

**Лабораторная работа №9.
Программирование цикла. Обработка
аргументов командной строки**

Дисциплина: Архитектура ЭВМ

Алексей Назаров НММбд-02-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Изменим текст программы следующим образом:	8
2.2	Изменим код и добавим push и pop.	10
2.3	Вывод на экран аргументов командной строки	11
2.4	Вычисление суммы аргументов командной строки.	11
2.5	Вычисление произведения аргументов командной строки . . .	12
3	Задание для самостоятельной работы.	14
4	Выводы	17

Список иллюстраций

2.1	Создание файла	6
2.2	Текст из листинга 9.1	7
2.3	Исполнение lab9-1.asm	7
2.4	Измененный текст программы	8
2.5	Запуск измененный программы	9
2.6	Измененный код lab9-1	10
2.7	Запустим измененный код	10
2.8	Текст lab9-2.asm	11
2.9	Запуск lab9-2	11
2.10	код lab9-3.asm	12
2.11	Запуск lab9-3	12
2.12	Измененный код lab9-3.1.asm	13
2.13	Запуск программы lab9-3.1	13
3.1	Код программы lab9-4.asm	15
3.2	Запуск lab9-4	16

Список таблиц

1 Цель работы

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

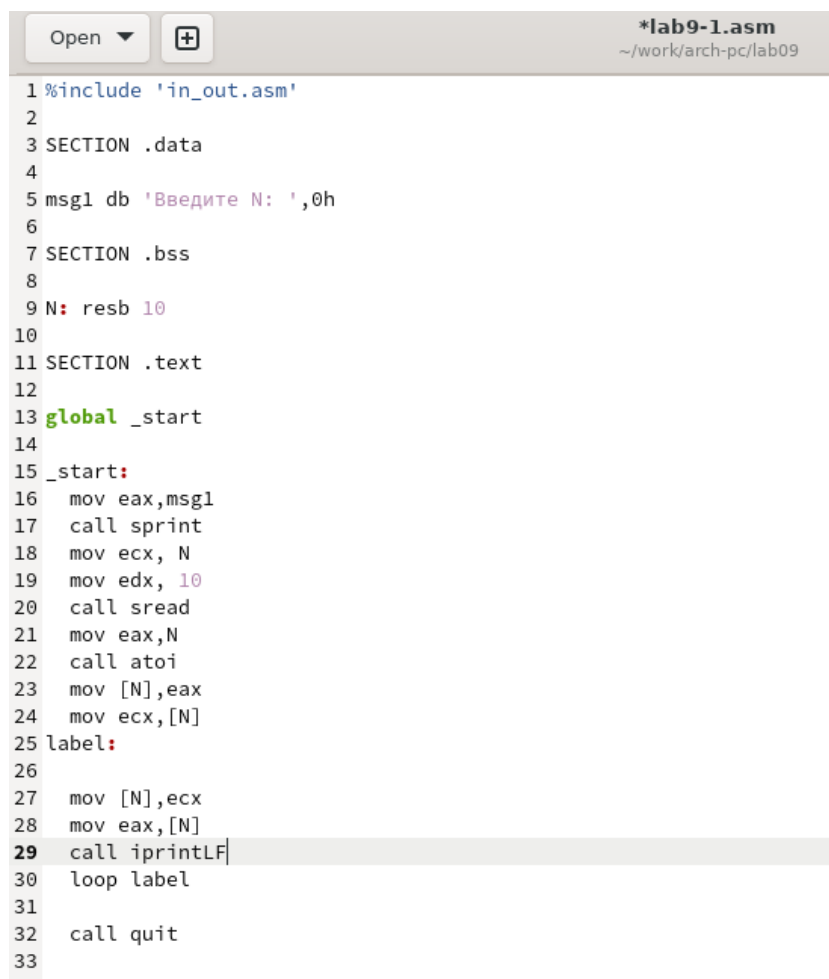
2 Выполнение лабораторной работы

Создадим каталог для программ лабораторной №9, создадим в нем файл.

```
[amnazarov@localhost ~]$ mkdir ~/work/arch-pc/lab09  
cd ~/work/arch-pc/lab09  
touch lab9-1.asm  
[amnazarov@localhost lab09]$
```

Рис. 2.1: Создание файла

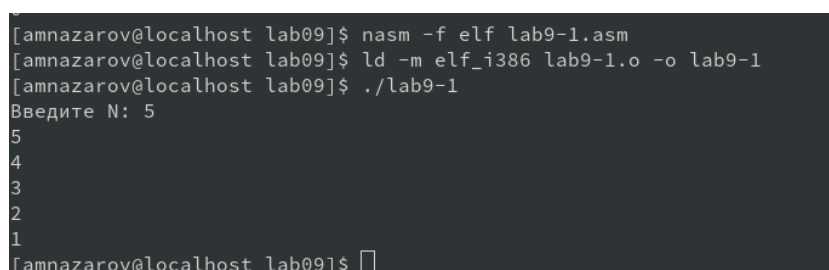
Введем код из листинга 9.1



```
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 msg1 db 'Введите N: ',0h
6
7 SECTION .bss
8
9 N: resb 10
10
11 SECTION .text
12
13 global _start
14
15 _start:
16     mov eax,msg1
17     call sprint
18     mov ecx, N
19     mov edx, 10
20     call sread
21     mov eax,N
22     call atoi
23     mov [N],eax
24     mov ecx,[N]
25 label:
26
27     mov [N],ecx
28     mov eax,[N]
29     call iprintLF
30     loop label
31
32     call quit
33
```

Рис. 2.2: Текст из листинга 9.1

Запустим код и введем 5



```
[amnazarov@localhost lab09]$ nasm -f elf lab9-1.asm
[amnazarov@localhost lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[amnazarov@localhost lab09]$ ./lab9-1
Введите N: 5
5
4
3
2
1
[amnazarov@localhost lab09]$
```

Рис. 2.3: Исполнение lab9-1.asm

2.1 Изменим текст программы следующим образом:

Добавим строчку с уменьшением ecx

```
27  
28     sub ecx, 1  
29     mov [N],ecx  
30     mov eax,[N]  
31     call iprintLF  
32
```

Рис. 2.4: Измененный текст программы

Исполним программу.


```
4294962454
4294962452
4294962450
4294962448
4294962446
4294962444
4294962442
4294962440
4294962438
4294962436
4294962434
4294962432
4294962430
4294962428
4294962426
4294962424
4294962422
4294962420
4294962418
4294962416
4294962414
4294962412
4294962410
4294962408
4294962406
4294962404
429496240^C
[amnazarov@localhost lab09]$
```

Рис. 2.5: Запуск измененный программы

Как видим, программа начала выводить очень большие числа, потому что произошло переполнение буфера. Когда мы от 0 отняли 1.

2.2 Изменим код и добавим push и pop.

```
26
27  push ecx
28  sub ecx, 1
29  mov [N],ecx
30  mov eax,[N]
31  call iprintLF
32  pop ecx
33
34  loop label
35  call quit
36
```

Рис. 2.6: Измененный код lab9-1

Исполним и посмотрим на результат.

```
[amnazarov@localhost lab09]$ nasm -f elf lab9-1.asm
[amnazarov@localhost lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[amnazarov@localhost lab09]$ ./lab9-1
Введите N: 5
4
3
2
1
0
```

Рис. 2.7: Запустим измененный код

Видим, что выводится 4 3 2 1, так как мы уменьшили значение ecx на 1, но переполнения буфера не случилось.

2.3 Вывод на экран аргументов командной строки

Введем в файл lab9-2.asm листинг 9.2

```
1 %include 'in_out.asm'
2
3 SECTION .text
4 global _start
5
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 next:
14     cmp ecx, 0 ; проверяем, есть ли еще аргументы
15     jz _end ; если аргументов нет выходим из цикла
16            ; (переход на метку `_end`)
17     pop eax ; иначе извлекаем аргумент из стека
18     call printf ; вызываем функцию печати
19     loop next ; переход к обработке следующего
20              ; аргумента (переход на метку `next`)
21 _end:
22     call quit
23
```

Рис. 2.8: Текст lab9-2.asm

Странслируем, слинкуем и запустим программу.

```
[amnazarov@localhost lab09]$ nasm -f elf lab9-2.asm
[amnazarov@localhost lab09]$ ld -m elf_i386 lab9-2.o -o lab9-2
[amnazarov@localhost lab09]$ ./lab9-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
2
аргумент3
[amnazarov@localhost lab09]$
```

Рис. 2.9: Запуск lab9-2

Для четырех аргументов вывелось 4 строки.

2.4 Вычисление суммы аргументов командной строки.

Напишем текст из листинга 9.3 в файл lab9-3.asm.

```

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
25 _end:
26  mov eax, msg ; вывод сообщения "Результат: "
27  call sprint
28  mov eax, esi ; записываем сумму в регистр `eax`
29  call iprintLF ; печать результата
30
31  call quit ; завершение программы
32

```

Рис. 2.10: код lab9-3.asm

Скомпилируем и запустим lab9-3 с аргументами (12 13 7 10 5)

```

[amnazarov@localhost lab09]$ nasm -f elf lab9-3.asm
[amnazarov@localhost lab09]$ ld -m elf_i386 lab9-3.o -o lab9-3
[amnazarov@localhost lab09]$ ./lab9-3 12 13 7 10 5
Результат: 47
[amnazarov@localhost lab09]$

```

Рис. 2.11: Запуск lab9-3

Программа вывела сумму аргументов

2.5 Вычисление произведения аргументов командной строки

Скопируем файл lab9-3.asm в lab9-3.1.asm и изменим следующие строчки:

```

19  pop eax ; иначе извлекаем следующий аргумент из стека
20  call atoi ; преобразуем символ в число
21  mul esi ; умножаем промежуточной сумме
22  mov esi, eax ; след. аргумент `esi=esi*eax`
23  loop next : переход к обработке следвющего аргумента

```

Рис. 2.12: Измененный код lab9-3.1.asm

Запустим программу lab9-3.1 с аргументами 1 2 3 4 и 1 2 3 4 5

```

[amnazarov@localhost lab09]$ nasm -f elf lab9-3.1.asm
[amnazarov@localhost lab09]$ ld -m elf_i386 lab9-3.1.o -o lab9-3.1
[amnazarov@localhost lab09]$ ./lab9-3.1 1 2 3 4
Результат: 24
[amnazarov@localhost lab09]$ ./lab9-3.1 1 2 3 4 5
Результат: 120
[amnazarov@localhost lab09]$ █

```

Рис. 2.13: Запуск программы lab9-3.1

3 Задание для самостоятельной работы.

Мой вариант 13, поэтому напишем программу которая будет вычислять сумму функций $f(x) = 12x - 7$ от значений аргументов, введенных в командной строке.

Напишем программу в файл lab9-4.asm по вычислению этой функции

```

1 %include 'in_out.asm'
2
3
4 SECTION .data
5 func db "Функция: f(x)=12x-7",0
6 msg db "Результат: ",0
7 SECTION .text
8
9 global _start
10
11 _start:
12     pop ecx    ; Извлекаем из стека в `ecx` количество
13                ; аргументов (первое значение в стеке)
14     pop edx    ; Извлекаем из стека в `edx` имя программы
15                ; (второе значение в стеке)
16     sub ecx,1  ; Уменьшаем `ecx` на 1 (количество
17                ; аргументов без названия программы)
18     mov esi, 0 ; Используем `esi` для хранения
19                ; промежуточных сумм
20     mov eax, func
21
22     call sprintf
23
24
25 next:
26     cmp ecx,0h
27     jz _end
28
29     pop eax
30     call atoi
31
32     push ecx
33     mov ecx,12
34     mul ecx
35     sub eax, 7
36     pop ecx
37
38     add esi,eax
39
40     loop next
41
42 _end:
43
44     mov eax, msg ; вывод сообщения "Результат: "
45     call sprint
46     mov eax, esi ; записываем сумму в регистр `eax`
47     call iprintLF ; печать результата
48
49     call quit ; завершение программы
50

```

Рис. 3.1: Код программы lab9-4.asm

Скомпилируем и запустим программу с аргументами 1 2 3 4.

```
[amnazarov@localhost lab09]$ nasm -f elf lab9-4.asm  
[amnazarov@localhost lab09]$ ld -m elf_i386 lab9-4.o -o lab9-4  
[amnazarov@localhost lab09]$ ./lab9-4 1 2 3 4  
Функция:  $f(x)=12x-7$   
Результат: 92
```

Рис. 3.2: Запуск lab9-4

4 Выводы

Мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.