

# **Отчёт по лабораторной работе 8**

**Архитектура компьютеров**

Хаммудех Салех

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 Выполнение лабораторной работы</b>	<b>6</b>
2.1 Реализация циклов в NASM . . . . .	6
2.2 Самостоятельное задание . . . . .	17
<b>3 Выводы</b>	<b>20</b>

# **Список иллюстраций**

2.1 Создан каталог . . . . .	6
2.2 Программа lab8-1.asm . . . . .	7
2.3 Запуск программы lab8-1.asm . . . . .	8
2.4 Программа lab8-1.asm . . . . .	9
2.5 Запуск программы lab8-1.asm . . . . .	10
2.6 Программа lab8-1.asm . . . . .	11
2.7 Запуск программы lab8-1.asm . . . . .	12
2.8 Программа lab8-2.asm . . . . .	13
2.9 Запуск программы lab8-2.asm . . . . .	13
2.10 Программа lab8-3.asm . . . . .	14
2.11 Запуск программы lab8-3.asm . . . . .	15
2.12 Программа lab8-3.asm . . . . .	16
2.13 Запуск программы lab8-3.asm . . . . .	16
2.14 Программа lab8-prog.asm . . . . .	18
2.15 Запуск программы lab8-prog.asm . . . . .	19

# **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

Создал каталог для программ лабораторной работы №8 и файл lab8-1.asm (рис. 2.1).



Рисунок 2.1: Создан каталог

При реализации циклов в NASM с использованием инструкции loop необходимо помнить, что эта инструкция использует регистр ecx в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра ecx.

Записал в файл lab8-1.asm текст программы из листинга 8.1 (рис. 2.2). Создал исполняемый файл и проверил его работу (рис. 2.3).

The screenshot shows a text editor window titled "lab8-1.asm" with the file path "~/work/arch-pc/lab...". The code is written in assembly language and includes comments in Russian. The code reads a value from the user, converts it to an integer, and then prints it back out.

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

Matlab ▾ Tab Width: 8 ▾ Ln 1, Col 1 INS

Рисунок 2.2: Программа lab8-1.asm

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

Рисунок 2.3: Запуск программы lab8-1.asm

Данный пример демонстрирует, что изменение значения регистра ecx в теле цикла loop может привести к некорректной работе программы. Изменил текст программы, добавив изменение значения регистра ecx в цикле (рис. 2.4). Программа запускает бесконечный цикл при нечетном N и выводит только нечетные числа при четном N (рис. 2.5).

The screenshot shows a text editor window titled "lab8-1.asm" with the file path "~/work/arch-pc/lab...". The code is written in assembly language:

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

Рисунок 2.4: Программа lab8-1.asm

```
4294760110
4294760108
4294760106
4294760104
4294760102
4294760100
4294^C
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
hammudehsaleh@fedora:~/work/arch-pc/lab08$ █
```

Рисунок 2.5: Запуск программы lab8-1.asm

Для корректного использования регистра есх в цикле можно использовать стек. Внес изменения в текст программы, добавив команды push и pop для сохранения и восстановления значения счетчика цикла loop (рис. 2.6). Создал исполняемый файл и проверил его работу (рис. 2.7). Программа корректно выводит числа от N-1 до 0, при этом число проходов цикла соответствует N.

The screenshot shows a window titled "lab8-1.asm" with the following assembly code:

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

Рисунок 2.6: Программа lab8-1.asm

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

Рисунок 2.7: Запуск программы lab8-1.asm

Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и записал в него текст программы из листинга 8.2 (рис. 2.8). Скомпилировал исполняемый файл и запустил его с указанием аргументов. Программа обработала 4 аргумента – слова или числа, разделенные пробелом (рис. 2.9).

The screenshot shows a text editor window with the following interface elements:

- Top bar: "Open" (with a dropdown arrow), "+", "lab8-2.asm", "Save", menu icon, close button.
- File path: "~ / work / arch - pc / lab ...".

The assembly code in the editor is:

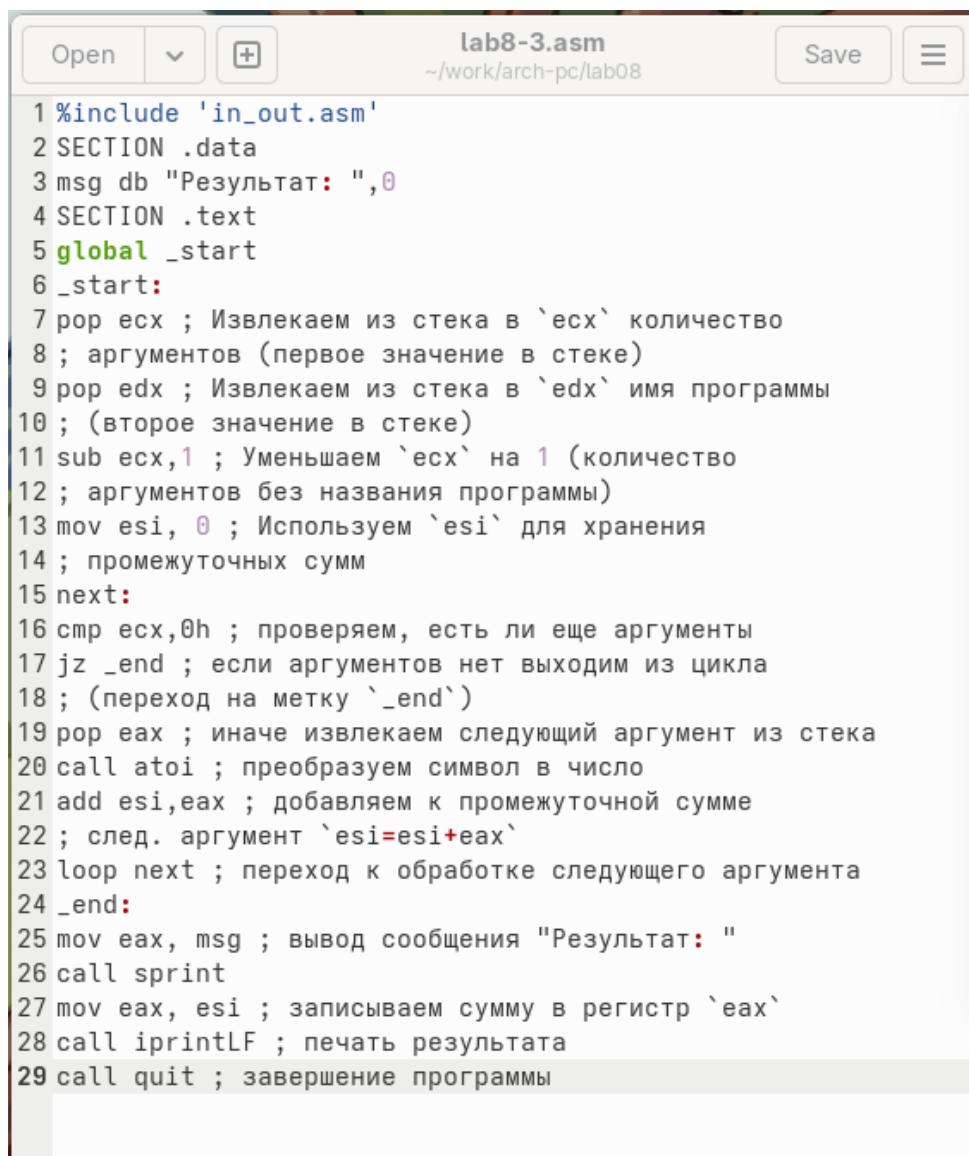
```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов 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
21
```

Рисунок 2.8: Программа lab8-2.asm

```
hammudehsaleh@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-2 argument1 argument 2 'argume
nt 3'
argument1
argument
2
argument 3
hammudehsaleh@fedora:~/work/arch-pc/lab08$
```

Рисунок 2.9: Запуск программы lab8-2.asm

Рассмотрел пример программы, которая вычисляет сумму чисел, переданных в программу в качестве аргументов (рис. 2.10, рис. 2.11).



The screenshot shows a text editor window with the following details:

- File name: lab8-3.asm
- Path: ~/work/arch-pc/lab08
- Buttons: Open, Save, and a menu icon.

The assembly code is as follows:

```
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
```

Рисунок 2.10: Программа lab8-3.asm

```
hammudehsaleh@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-3 4 5
Результат: 9
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-3 4 5 6 7 8 9
Результат: 39
hammudehsaleh@fedora:~/work/arch-pc/lab08$ █
```

Рисунок 2.11: Запуск программы lab8-3.asm

Изменил текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 2.12, рис. 2.13).

The screenshot shows a text editor window with the following details:

- File menu: Open, Save.
- File path: lab8-3.asm, ~work/arch-pc/lab08.
- Code content:

```
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, 1 ; 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 mov ebx,eax
22 mov eax,esi
23 mul ebx
24 mov esi,eax ; добавляем к промежуточной сумме
25 ; след. аргумент `esi=esi+eax`
26 loop next ; переход к обработке следующего аргумента
27 _end:
28 mov eax, msg ; вывод сообщения "Результат: "
29 call sprint
30 mov eax, esi ; записываем сумму в регистр `eax`
31 call iprintLF ; печать результата
32 call quit ; завершение программы
```

Рисунок 2.12: Программа lab8-3.asm

```
hammudehsaleh@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-3 4 5
Результат: 20
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-3 4 5 6 7 8 9
Результат: 60480
hammudehsaleh@fedora:~/work/arch-pc/lab08$
```

Рисунок 2.13: Запуск программы lab8-3.asm

## **2.2 Самостоятельное задание**

Написал программу, которая вычисляет сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , где значения  $x$  передаются как аргументы. Функция  $f(x)$  выбрана из таблицы 8.1 в соответствии с вариантом 6:

$$f(x) = 6x + 13.$$

Программа корректно работает, выводя сумму значений  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Создал исполняемый файл и проверил его работу на нескольких наборах  $x$  (рис. 2.14, рис. 2.15).

The screenshot shows a text editor window with the following details:

- File menu: Open, Save
- Title bar: lab8-prog.asm, ~/work/arch-pc/lab08
- Code area:

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 6x + 13',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintLF
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 mov ebx,6
22 mul ebx
23 add eax,13|           ; The cursor is here
24 add esi,eax
25
26 loop next
27
28 _end:
29 mov eax, msg
30 call sprint
31 mov eax, esi
32 call iprintLF
33 call quit
```

Рисунок 2.14: Программа lab8-prog.asm

```
hammudehsaleh@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-prog.asm
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-prog.o -o lab8-pr
og
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-prog
f(x)= 6x + 13
Результат: 0
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-prog 1
f(x)= 6x + 13
Результат: 19
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-prog 2
f(x)= 6x + 13
Результат: 25
hammudehsaleh@fedora:~/work/arch-pc/lab08$ ./lab8-prog 2 3 4 5
f(x)= 6x + 13
Результат: 136
hammudehsaleh@fedora:~/work/arch-pc/lab08$ █
```

Рисунок 2.15: Запуск программы lab8-prog.asm

Программа правильно считает, например,  $f(1) = 19$ ,  $f(2) = 25$ .

## **3 Выводы**

В ходе работы освоил использование стека, инструкции loop и работу с аргументами командной строки в языке ассемблера NASM.