.7	Терминал. Создание исполняемого файла lab8-1. Проверка работы lab8-1 . . . . .	10
3.8	Создание файла lab8-2.asm. Содержание файла lab8-2.asm . . . . .	10
3.9	Терминал. Компиляция исполняемого файла lab8-2. Проверка работы lab8-2 . . . . .	11
3.10	Создание файла lab8-3.asm. Содержание файла lab8-3.asm . . . . .	11
3.11	Терминал. Создание исполняемого файла lab8-3. Проверка работы lab8-3 . . . . .	11
3.12	Окно Midnight Commander. Содержание измененного файла lab8-3.asm	12
3.13	Терминал. Создание исполняемого файла lab8-3. Проверка работы lab8-3 . . . . .	12
4.1	Терминал. Создание исполняемого файла lab8-4. Проверка работы lab8-4 . . . . .	14

## **Список таблиц**

# 1 Цель работы

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

## 2 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды.

## 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).

```
taonomareva@2c7fe9w:~$ mkdir ~/work/arch-pc/lab08
taonomareva@2c7fe9w:~$ cd ~/work/arch-pc/lab08
taonomareva@2c7fe9w:~/work/arch-pc/lab08$ touch lab8-1.asm
```

Рис. 3.1: Терминал. Создание каталога lab08. Создание файла lab8-1.asm

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

```
GNU nano 7.2 /home/taonomareva/work/arch-pc/lab08/lab8-1.asm
;-----
; 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]
call iprintLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit
```

Рис. 3.2: Окно Midnight Commander. Содержание файла lab8-1.asm

Создаю исполняемый файл и проверяю его работу (рис. 3.3). Программа демонстрирует работу циклов в NASM.

```
taonomareva@2c7fe9w:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
taonomareva@2c7fe9w:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
taonomareva@2c7fe9w:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
```

Рис. 3.3: Терминал. Создание исполняемого файла lab8-1. Проверка работы lab8-1

Изменяю текст программы, добавив изменение значения регистра ecx в цикле (рис. 3.4).

```
GNU nano 7.2 /home/taonomareva/work/arch-pc/lab08/lab8-1.asm
;-----
; 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:
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintfLF
loop label
call quit
```

Рис. 3.4: Окно Midnight Commander. Содержание измененного файла lab8-1.asm

Создаю исполняемый файл с измененным содержанием и проверяю его работу (рис. 3.5). Поскольку значение регистра ecx уменьшается на 2 в каждой итерации, общее количество итераций сокращается вдвое. Число проходов цикла не соответствует значению N, введенному с клавиатуры.



```

taonomareva@2c7fe9w:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
taonomareva@2c7fe9w:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
taonomareva@2c7fe9w:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1

```

Рис. 3.5: Терминал. Создание исполняемого файла lab8-1. Проверка работы lab8-1

Внесу изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop (рис. 3.6).

```

GNU nano 7.2 /home/taonomareva/work/arch-pc/lab08/lab8-1.asm
;-----
; 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:
push ecx
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label
call quit

```

Рис. 3.6: Окно Midnight Commander. Содержание измененного файла lab8-1.asm

Снова создаю исполняемый файл и проверяю его работу (рис. 3.7). Заметим, что число проходов цикла соответствует значению N, введенному с клавиатуры, но выводимые значения смещены на -1.

```

taonomareva@2c7fe9w:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
taonomareva@2c7fe9w:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
taonomareva@2c7fe9w:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0

```

Рис. 3.7: Терминал. Создание исполняемого файла lab8-1. Проверка работы lab8-1

## 3.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы из листинга 8.2 (рис. 3.8).

```

taonomareva@2c7fe9w:~/work/arch-pc/lab08$ touch lab8-2.asm
GNU nano 7.2 /home/taonomareva/work/arch-pc/lab08/lab8-2.asm
;
;-----
; Обработка аргументов командной строки
;-----
%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 printf  ; вызываем функцию печати
loop next    ; переход к обработке следующего
              ; аргумента (переход на метку `next`)
_end:
call quit

```

Рис. 3.8: Создание файла lab8-2.asm. Содержание файла lab8-2.asm

Компилирую исполняемый файл и проверяю его выполнение (рис. 3.9). Программой было обработано 3 аргумента.

```

taponomareva@2c7fe9w:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3

```

Рис. 3.9: Терминал. Компиляция исполняемого файла lab8-2. Проверка работы lab8-2

Создаю файл lab8-3.asm в каталоге ~/work/archpc/lab08 и ввожу в него текст программы из листинга 8.3 (рис. 3.10).

```

taponomareva@2c7fe9w:~/work/arch-pc/lab08$ touch lab8-3.asm
GNU nano 7.2 /home/taponomareva/work/arch-pc/lab08/lab8-3.asm
#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. 3.10: Создание файла lab8-3.asm. Содержание файла lab8-3.asm

Создаю исполняемый файл и запускаю его, указав аргументы (рис. 3.11). Результат программы равен 47.

```

taponomareva@2c7fe9w:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47

```

Рис. 3.11: Терминал. Создание исполняемого файла lab8-3. Проверка работы lab8-3

Изменяю текст программы из листинга 8.3 для вычисления произведения

аргументов командной строки (рис. 3.12).

```
GNU nano 7.2 /home/taponomareva/work/arch-pc/lab08/lab8-3.asm
#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 имя программы
    sub ecx,1    ; Уменьшаем `ecx` на 1 (учитываем только аргументы)
    mov esi, 1   ; Инициализируем `esi` как 1 для корректного умножения

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 `esi` на `eax` (`esi = esi * eax`)
    loop next    ; Переход к обработке следующего аргумента

_end:
    mov eax, msg ; Вывод сообщения "Результат: "
    call sprint
    mov eax, esi ; Передаем результат умножения в `eax`
    call iprintLF ; Печать результата
    call quit    ; Завершение программы
```

Рис. 3.12: Окно Midnight Commander. Содержание измененного файла lab8-3.asm

Создаю исполняемый файл и запускаю его (рис. 3.13). Программа перемножает вводимые числа.

```
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ ./lab8-3 1 23 5
Результат: 115
```

Рис. 3.13: Терминал. Создание исполняемого файла lab8-3. Проверка работы lab8-3

## 4 Задание для самостоятельной работы

У меня вариант №3.

Пишу программу, которая находит сумму значений функции  $f(x)=10x-5$  для  $x = x_1, x_2, x_3, \dots, x_n$ , т.е. выводит  $f(x_1)+f(x_2)+\dots+f(x_n)$ .

Код программы для варианта №3

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg_function db "Функция:  $f(x) = 10x - 5$ ", 0
```

```
msg_res db "Результат: ", 0
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
    mov eax, msg_function
```

```
    call printf
```

```
    pop ecx
```

```
    pop edx
```

```
    sub ecx, 1
```

```

mov esi, 0

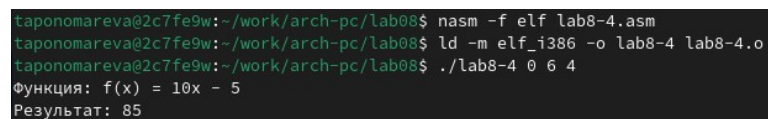
next:
    cmp ecx, 0h
    jz _end
    pop eax
    call atoi

    mov ebx, 10
    mul ebx
    sub eax, 5
    add esi, eax
    loop next

_end:
    mov eax, msg_res
    call sprint
    mov eax, esi
    call iprintLF
    call quit

```

Создаю исполняемый файл lab8-4 и запускаю его (рис. 4.1). Программа работает корректно (рис. 4.1).



```

taponomareva@2c7fe9w:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
taponomareva@2c7fe9w:~/work/arch-pc/lab08$ ./lab8-4 0 6 4
Функция: f(x) = 10x - 5
Результат: 85

```

Рис. 4.1: Терминал. Создание исполняемого файла lab8-4. Проверка работы lab8-4

Загружаю отчёт на GitHub.

## **5 Выводы**

В ходе выполнения лабораторной работы были приобретены навыки написания программ с использованием циклов и обработки аргументов командной строки.

# **Список литературы**

1. Курс на ТУИС
2. Лабораторная работа №8