

Отчет по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Полина Алексеевна Ларионова

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задания для самостоятельной работы	11
4	Выводы	13

Список иллюстраций

2.1	lab08	5
2.2	Листинг 8.1	5
2.3	lab8-1.o	6
2.4	Измененный текст	6
2.5	Исполняемый файл	7
2.6	Второе изменение	7
2.7	Работа программы	8
2.8	Листинг 8.2	8
2.9	lab8-2.o	9
2.10	Листинг 8.3	9
2.11	lab8-3.o	9
2.12	Измененная программа	10
2.13	Исполняемый файл	10
3.1	Вариант 10	11
3.2	Variant-10.o	12

1 Цель работы

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

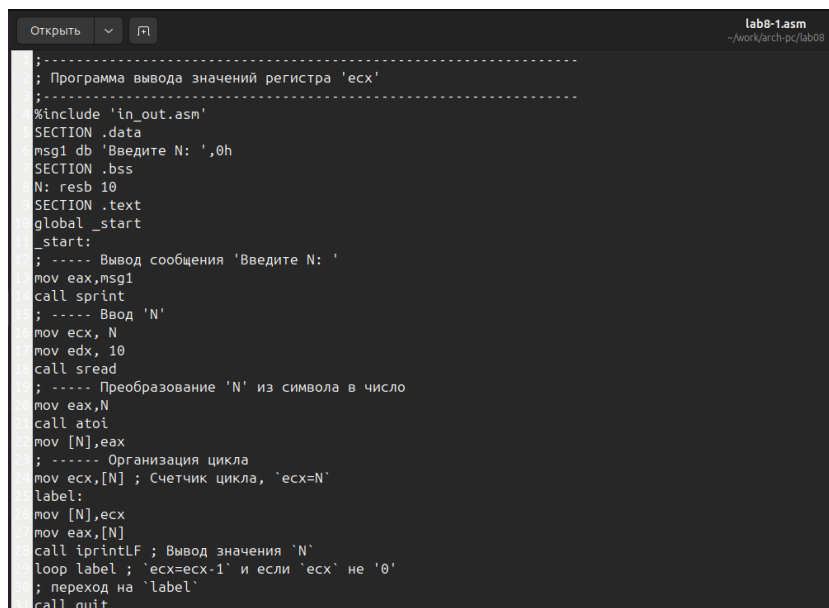
2 Выполнение лабораторной работы

Я создала каталог с файлом для выполнения лабораторной работа №8.

```
palarionova@linux:~$ mkdir ~/work/arch-pc/lab08
palarionova@linux:~$ cd ~/work/arch-pc/lab08
palarionova@linux:~/work/arch-pc/lab08$ touch lab8-1.asm
palarionova@linux:~/work/arch-pc/lab08$
```

Рис. 2.1: lab08

Я внесла в файл текст программы из листинга 8.1ю



```
lab8-1.asm
~/work/arch-pc/lab08

;-----
; Программа вывода значений регистра 'ecx'
;-----
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
1; ----- Вывод сообщения 'Введите N: '
2mov eax,msg1
3call sprint
4; ----- Ввод 'N'
5mov ecx, N
6mov edx, 10
7call sread
8; ----- Преобразование 'N' из символа в число
9mov eax,N
10call atoi
11mov [N],eax
12; ----- Организация цикла
13mov ecx,[N] ; Счетчик цикла, 'ecx=N'
14label:
15mov [N],ecx
16mov eax,[N]
17call iprintf ; Вывод значения 'N'
18loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
19; переход на 'label'
20call quit
```

Рис. 2.2: Листинг 8.1

Далее я создала исполняемый файл, чтобы проверить работу программы.

```

palarionova@linux:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
palarionova@linux:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
palarionova@linux:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 3
3
2
1
palarionova@linux:~/work/arch-pc/lab08$

```

Рис. 2.3: lab8-1.o

Затем я внесла изменения в программу и создала исполняемый файл.

```

lab8-1.asm
~/work/arch-pc/lab08
Открыть Сохранить
1;-----
2; Программа вывода значений регистра 'ecx'
3;-----
4#include 'in_out.asm'
5SECTION .data
6msg1 db 'Введите N: ',0h
7SECTION .bss
8N: resb 10
9SECTION .text
10global _start
11_start:
12; ----- Вывод сообщения 'Введите N: '
13mov eax,msg1
14call sprint
15; ----- Ввод 'N'
16mov ecx, N
17mov edx, 10
18call sread
19; ----- Преобразование 'N' из символа в число
20mov eax,N
21call atoi
22mov [N],eax
23; ----- Организация цикла
24mov ecx,[N] ; Счетчик цикла, 'ecx=N'
25label:
26push ecx ; добавление значения ecx в стек
27sub ecx,1
28mov [N],ecx
29mov eax,[N]
30call iprintLF
31pop ecx ; извлечение значения ecx из стека
32loop label
33; переход на 'label'
34call quit
35

```

Рис. 2.4: Измененный текст

```

palarionova@linux:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
palarionova@linux:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
palarionova@linux:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
palarionova@linux:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
palarionova@linux:~/work/arch-pc/lab08$ 

```

Рис. 2.5: Исполняемый файл

Я снова изменила текст программы и создала исполняемый файл.

```

Открыть  ▾  *lab8-2.asm  Сохранить  ≡
~/work/arch-pc/lab08

1 ;-----
2 ; Обработка аргументов командной строки
3 ;-----
4 %include 'in_out.asm'
5 SECTION .text
6 global _start
7 _start:
8     pop ecx ; Извлекаем из стека в `ecx` количество
9     ; аргументов (первое значение в стеке)
10    pop edx ; Извлекаем из стека в `edx` имя программы
11    ; (второе значение в стеке)
12    sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
13    ; аргументов без названия программы)
14    next:
15    cmp ecx, 0 ; проверяем, есть ли еще аргументы
16    jz _end ; если аргументов нет выходим из цикла
17    ; (переход на метку `_end`)
18    pop eax ; иначе извлекаем аргумент из стека
19    call sprintf ; вызываем функцию печати
20    loop next ; переход к обработке следующего
21    ; аргумента (переход на метку `next`)
22    _end:
23    call quit
24

```

Рис. 2.6: Второе изменение

```

palarionova@linux:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
palarionova@linux:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
ld: не распознан режим эмуляции: elf_i386
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu i386pep i386pe
palarionova@linux:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
palarionova@linux:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
palarionova@linux:~/work/arch-pc/lab08$

```

Рис. 2.7: Работа программы

Я создала файл lab8-2.asm и внесла в него текст из листинга 8.2.

```

Открыть  lab8-3.asm  Сохранить
~/work/arch-pc/lab08

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
    ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
    ; (второе значение в стеке)
    sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
    ; аргументов без названия программы)
    mov esi,0 ; Используем `esi` для хранения
    ; промежуточных сумм
next:
    cmp ecx,0h ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
    ; (переход на метку `_end`)
    pop eax ; иначе извлекаем следующий аргумент из стека
    call atoi ; преобразуем символ в число
    add esi,eax ; добавляем к промежуточной сумме
    ; след. аргумент `esi=esi+eax`
    loop next ; переход к обработке следующего аргумента
_end:
    mov eax,msg ; вывод сообщения "Результат: "
    call sprint
    mov eax,esi ; записываем сумму в регистр `eax`
    call iprintLF ; печать результата
    call quit ; завершение программы

```

Рис. 2.8: Листинг 8.2

Проверила его работу, задав аргументы.


```

palarionova@linux:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
palarionova@linux:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
palarionova@linux:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
palarionova@linux:~/work/arch-pc/lab08$

```

Рис. 2.9: lab8-2.o

Далее я создала файл lab8-3.asm и ввела в него текст программы из листинга 8.3.

```

lab8-3.asm
~/work/arch-pc/lab08
Сохранить

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    mov ebx,eax;
22    mov eax,esi
23    mul ebx
24    mov esi,eax
25    loop next ; переход к обработке следующего аргумента
26    _end:
27    mov eax, msg ; вывод сообщения "Результат: "
28    call sprint
29    mov eax, esi ; записываем сумму в регистр `eax`
30    call iprintLF ; печать результата
31    call quit ; завершение программы
32

```

Рис. 2.10: Листинг 8.3

Я проверила его работу, указав аргументы.

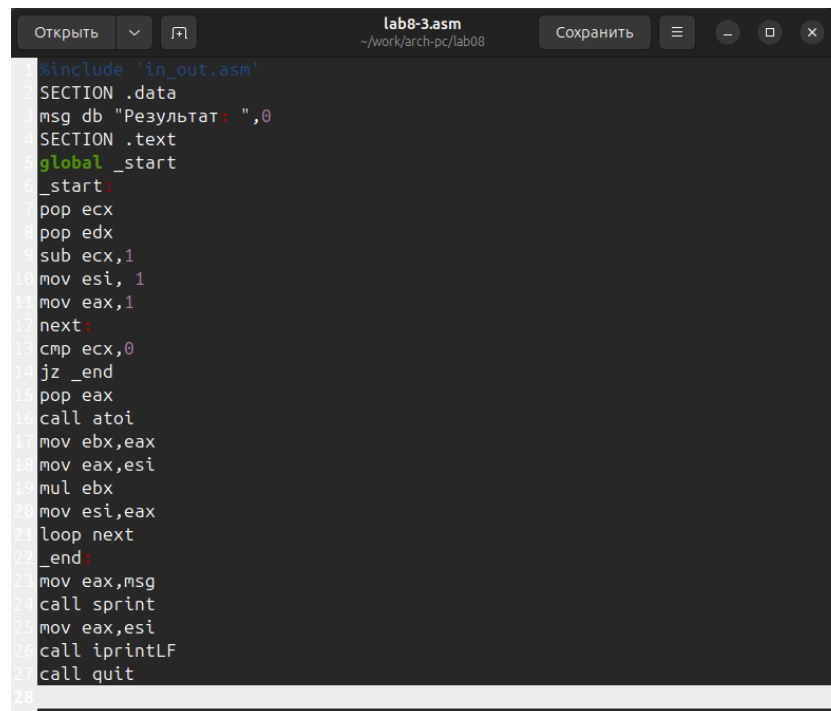
```

palarionova@linux:~/work/arch-pc/lab08$ ld -m elf_i386 -o main lab8-3.o
palarionova@linux:~/work/arch-pc/lab08$ ./main 12 13 7 10 5
Результат: 47
palarionova@linux:~/work/arch-pc/lab08$

```

Рис. 2.11: lab8-3.o

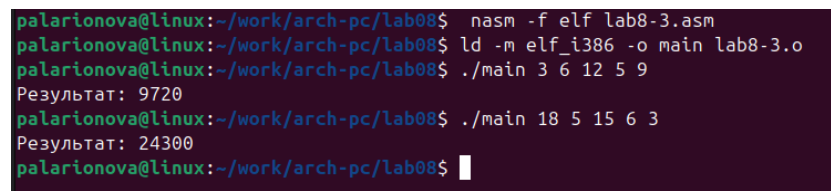
Для вычисления произведения аргументов я изменила текст программы.



```
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 mov eax,1
12 next:
13 cmp ecx,0
14 jz _end
15 pop eax
16 call atoi
17 mov ebx,eax
18 mov eax,esi
19 mul ebx
20 mov esi,eax
21 loop next
22 _end:
23 mov eax,msg
24 call sprint
25 mov eax,esi
26 call iprintLF
27 call quit
28
```

Рис. 2.12: Измененная программа

Затем проверила его работу.

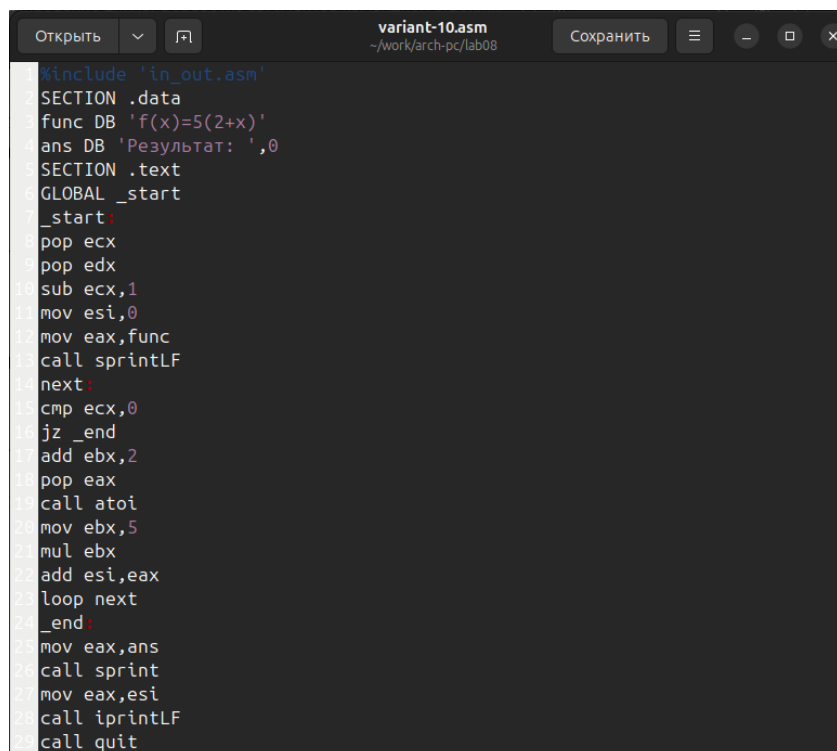


```
palarionova@linux:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
palarionova@linux:~/work/arch-pc/lab08$ ld -m elf_i386 -o main lab8-3.o
palarionova@linux:~/work/arch-pc/lab08$ ./main 3 6 12 5 9
Результат: 9720
palarionova@linux:~/work/arch-pc/lab08$ ./main 18 5 15 6 3
Результат: 24300
palarionova@linux:~/work/arch-pc/lab08$
```

Рис. 2.13: Исполняемый файл

3 Задания для самостоятельной работы

Я написала текст программы, которая находит сумму значений функции в соответствии с 10 вариантом.



```
variant-10.asm
~/work/arch-pc/lab08

1 %include 'in_out.asm'
2 SECTION .data
3 func DB 'f(x)=5(2+x)'
4 ans DB 'Результат: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 pop ecx
9 pop edx
10 sub ecx,1
11 mov esi,0
12 mov eax,func
13 call sprintf
14 next:
15 cmp ecx,0
16 jz _end
17 add ebx,2
18 pop eax
19 call atoi
20 mov ebx,5
21 mul ebx
22 add esi,eax
23 loop next
24 _end:
25 mov eax,ans
26 call sprintf
27 mov eax,esi
28 call iprintLF
29 call quit
```

Рис. 3.1: Вариант 10

Далее я создала исполняемый файл и ввела несколько аргументов.

```
palarionova@linux:~/work/arch-pc/lab08$ nasm -f elf variant-10.asm
palarionova@linux:~/work/arch-pc/lab08$ ld -m elf_i386 -o variant-10 variant-10.o
palarionova@linux:~/work/arch-pc/lab08$ ./variant-10 4 3 2 1
f(x)=5(2+x)Результат:
Результат: 50
palarionova@linux:~/work/arch-pc/lab08$ ./variant-10 5 2 3 1
f(x)=5(2+x)Результат:
Результат: 55
palarionova@linux:~/work/arch-pc/lab08$ ./variant-10 5 8 4 2 1
f(x)=5(2+x)Результат:
Результат: 100
palarionova@linux:~/work/arch-pc/lab08$
```

Рис. 3.2: Variant-10.o

4 Выводы

Я научилась программировать цикл и обрабатывать аргументы командной строки при выполнении данной лабораторной работы.