

lab-09

Программирование цикла. Обработка аргументов командной строки.

Владимир Андреевич Баранов

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	14

Список иллюстраций

2.1	Создание файла.	6
2.2	Текст программы.	6
2.3	Проверка работы программы.	7
2.4	Текст программы.	7
2.5	Проверка работы программы.	8
2.6	Текст программы.	8
2.7	Проверка работы программы.	9
2.8	Текст программы.	10
2.9	Проверка работы программы.	10
2.10	Текст программы.	11
2.11	Проверка работы программы.	11
2.12	Текст программы.	12
2.13	Проверка работы программы.	12
2.14	Программа.	13
2.15	Проверка работы программы.	13

Список таблиц

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.


2 Выполнение лабораторной работы

1. Создаю каталог для лабораторной работы No 9, перехожу в него и создаю файл lab9-1.asm (рис. 2.1).

```
vabaranov@dk8n80 ~ $ mkdir ~/work/arch-pc/lab09
vabaranov@dk8n80 ~ $ cd ~/work/arch-pc/lab09
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ touch lab9-1.asm
```

Рис. 2.1: Создание файла.

2. Ввожу в файл lab9-1.asm текст программы из листинга 9.1. (рис. 2.2).



```
lab9-1.asm
~/work/arch-pc/lab09

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

Рис. 2.2: Текст программы.

3. Создаю исполняемый файл и проверяю его работу (рис. 2.3).

```

vabaranov@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 14
14
13
12
11
10
9
8
7
6
5
4
3
2
1
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ █

```

Рис. 2.3: Проверка работы программы.

4. Изменяю текст программы, добавив изменение значение регистра ecx в цикле (рис. 2.4).

```

Открыть ▼ + lab9-1.asm
~/work/arch-pc/lab09
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 call quit

```

Рис. 2.4: Текст программы.

5. Создаю исполняемый файл и проверяю его работу (рис. 2.5).

```
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 14
13
11
9
7
5
3
1
vabaranov@dk8n80 ~/work/arch-pc/lab09 $
```

Рис. 2.5: Проверка работы программы.

Число проходов цикла не соответствует значению N, введенному с клавиатуры.

6. Вношу изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop (рис. 2.6).

```
Открыть ▾ + lab9-1.asm
~/work/arch-pc/lab09
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
```

Рис. 2.6: Текст программы.

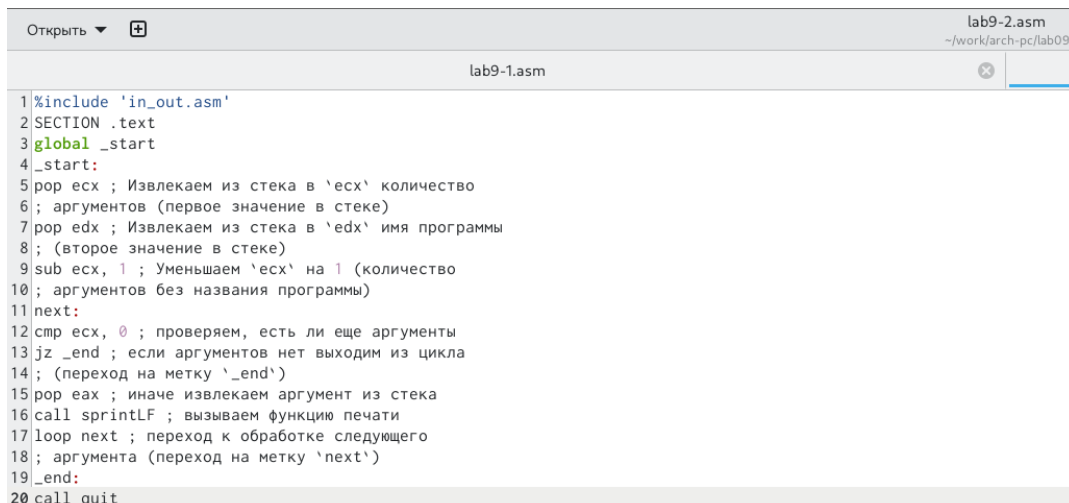
7. Создаю исполняемый файл и проверяю его работу (рис. 2.7).

```
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
vabaranov@dk8n80 ~/work/arch-pc/lab09 $
```

Рис. 2.7: Проверка работы программы.

В данном случае число проходов цикла соответствует значению N, введенному с клавиатуры.

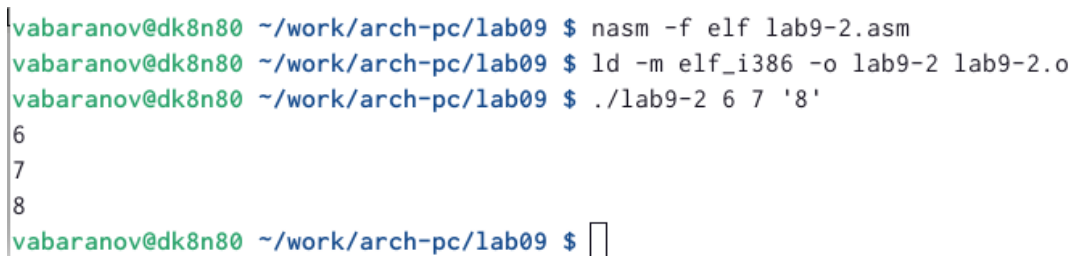
8. Создаю файл lab9-2.asm в каталоге ~/work/arch-pc/lab09 и ввожу в него текст программы из листинга 9.2. (рис. 2.8).



```
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. 2.8: Текст программы.

9. Создаю исполняемый файл и запускаю его, указав аргументы (рис. 2.9).




```
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab9-2.asm
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ./lab9-2 6 7 '8'
6
7
8
vabaranov@dk8n80 ~/work/arch-pc/lab09 $
```

Рис. 2.9: Проверка работы программы.

10. Создаю файл lab9-3.asm в каталоге ~/work/arch-pc/lab09 и ввожу в него текст программы из листинга 9.3. (рис. 2.10).

```


Открыть ▾  lab9-3.asm
~/work/arch-pc/lab09
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. 2.10: Текст программы.

11. Создаю исполняемый файл и запускаю его, указав аргументы (рис. 2.11).

```

vabaranov@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab9-3.asm
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ./lab9-3 36 24 7 33 50
Результат: 150
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ 

```

Рис. 2.11: Проверка работы программы.

12. Изменяю текст программы из листинга 9.3 для вычисления произведения аргументов командной строки (рис. 2.12).

```

Открыть ▾ + *lab9-3.asm  
~/work/arch-pc/lab09
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg DB "Результат: ",0
5
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10
11 pop ecx
12
13 pop edx
14
15 sub ecx,1
16
17 mov esi,1
18 mov eax,1
19
20 next:
21 cmp ecx,0
22 jz _end
23
24 pop eax
25 call atoi
26 mov ebx,eax
27 mov eax,esi
28 mul ebx
29 mov esi,eax
30 loop next
31
32
33
34 _end:
35 mov eax,msg
36 call sprint
37 mov eax,esi
38 call iprintLF
39
40 call quit

```

Рис. 2.12: Текст программы.

13. Создаю исполняемый файл и запускаю его, указав аргументы (рис. 2.13).

```

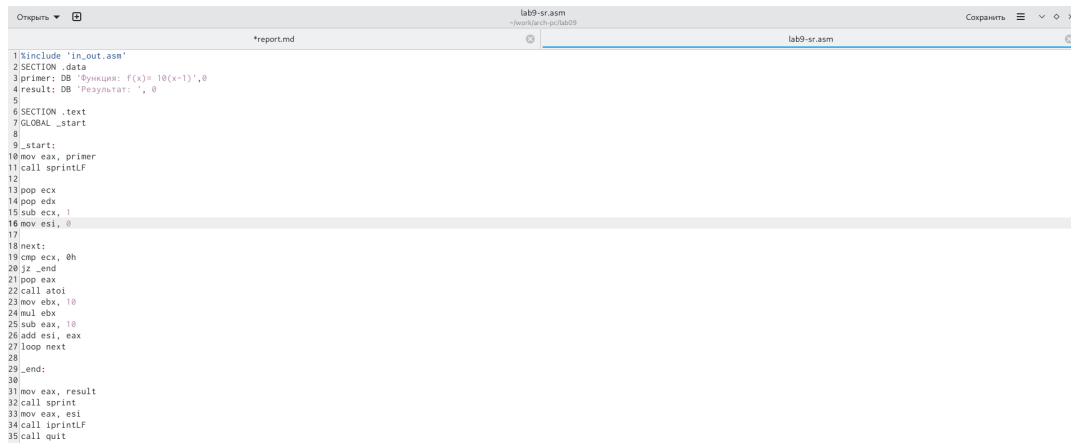
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab9-3.asm
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ ./lab9-3 36 24 7 33 50
Результат: 9979200
vabaranov@dk8n80 ~/work/arch-pc/lab09 $ 

```

Рис. 2.13: Проверка работы программы.

#Самостоятельная работа

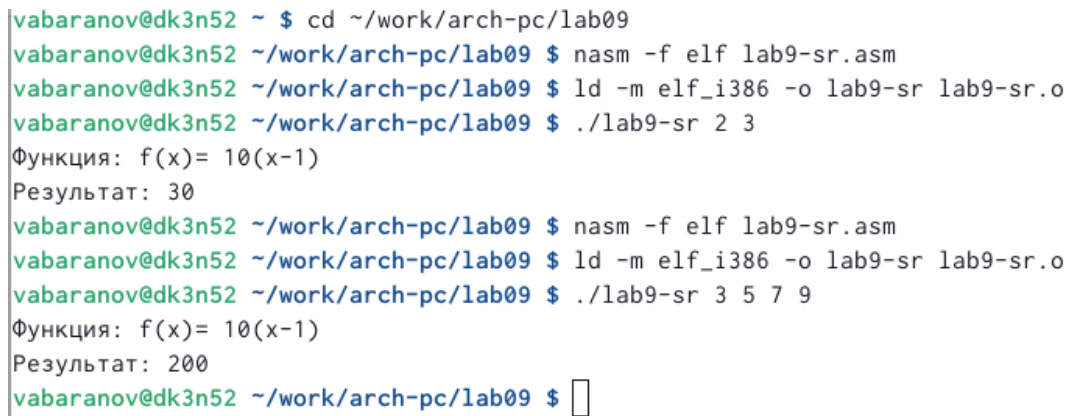
Пишу программу, которая находит сумму значений функции (рис. 2.14).



```
1 %include 'in_out.asm'
2 SECTION .data
3 primer: DB "Функция: f(x)= 10(x-1)",0
4 result: DB "Результат: ", 0
5
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10 mov eax, primer
11 call sprintf
12
13 pop ecx
14 pop edx
15 sub ecx, 1
16 mov esi, 0
17
18 next:
19 cmp ecx, 0h
20 ja _end
21 pop eax
22 call atoi
23 mov ebx, 10
24 mul ebx
25 sub eax, 10
26 add esi, eax
27 loop next
28
29 _end:
30
31 mov eax, result
32 call sprintf
33 mov eax, esi
34 call sprintf
35 call quit
```

Рис. 2.14: Программа.

Создаю исполняемый файл и проверяю его работу на нескольких наборах (рис. 2.15).



```
vabaranov@dk3n52 ~ $ cd ~/work/arch-pc/lab09
vabaranov@dk3n52 ~/work/arch-pc/lab09 $ nasm -f elf lab9-sr.asm
vabaranov@dk3n52 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-sr lab9-sr.o
vabaranov@dk3n52 ~/work/arch-pc/lab09 $ ./lab9-sr 2 3
Функция: f(x)= 10(x-1)
Результат: 30
vabaranov@dk3n52 ~/work/arch-pc/lab09 $ nasm -f elf lab9-sr.asm
vabaranov@dk3n52 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-sr lab9-sr.o
vabaranov@dk3n52 ~/work/arch-pc/lab09 $ ./lab9-sr 3 5 7 9
Функция: f(x)= 10(x-1)
Результат: 200
vabaranov@dk3n52 ~/work/arch-pc/lab09 $
```

Рис. 2.15: Проверка работы программы.

3 Выводы

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