

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Прозорова Елизавета Евгеньевна

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Выполнение самостоятельной работы	13
4	Выводы	16

1 Цель работы

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

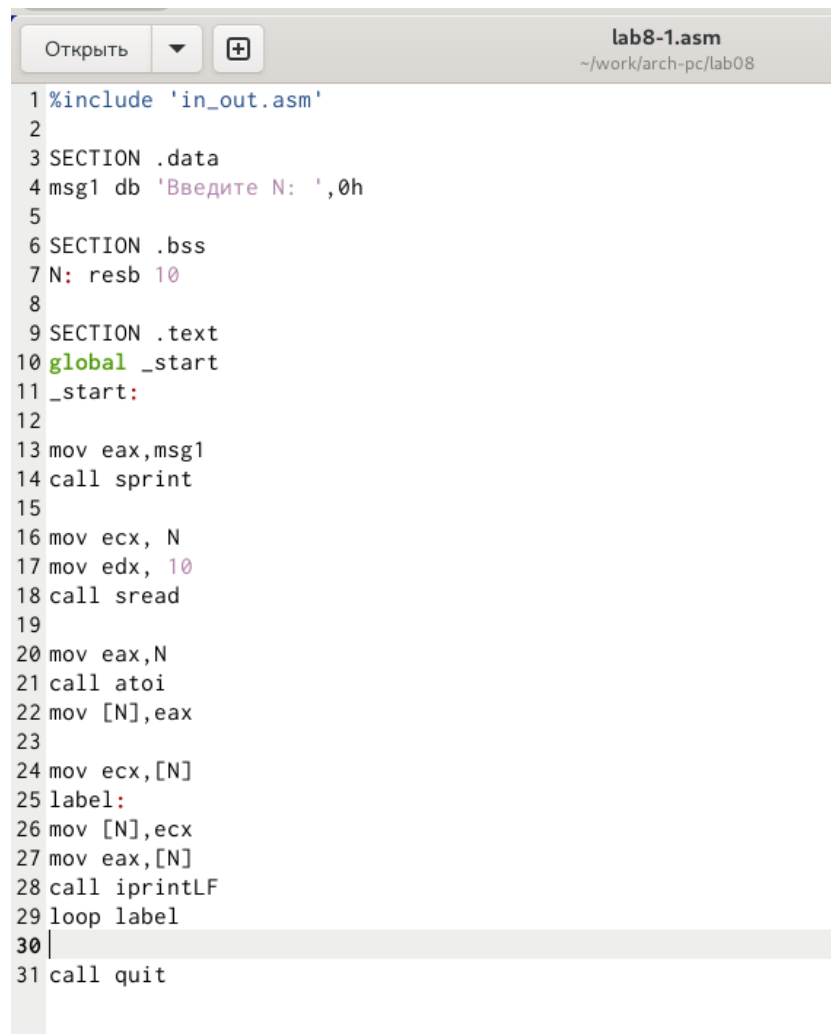
2 Выполнение лабораторной работы

1. Сначала я создала каталог для программ лабораторной работы № 8, затем перешла в него и создала файл lab8-1.asm

```
eeprozorova@dk8n60 ~/.cache/fontconfig $ cd
eeprozorova@dk8n60 ~ $ mkdir ~/work/arch-pc/lab08
eeprozorova@dk8n60 ~ $ cd ~/work/arch-pc/lab08
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ louch lab8-1.asm
bash: louch: команда не найдена
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ touch lab8-1.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 2.1: Создание каталога и файла lab8-1

2. Я ввела в файл lab8-1.asm текст программы из листинга 8.1.



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

Рис. 2.2: Текст программы lab8-1

Я создала исполняемый файл и запустила его. Для N я выбрала число 5.

```

eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ld - elf_386 -o lab8-1 lab8-1.o
ld: невозможно найти -: Нет такого файла или каталога
ld: невозможно найти elf_386: Нет такого файла или каталога
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 1
1
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1

```

Рис. 2.3: Создание и запуск lab8-1

Я изменила программу таким образом, добавив изменение значение регистра есх в цикле.

```

20 mov eax,N
21 call atoi
22 mov [N],eax
23
24 mov ecx,[N]
25 label:
26 sub ecx,1
27 mov [N],ecx
28 mov eax,[N]
29 call iprintLF
30
31 loop label
32
33 call quit

```

Рис. 2.4: Изменения текста

Затем я создала и проверила измененный файл. Цикл закольцевался и стал беконечным.

A terminal window titled 'eeprozorova@dk8n60 - lab08' with search and menu icons. It displays a list of memory addresses: 4294775680, 4294775678, 4294775676, 4294775674, 4294775672, 4294775670, 4294775668, 4294775666, 4294775664, 4294775662, 4294775660, 4294775658, 4294775656, and 4294775654.

```
4294775680
4294775678
4294775676
4294775674
4294775672
4294775670
4294775668
4294775666
4294775664
4294775662
4294775660
4294775658
4294775656
4294775654
```

Рис. 2.5: Создание и запуск lab8-1

Затем я снова изменила текст добавив команды push и pop для сохранения значения счетчика цикла loop

```
20 mov eax,N
21 call atoi
22 mov [N],eax
23
24 mov ecx,[N]
25 label:
26 push ecx
27 sub ecx,1
28 mov [N],ecx
29 mov eax,[N]
30 call iprintLF
31 pop ecx
32
33 loop label
```

Рис. 2.6: Измененный текст программы

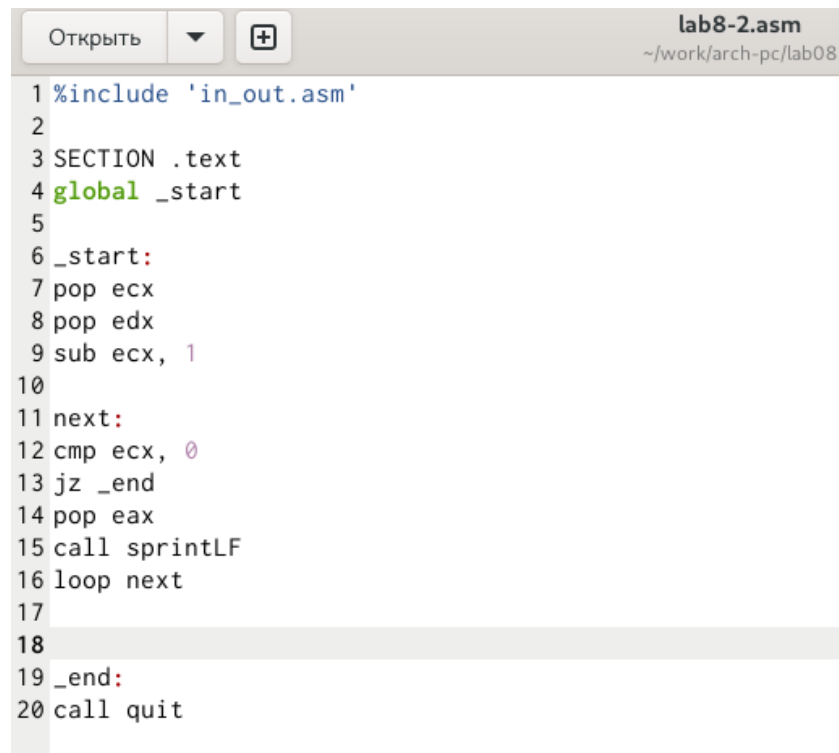
Затем я создала и проверила измененный файл. В этот раз число проходов соответствует N=5.

A terminal window titled 'eeprozorova@dk8n60 - lab08' with search and menu icons. It shows the execution of assembly code. The user runs 'cd ~/work/arch-pc/lab08', 'nasm -f elf lab8-1.asm', 'ld -m elf_i386 -o lab8-1 lab8-1.o', and './lab8-1'. The program prompts 'Введите N: 5' and the user enters '4', '3', '2', '1', and '0' on separate lines.

```
eeprozorova@dk8n60 ~ $ cd ~/work/arch-pc/lab08
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
4
3
2
1
0
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 2.7: Проверка программы

2. Я создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab07 и ввела в него текст программы из листинга 8.2.

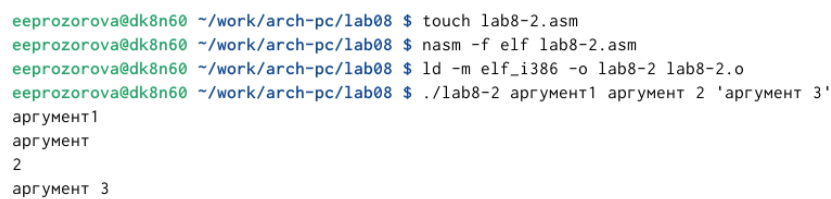


```
lab8-2.asm
~/work/arch-pc/lab08

1 %include 'in_out.asm'
2
3 SECTION .text
4 global _start
5
6 _start:
7 pop ecx
8 pop edx
9 sub ecx, 1
10
11 next:
12 cmp ecx, 0
13 jz _end
14 pop eax
15 call sprintLF
16 loop next
17
18
19 _end:
20 call quit
```

Рис. 2.8: Текст программы lab8-2.asm

Затем я создала и проверила работу файла, указав аргументы так: аргумент1 аргумент 2 'аргумент 3'



```
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ touch lab8-2.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рис. 2.9: Проверка работы программы

Все 3 аргумента были обработаны по разному.

Я создала файл lab8-3.asm в каталоге ~/work/arch-pc/lab07 и ввела в него текст программы из листинга 8.3.


```

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 pop edx
13 sub ecx,1
14 mov esi,0
15
16 next:
17 cmp ecx,0h
18 jz _end
19
20 pop eax
21 call atoi
22 add esi,eax
23 loop next
24
25 _end:
26 mov eax, msg
27 call sprint
28 mov eax, esi
29 call iprintLF
30
31 call quit

```

Рис. 2.10: Текст программы lab8-3.asm

Затем я создала и проверила файл. Я указала приведенные в пример аргументы 12 13 7 01 5 и выбрала свои 47 9 23 11

```
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-3 47 9 23 11
Результат: 90
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 2.11: Проверка программы

Затем я изменила текст программы для вычисления произведения аргументов командной строки.

```

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 pop edx
13 sub ecx,1
14 mov esi,1
15 mov eax,1
16
17 next:
18 cmp ecx,0h
19 jz _end
20
21 pop eax
22 call atoi
23 mov ebx,eax
24 mov eax,esi
25 mul ebx
26 mov esi,eax
27 loop next
28
29 _end:
30 mov eax, msg
31 call sprint
32 mov eax, esi
33 call iprintLF
34
35 call quit

```

Рис. 2.12: Текст файла

Я создала и проверила измененный файл. Для проверки я выбрала следующие группы чисел: 1 2 3 4 и 8 10 11.

```
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4
Результат: 24
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-3 8 10 11
Результат: 880
```

Рис. 2.13: Проверка программы

3 Выполнение самостоятельной работы

Я создала файл var, в котором написала программу которая находит сумму значений функции $f(x)$ для каких-то x . Мой вариант - 8, поэтому мой вид функции $f(x)=7+2x$.

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db "Функция: f(x)=7+2x",0
5 msg2 db "Результат: ",0
6
7 SECTION .text
8 global _start
9
10 _start:
11
12 pop ecx
13 pop edx
14 sub ecx,1
15 mov esi,0
16 mov eax,msg1
17 call sprintf
18
19 next:
20 cmp ecx,0h
21 jz _end
22
23 mov ebx,2
24 pop eax
25 call atoi
26 mul ebx
27 add eax,7
28 add esi,eax
29 loop next
30
31 _end:
32
33 mov eax, msg2
34 call sprintf
35 mov eax, esi
36 call iprintLF

```

Рис. 3.1: Текст моей программы

Теперь проверим как она работает. Для проверки я выбрала следующие значения: 1 2 и 3 7.

```
eeprozorova@dk8n60 - lab08
eeprozorova@dk8n60 ~ $ cd ~/work/arch-pc/lab08
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf var.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o var var.o
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./var 1 2
Функция:  $f(x)=7+2x$ 
Результат: 20
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $ ./var 3 7
Функция:  $f(x)=7+2x$ 
Результат: 34
eeprozorova@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 3.2: Проверка работы программы

4 Выводы

В ходе лабораторной работы мной были изучены навыки написания программ с использованием циклов и обработкой аргументов командной строки