

# **Отчет по лабораторной работе №8**

**Дисциплина: архитектура компьютера**

Ицков Андрей Станиславович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Реализация циклов в NASM . . . . .	8
4.2	Обработка аргументов командной строки . . . . .	11
4.3	Задание для самостоятельной работы . . . . .	13
<b>5</b>	<b>Выводы</b>	<b>16</b>
<b>6</b>	<b>Список литературы</b>	<b>17</b>

# Список иллюстраций

4.1	Создание каталога и файла . . . . .	8
4.2	Копирование программы из листинга . . . . .	8
4.3	Запуск программы . . . . .	9
4.4	Изменение программы . . . . .	9
4.5	Запуск программы . . . . .	10
4.6	Изменение программы . . . . .	10
4.7	Запуск программы . . . . .	10
4.8	Копирование программы из листинга . . . . .	11
4.9	Запуск программы . . . . .	11
4.10	Копирование программы из третьего листинга . . . . .	12
4.11	Запуск программы . . . . .	12
4.12	Изменение программы . . . . .	13
4.13	Запуск программы . . . . .	13
4.14	Текст программы . . . . .	14
4.15	Запуск программы . . . . .	15

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной работы

## 3 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. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

## 4 Выполнение лабораторной работы


### 4.1 Реализация циклов в NASM

Создаю каталог и пустой файл lab8-1.asm в нем (рис. -fig. 4.1).

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

Рис. 4.1: Создание каталога и файла

Вставляю в файл данную мне программу из листинга (рис. -fig. 4.2).



```
lab8-1.asm
~/work/arch-pc/lab08

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 'printf'; Вывод значения 'N'
26 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
27 ; переход на 'label'
28 call quit
```

Рис. 4.2: Копирование программы из листинга

Компилирую и запускаю программу (рис. -fig. 4.3).



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

Рис. 4.3: Запуск программы

Изменяю текст программы, добавляя изменение значения регистра `eax` в цикле (рис. -fig. 4.4).



```
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
```

Рис. 4.4: Изменение программы

Запускаю программу и вижу, что количество проходов уменьшилось вдвое (рис. -fig. 4.5).

```

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

```

Рис. 4.5: Запуск программы

Изменяю текст программы, добавляя команды push и pop (рис. -fig. 4.6).

```

Открыть ▼ + *lab8-1.asm Сохранить ☰ x
~/work/arch-pc/lab08
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 read
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

```

Рис. 4.6: Изменение программы

Запускаю программу и вижу, что количество проходов совпадает введенному N, но произошло смещение выводимых чисел на один (рис. -fig. 4.7).

```

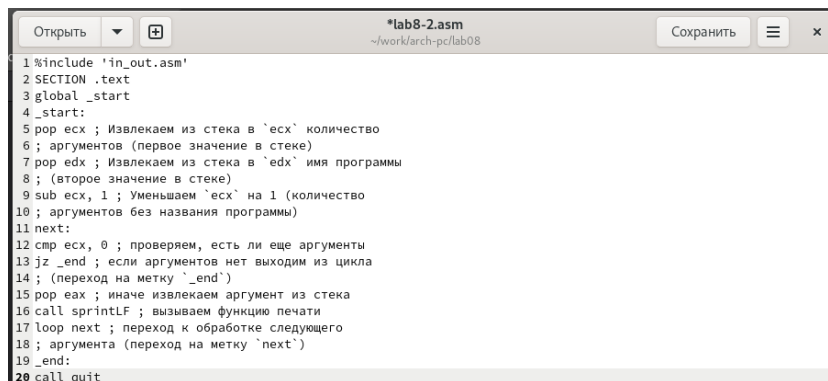
asitskov@vbox:~/work/arch-pc/lab08$ gedit lab8-1.asm
asitskov@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
asitskov@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
asitskov@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
asitskov@vbox:~/work/arch-pc/lab08$

```

Рис. 4.7: Запуск программы

## 4.2 Обработка аргументов командной строки

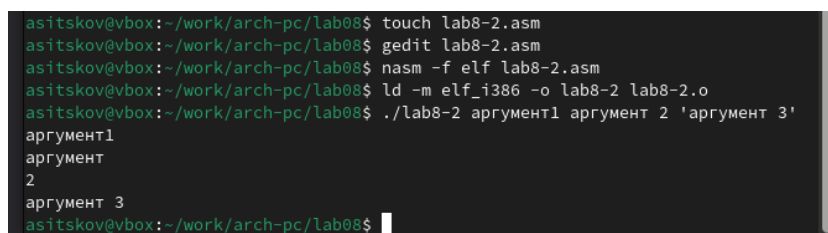
Создаю новый пустой файл и копирую в него программу из листинга (рис. -fig. 4.8).



```
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 sprintf ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку 'next')
19 _end:
20 call quit
```

Рис. 4.8: Копирование программы из листинга

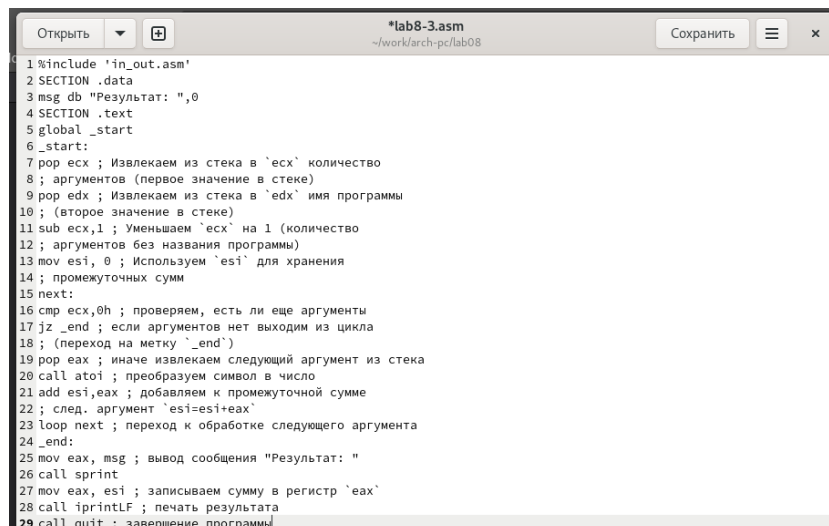
Компилирую и запускаю программу, указав аргументы (рис. -fig. 4.9).



```
asitskov@vbox: ~/work/arch-pc/lab08$ touch lab8-2.asm
asitskov@vbox: ~/work/arch-pc/lab08$ gedit lab8-2.asm
asitskov@vbox: ~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
asitskov@vbox: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
asitskov@vbox: ~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
asitskov@vbox: ~/work/arch-pc/lab08$
```

Рис. 4.9: Запуск программы

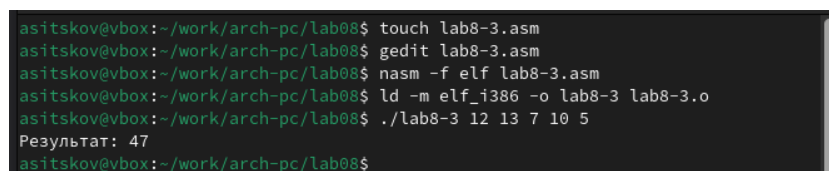
Создаю новый пустой файл и копирую в него программу из третьего листинга (рис. -fig. 4.10).



```
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. 4.10: Копирование программы из третьего листинга

Запускаю программу, добавив в аргументы данные мне числа (рис. -fig. 4.11).



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

Рис. 4.11: Запуск программы

Изменяю программу так, чтобы числа из аргументов умножались а не складывались (рис. -fig. 4.12).

Рис. 4.12: Изменение программы

Запускаю программу и вижу что числа из аргументов умножились (рис. -fig. 4.13).

Рис. 4.13: Запуск программы

## 4.3 Задание для самостоятельной работы

Пишу программму, которая находит сумму значений функции  $17 + 5x$  (данную для моего 18 варианта) (рис. -fig. 4.14).



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

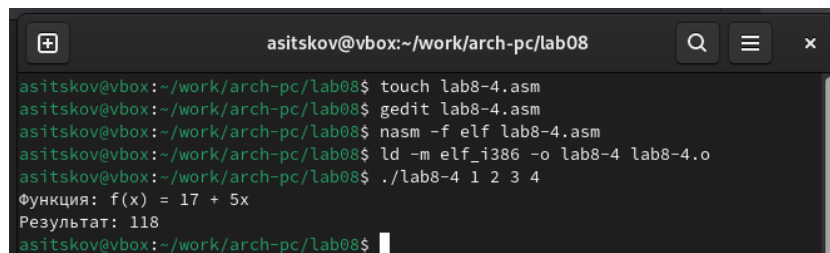
Рис. 4.14: Текст программы

Текст программы:

```
%include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 17 + 5x", 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, 0h
jz _end
pop eax
```

```
call atoi
mov ebx, 5
mul ebx
add eax, 17
add esi, eax
loop next
_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

Запускаю программу и ввожу несколько чисел, вижу, что программа работает корректно (рис. -fig. 4.15).



```
asitskov@vbox:~/work/arch-pc/lab08
asitskov@vbox:~/work/arch-pc/lab08$ touch lab8-4.asm
asitskov@vbox:~/work/arch-pc/lab08$ gedit lab8-4.asm
asitskov@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
asitskov@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
asitskov@vbox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: f(x) = 17 + 5x
Результат: 118
asitskov@vbox:~/work/arch-pc/lab08$
```

Рис. 4.15: Запуск программы

## **5 Выводы**

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.



## **6 Список литературы**

1. Курс на ТУИС
2. Лабораторная работа №8
3. Программирование на языке ассемблера NASM Столяров А. В.