

Программирование цикла. Обработка аргументов командной строки

Лабораторная работа №8

Приходько Иван Иванович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение лабораторной работы	16
4	Выводы	19

Список иллюстраций

2.1	Создание файла lab8-1.asm	6
2.2	Вставка кода из листинга 8.1	7
2.3	Копирование in_out.asm	8
2.4	Запуск lab8-1.asm	8
2.5	Редактирование lab8-1.asm	8
2.6	Повторный запуск lab8-1.asm	9
2.7	Повторный запуск lab8-1.asm	9
2.8	Редактирование файла lab8-1.asm	10
2.9	Повторный запуск lab8-1.asm	10
2.10	Создание lab8-2.asm	10
2.11	Редактирование lab8-2.asm	11
2.12	Запуск lab8-2.asm	11
2.13	Создание lab8-3.asm	11
2.14	Редактирование lab8-3.asm	12
2.15	Запуск lab8-3.asm	13
2.16	Редактирование lab8-3.asm	14
2.17	Повторный запуск lab8-3.asm	15
3.1	Запуск task1v6.asm	16
3.2	Код файла самостоятельной работы	17
3.3	Сборка и запуск task1v6.asm	18

Список таблиц

1 Цель работы

Научиться работать с циклами на языке Ассемблера, а также научиться обрабатывать аргументы командной строки

2 Выполнение лабораторной работы

Для начала выполнения лабораторной работы перейдем рабочую директорию и создадим файл lab8-1.asm (рис. 2.1).

```
ivanprithodko@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/  
ivanprithodko@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ touch lab8-1.asm  
ivanprithodko@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 2.1: Создание файла lab8-1.asm

Теперь вставим код из листинга 8.1. Он должен запускать цикл и выводить каждую итерацию число, на единицу меньше предыдущего (рис. 2.2).

```
GNU nano 7.2 /home
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2.2: Вставка кода из листинга 8.1

Скопируем файл in_out.asm из предыдущей работы (рис. 2.3).

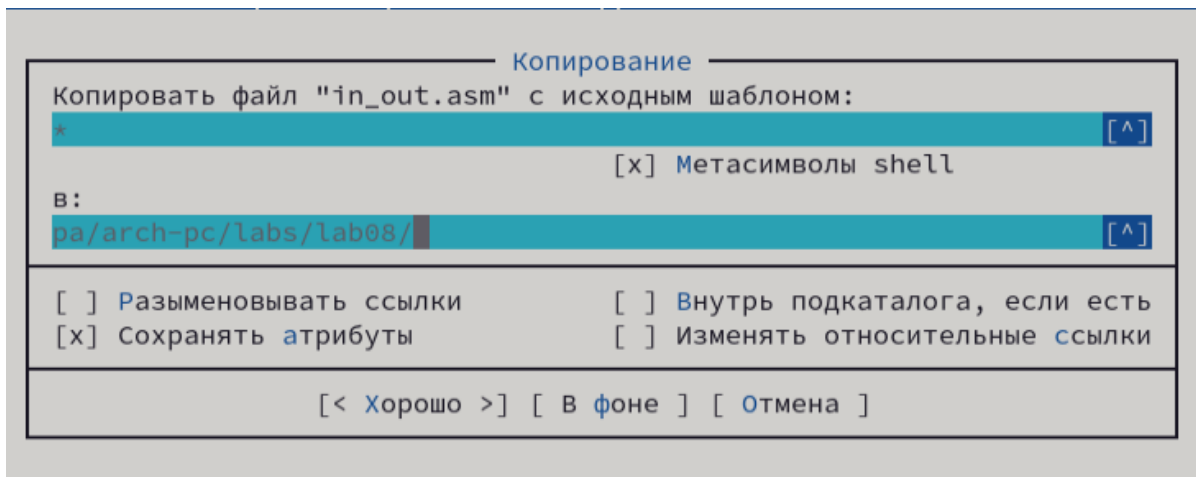


Рис. 2.3: Копирование in_out.asm

Теперь создадим и запустим файл (рис. 2.4).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-1.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-1
Введите N: 6
6
5
4
3
2
1
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 2.4: Запуск lab8-1.asm

Теперь попробуем изменить код, чтобы в цикле также отнималась единица у регистра ecx (рис. 2.5).

```
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не
```

Рис. 2.5: Редактирование lab8-1.asm

Теперь соберем и запустим файл (рис. 2.6).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-1.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-1
Введите N: 8
7
5
3
1
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 2.6: Повторный запуск lab8-1.asm

```
4294959166
4294959164
4294959162
4294959160
4294959158
4294959156
4294959154
4294959152
4294959150
4294959148
4294959146
4294959144
4294959142
4294959140
4294959138
4294959136
4294959134
4294959132
4294959130
4294959128
4294959126
4294959124
4294959122^C
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 2.7: Повторный запуск lab8-1.asm

Как видим, цикл выполняется бесконечное количество раз. Это связано с тем, что цикл останавливается в тот момент, когда при проверке `ecx` равен 0, но он каждое выполнение цикла уменьшается на 2, из-за чего, в случае нечётного числа, никогда не достигнет нуля. Если на вход подать чётное число, цикл прогонится $N/2$ раз, выводя числа от $N-1$ до 1.

Теперь попробуем изменить программу так, чтобы она сохраняла значение регистра `ecx` в стек (рис. 2.8).

```

mov ecx,[N] ; C4
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ;
pop ecx
loop label ; `ec

```

Рис. 2.8: Редактирование файла lab8-1.asm

Теперь соберем и запустим файл (рис. 2.9).

```

ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-1.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-1
Введите N: 7
6
5
4
3
2
1
0
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$

```

Рис. 2.9: Повторный запуск lab8-1.asm

Теперь, программа выводит все числа от N-1 до нуля. Таким образом, число прогонов цикла равно числу N. Создадим второй файл (рис. 2.10).

```

ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ touch lab8-2.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$

```

Рис. 2.10: Создание lab8-2.asm

Вставим код из листинга 8.2 (рис. 2.11).

```
GNU nano 7.2 /home/ivanpri
%include 'in_out.asm'
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
por eax ; иначе извлекаем аргумент из стека
call sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 2.11: Редактирование lab8-2.asm

Теперь соберем и запустим файл (рис. 2.12).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-2.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

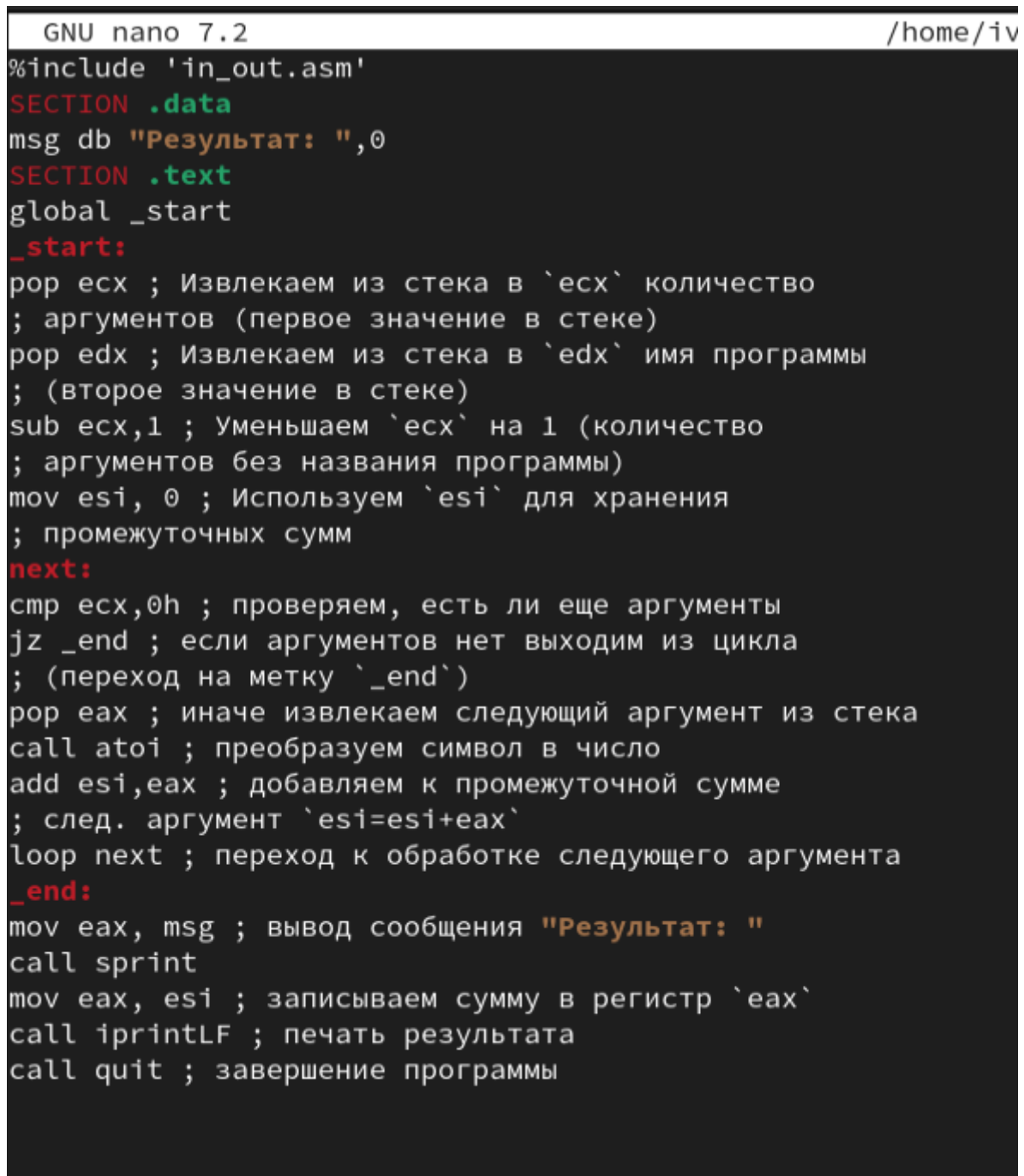
Рис. 2.12: Запуск lab8-2.asm

Теперь создадим lab8-3.asm (рис. 2.13).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ touch lab8-3.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 2.13: Создание lab8-3.asm

Вставим в него код из листинга 8.3 (рис. 2.14).



```
GNU nano 7.2 /home/iv
%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. 2.14: Редактирование lab8-3.asm

Соберем и запустим файл (рис. 2.15).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-3.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 2.15: Запуск lab8-3.asm

Изменим файл так, чтобы она находила не сумму, а произведение всех аргументов (рис. 2.16).

```
GNU nano 7.2 /home/i'
%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, 1 ; 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
mul esi
mov esi,eax
loop next ; переход к обработке следующего аргумента
_end:
mov ebx,eax
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, ebx ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис. 2.16: Редактирование lab8-3.asm

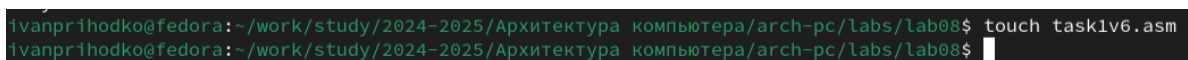
Соберем и запустим файл (рис. 2.17).

```
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-3.asm
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
ivanprikhodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 2.17: Повторный запуск lab8-3.asm

3 Выполнение лабораторной работы

Создадим файл для лабораторной работы (рис. 3.1).



```
ivanpr1hodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ touch task1v6.asm
ivanpr1hodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 3.1: Запуск task1v6.asm

В рамках самостоятельной работы необходимо сделать задание под вариантом 6. Необходимо сложить результаты выполнения функции $f(x)=4x-3$ для всех введённых аргументов (рис. 3.2).


```
GNU nano 7.2 /home/ivanpriho
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg2 db "Функция: f(x)=4x-3"
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
mov ebx, 4
mul ebx
sub eax, 3
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg2
call sprintLF
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. 3.2: Код файла самостоятельной работы

Соберем и запустим файл (рис. 3.3).

```
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf task1v6.asm
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o task1v6 task1v6.o
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./task1v6 1
Функция:  $f(x)=4x-3$ 
Результат: 1
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./task1v6 1 2
Функция:  $f(x)=4x-3$ 
Результат: 6
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./task1v6 1 1
Функция:  $f(x)=4x-3$ 
Результат: 2
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./task1v6 2 2
Функция:  $f(x)=4x-3$ 
Результат: 10
ivanprihodko@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$
```

Рис. 3.3: Сборка и запуск task1v6.asm

4 Выводы

В результате выполнения лабораторной работы были получены навыки работы с циклами и обработкой аргументов из командной строки. Были написаны программы, использующие все вышеописанные аспекты.