### Отчёт по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Попутников Егор Сергеевич

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	10
5	Выводы	12

# Список иллюстраций

3.1	Создание файла	7
3.2	Создание исполняемого файла	7
3.3	Создание исполняемого файла	8
3.4	Создание файла листинга	8
3.5	Создание исполняемого файла	ç
3.6	Программа вывода суммы	Ç
3.7	Программа вывода произведения	ç
4 1	Самостоятельная работа	11

### Список таблиц

## 1 Цель работы

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

#### 2 Задание

1. Напишите программу, которая находит сумму значений функции f(x) для x = x1, x2, ..., xn, т.е. программа должна выводить значение f(x1) + f(x2) + ... + f(xn). Значения xi передаются как аргументы. Вид функции f(x) выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x = x1, x2, ..., xn.

#### 3 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm:(3.1)

```
egor@espoputnikov-dk3n56:~$ mkdir ~/work/arch-pc/lab08
egor@espoputnikov-dk3n56:~$ cd ~/work/arch-pc/lab08
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ touch lab8-1.asm
```

Рис. 3.1: Создание файла

Введем в файл lab8-1.asm текст программы из листинга 8.1. Создадим исполняемый файл и проверим его работу.(3.2)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ./lab8-1

BBEQUITE N: 10
10
9
8
7
6
5
4
3
2
1
```

Рис. 3.2: Создание исполняемого файла

Данный пример показывает, что использование регистра есх в теле цилка loop может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра есх в цикле:(3.3)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
```

Рис. 3.3: Создание исполняемого файла

Регистр есх принимает нечётные значения, число проходов цикла не соответствует значению N.

Для использования регистра есх в цикле и сохранения корректности работы программы можно использовать стек. Внесем изменения в текст программы добавив команды push и рор (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:(3.4)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ./lab8-1
BBEQUTE N: 10
9
8
7
6
5
4
3
2
1
0
```

Рис. 3.4: Создание файла листинга

В данном случае число проходов цикла соответствует значению N.

При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. При запуске программы в NASM аргументы командной строки загружаются в стек в обрат- ном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно из- влечь из стека

количество аргументов, а затем циклично для каждого аргумента выполнить логику программы.Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы из листинга 8.2. Создадим исполняемый файл и запустим его, указав аргументы:(3.5)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ touch lab8-2.asm

egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ./lab8-2 apryment1 apryment2 'apryment3'
apryment1
apryment2
apryment3
```

Рис. 3.5: Создание исполняемого файла

Программой было обработано 4 аргумента.

Создадим файл lab8-3.asm в каталоге ~/work/arch- pc/lab08 и введем в него текст программы, которая выводит сумму чисел, которые пере- даются в программу как аргументы.(3.6)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab0%$ touch lab8-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab0%$ gedit lab8-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab0%$ nasm -f elf lab8-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab0%$ ld -m elf_i386 -o lab8-3 lab8-3.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab0%$ ./lab8-3 17 13 270
Pesynbiai: 300
egor@espoputnikov-dk3n56:~/work/arch-pc/lab0%$
```

Рис. 3.6: Программа вывода суммы

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.(3.7)

```
egor@espoputnikov-dk3n56:-/work/arch-pc/lab0000 gedit lab00-3.asm
egor@espoputnikov-dk3n56:-/work/arch-pc/lab0000 nasm -f elf lab00-3.asm
egor@espoputnikov-dk3n56:-/work/arch-pc/lab0000 ld -m elf_i3000 -0 lab00-3 lab00-3.o
egor@espoputnikov-dk3n56:-/work/arch-pc/lab00000 ./lab00-3 10 3 7
Peay/nb7aT: 210
```

Рис. 3.7: Программа вывода произведения

# 4 Выполнение самостоятельной работы

Напишите программу, которая находит сумму значений функции f(x) для x = x1, x2, ..., xn, т.е. программа должна выводить значение f(x1) + f(x2) + ... + f(xn). Значения xi передаются как аргументы. Вид функции f(x) выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x = x1, x2, ..., xn.Вариант №4.(4.1)

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ", 0
4 SECTION .bss
5 prm: RESB 80
6 SECTION .text
7 global _start
8 _start:
9 pop ecx
10 pop edx
11 sub ecx, 1
12 mov esi, 2
13
14 next:
15 cmp ecx, 0
16 jz end
17 pop eax
18 call atoi
19 mul esi
20 sub eax,2
21 add [prm],eax
22 loop next
23 _end:
24 mov eax, msg
25 call sprint
26 mov eax,[prm]
27 call iprintLF
28 call quit
29
```

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ nasm -f elf variant4.asm egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ld -m elf_i386 -o variant4 varian t4.o egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ./variant4 1 2 3 4 Результат: 12 egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$
```

Рис. 4.1: Самостоятельная работа

# 5 Выводы

Я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.