

# **Отчёт по лабораторной работе №8**

**Дисциплина: архитектура компьютера**

Лысенко Маргарита 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>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выполнение самостоятельной работы</b>	<b>11</b>
<b>6</b>	<b>Выводы</b>	<b>13</b>

## Список иллюстраций

4.1	Запуск файла . . . . .	8
4.2	Запуск файла . . . . .	8
4.3	Изменение программы . . . . .	9
4.4	Запуск файла . . . . .	9
4.5	Запуск программы . . . . .	10
4.6	Запуск изменённой программы . . . . .	10
5.1	Запуск программы . . . . .	11

## Список таблиц

# 1 Цель работы

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

## 2 Задание

Создать программу, которая находит сумму значений функции  $f(x)$

### 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. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop).

## 4 Выполнение лабораторной работы

Создала каталог для программ лабораторной работы No 8, перешла в него и создала файл lab8-1.asm. Ввела в файл lab8-1.asm текст программы из листинга 8.1. Создала исполняемый файл и проверила его работу.(рис. 4.1).

```
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ touch lab8-1.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рис. 4.1: Запуск файла

Изменила текст программы добавив изменение значение регистра ecx в цикле:  
label: sub ecx,1 ; ecx=ecx-1 mov [N],ecx mov eax,[N] call iprintLF loop label  
программа работает некорректно(рис. 4.2).

```
4292603044
4292603042
4292603040
4292603038
4292603036
4292603034
4292603032
4292603030
4292603028
4292603026
```

Рис. 4.2: Запуск файла

Внесла изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:

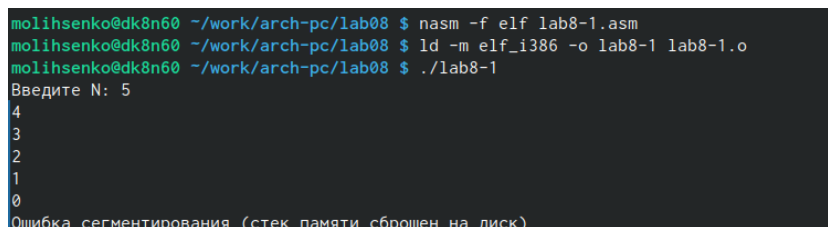


```

label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label

```

(рис. 4.3).



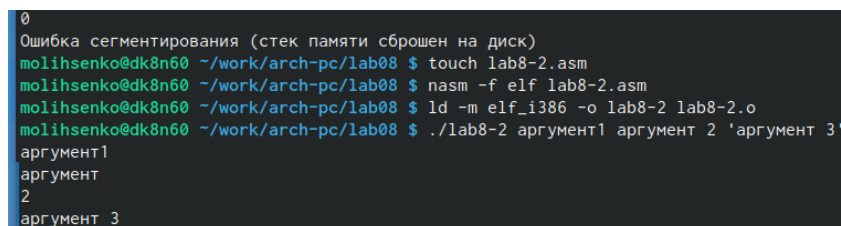
```

molihsenko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
4
3
2
1
0
Ошибка сегментирования (стек памяти сброшен на диск)

```

Рис. 4.3: Изменение программы

Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.2. Создала исполняемый файл и запустила его, указав аргументы: ./lab8-2 аргумент1 аргумент 2 'аргумент 3' (рис. 4.4).



```

0
Ошибка сегментирования (стек памяти сброшен на диск)
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ touch lab8-2.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3

```

Рис. 4.4: Запуск файла

Создала файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.3. (рис. 4.5).

```
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ touch lab8-3.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o main lab8-3.o
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
Результат: 47
```

Рис. 4.5: Запуск программы

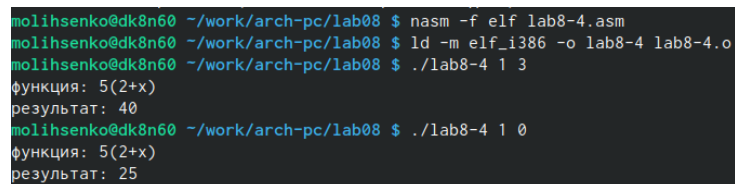
Изменила текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. 4.6).

```
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o main lab8-3.o
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
Результат: 54600
```

Рис. 4.6: Запуск изменённой программы

## 5 Выполнение самостоятельной работы

Написала программу, которая находит сумму значений функции  $f(x)$  с разными значениями  $x$  (рис. 5.1).



```
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-4 1 3
функция: 5(2+x)
результат: 40
molihsenko@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-4 1 0
функция: 5(2+x)
результат: 25
```

Рис. 5.1: Запуск программы

#Листинг

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

```
SECTION .data
```

```
f_x db "функция: 5(2+x)",0h
```

```
msg db 10,13,'результат: ',0h
```

```
SECTION .text
```

```
global _start
```

```
_start:
```

```
pop ecx
```

```
pop edx
```

```
sub ecx,1
```

```
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
;dec eax
add eax,2
mov ebx, 5
mul ebx
add esi, eax

loop next

_end:
mov eax, f_x
call sprint
mov eax, msg
call sprint
mov eax, esi
call iprintLF

call quit
```

## 6 Выводы

В проделанных лабораторной и самостоятельной работах мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.