

Лабораторная работа №9

Программирование цикла. Обработка аргументов командной строки

Вершинина Ангелина Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	11
4.3	Задание для самостоятельной работы	14
5	Выводы	17
	Список литературы	18

Список иллюстраций

4.1	Создание каталога и файла	8
4.2	Текст программы	9
4.3	Создание исполняемого файла и его работа	9
4.4	Изменения текста программы	10
4.5	Создание исполняемого файла и его работа	10
4.6	Изменения текста программы	11
4.7	Создание исполняемого файла и его работа	11
4.8	Создание файла	12
4.9	Текст программы	12
4.10	Создание исполняемого файла и его работа	12
4.11	Создание исполняемого файла и его работа	13
4.12	Текст программы	13
4.13	Создание исполняемого файла и его работа	14
4.14	Создание файла	14
4.15	Текст программы	15
4.16	Текст программы	16
4.17	Создание исполняемого файла и его работа	16

Список таблиц

1 Цель работы

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

2 Задание

Написание программ с использование циклов и обработкой аргументов командной строки

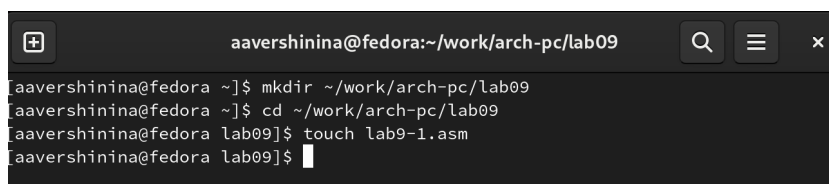
3 Теоретическое введение

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

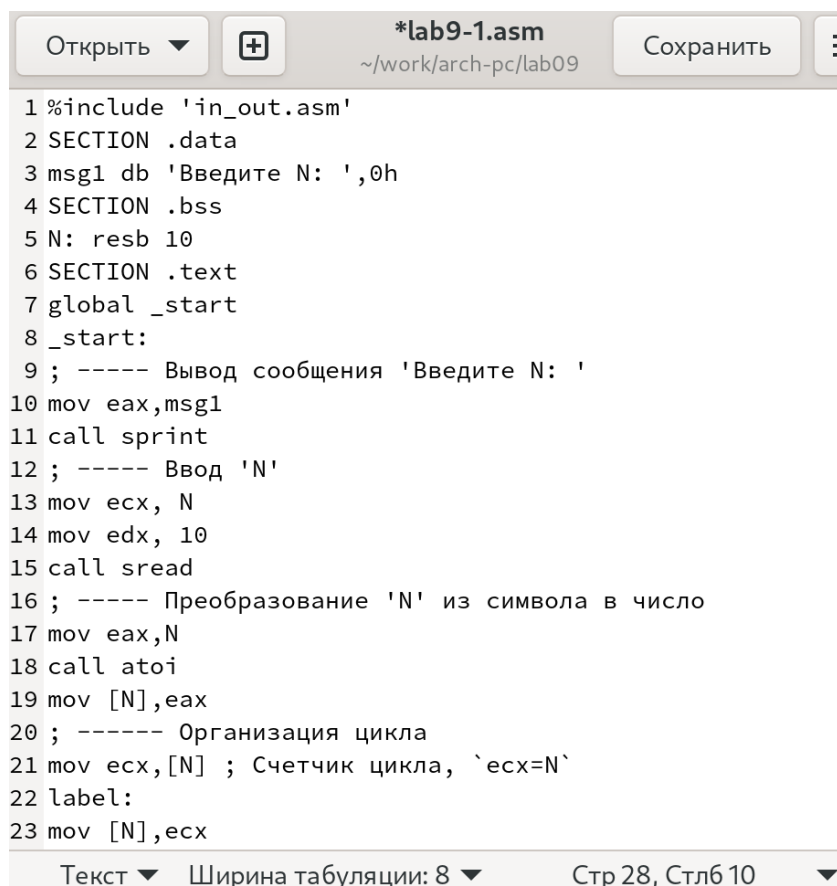
Создам каталог для программ лабораторной работы №9, перейду в него и создам файл lab9-1.asm (рис. 4.1)



```
aavershinina@fedora: ~/work/arch-pc/lab09
aavershinina@fedora ~]$ mkdir ~/work/arch-pc/lab09
aavershinina@fedora ~]$ cd ~/work/arch-pc/lab09
aavershinina@fedora lab09]$ touch lab9-1.asm
aavershinina@fedora lab09]$
```

Рис. 4.1: Создание каталога и файла

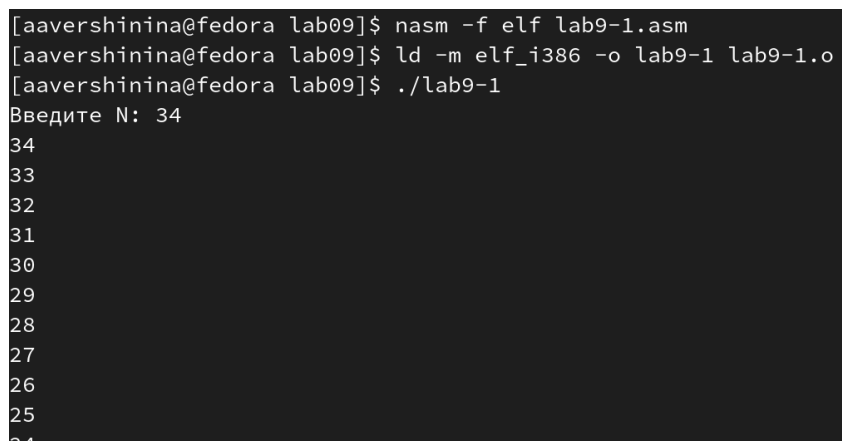
Введу в файл lab9-1.asm текст программы из листинга 9.1. (рис. 4.2) Создам исполняемый файл и проверю его работу. (рис. 4.3) Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы.



```
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
```

Текст ▼ Ширина табуляции: 8 ▼ Стр 28, Стлб 10 ▼

Рис. 4.2: Текст программы



```
[aavershinina@fedora lab09]$ nasm -f elf lab9-1.asm
[aavershinina@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[aavershinina@fedora lab09]$ ./lab9-1
Введите N: 34
34
33
32
31
30
29
28
27
26
25
24
```

Рис. 4.3: Создание исполняемого файла и его работа

Изменяю текст программы добавив изменение значение регистра ecx в цикле (рис. 4.4)

```

label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

```

```

22 label:
23 sub ecx,1 ; `ecx=ecx-1`
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF ; Вывод значения `N`
27 loop label ; `ecx=ecx-1` и если `ecx` не '0'
28 : переход на `label`

```

Рис. 4.4: Изменения текста программы

Создам исполняемый файл и проверьте его работу. (рис. 4.5) Значения принимаются в результате формулы, число подходов цикла не соответствует значению, введенному с клавиатуры, значений через одно нет.

```

[aavershinina@fedora lab09]$ nasm -f elf lab9-1.asm
[aavershinina@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[aavershinina@fedora lab09]$ ./lab9-1
Введите N: 10
9
7
5
3
1
[aavershinina@fedora lab09]$

```

Рис. 4.5: Создание исполняемого файла и его работа

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесу изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop: (рис. 4.6)

```

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

```

```

2 label:
3 sub ecx,1 ; `ecx=ecx-1`
4 mov [N],ecx
5 mov eax,[N]
6 call iprintLF ; Вывод значения `N`
7 pop ecx
8 loop label ; `ecx=ecx-1` и если `ecx` не '0'
9 ; переход на `label`

```

Рис. 4.6: Изменения текста программы

Создам исполняемый файл и проверьте его работу. (рис. 4.7)

```

[aavershinina@fedora lab09]$ nasm -f elf lab9-1.asm
[aavershinina@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[aavershinina@fedora lab09]$ ./lab9-1
Введите N: 10
9
[aavershinina@fedora lab09]$

```

Рис. 4.7: Создание исполняемого файла и его работа

4.2 Обработка аргументов командной строки

Создам файл lab9-2.asm в каталоге ~/work/arch-рс/lab09 (рис. 4.8) и введу в него текст программы из листинга 9.2. (рис. 4.9)

```
[aavershinina@fedora lab09]$ touch lab9-2.asm
[aavershinina@fedora lab09]$
```

Рис. 4.8: Создание файла

```
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. 4.9: Текст программы

Создам исполняемый файл и запущу его, указав аргументы: (рис. 4.10) обработано 3 из 3 аргументов

```
[aavershinina@fedora lab09]$ nasm -f elf lab9-2.asm
[aavershinina@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[aavershinina@fedora lab09]$ ./lab9-2 3 5 '2'
3
5
2
[aavershinina@fedora lab09]$
```

Рис. 4.10: Создание исполняемого файла и его работа

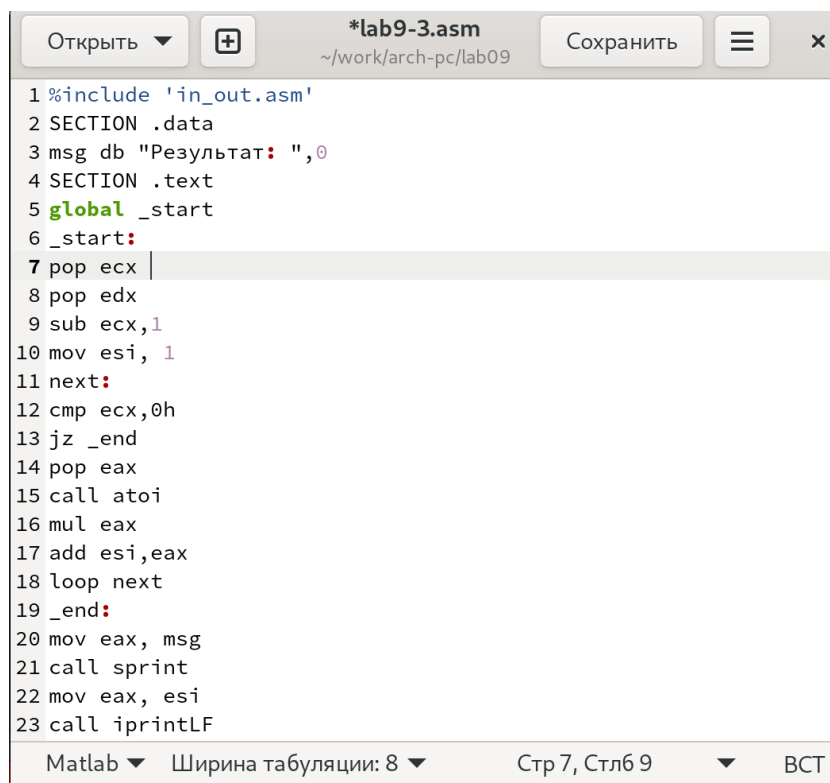
Рассмотрим еще один пример программы которая выводит сумму чисел, кото-

рые передаются в программу как аргументы. Создам файл lab9-3.asm в каталоге ~/work/arch-pc/lab09 и введем в него текст программы из листинга 9.3. Создам исполняемый файл и запущу его, указав аргументы: (рис. 4.11) Выводится сумма, введенных аргументов

```
[aavershinina@fedora lab09]$ touch lab9-3.asm
[aavershinina@fedora lab09]$ gedit lab9-3.asm
[aavershinina@fedora lab09]$ nasm -f elf lab9-3.asm
[aavershinina@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[aavershinina@fedora lab09]$ ./lab9-3 12 34 2 '6'
Результат: 54
[aavershinina@fedora lab09]$
```

Рис. 4.11: Создание исполняемого файла и его работа

Изменяю текст программы из листинга 9.3 для вычисления произведения аргументов командной строки (рис. 4.12)



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

Рис. 4.12: Текст программы

Создам исполняемый файл и запущу его, указав аргументы: (рис. 4.13) Выводится произведение, введенных аргументов

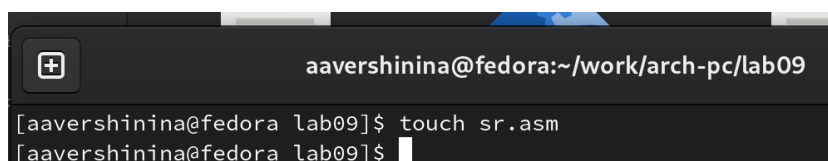
```
[aavershinina@fedora lab09]$ nasm -f elf lab9-3.asm
[aavershinina@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[aavershinina@fedora lab09]$ ./lab9-3 2 3 4
Результат: 24
[aavershinina@fedora lab09]$
```

Рис. 4.13: Создание исполняемого файла и его работа

4.3 Задание для самостоятельной работы

Задание: Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 9.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы No 7.


Создам файл для выполнения задания (рис. 4.14)



```
aavershinina@fedora:~/work/arch-pc/lab09
[aavershinina@fedora lab09]$ touch sr.asm
[aavershinina@fedora lab09]$
```

Рис. 4.14: Создание файла

Введу программу, которая просуммирует значения, введенные с клавиатуры и после рассчитанные по формуле $F(x) = 15x - 9$ (рис. 4.15 и 4.16)

Открыть ▾ 

sr.asm
~/work/arch-pc/lab09

```
%include 'in_out.asm'
SECTION .data
msg1 db "Функция: f(x) = 15x-9 "
msg db "Результат: ",0
SECTION .bss
prm: RESB 80
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 15
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul esi
sub eax,9
add [prm],eax
loop next
_end:
```

Рис. 4.15: Текст программы

```
_end:
mov eax, msg1
mov eax, msg
call sprint
mov eax, [prm]
call iprintLF
call quit
```

Рис. 4.16: Текст программы

Создам исполняемый файл и запущу его, указав аргументы: (рис. 4.17)

```
[aavershinina@fedora lab09]$ nasm -f elf sr.asm
[aavershinina@fedora lab09]$ ld -m elf_i386 -o sr sr.o
[aavershinina@fedora lab09]$ ./sr 2 3 4 5
Результат: 180
[aavershinina@fedora lab09]$ █
```

Рис. 4.17: Создание исполняемого файла и его работа

5 Выводы

В результате выполнения лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы