

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки

Колонтырский Илья Русланович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	17

Список иллюстраций

2.1	Создание папки и файла lab8-1.asm	6
2.2	Вставка кода в файл	7
2.3	Компиляция и запуск программы	8
2.4	Изменение кода в файле lab8-1.asm	8
2.5	Компиляция и запуск программы	9
2.6	Изменение программы lab8-1.asm	10
2.7	Компиляция и запуск программы	11
2.8	Создание файла lab8-2.asm	11
2.9	Вставка кода в файл	12
2.10	Компиляция и запуск	12
2.11	Создание файла lab8-3.asm	12
2.12	Вставка кода в файл	13
2.13	Компиляция и запуск	13
2.14	Изменение файла	14
2.15	Компиляция и запуск	14
2.16	Создание файла	15
2.17	Код для самостоятельной работы	15
2.18	Компиляция и запуск программы	16

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Создадим рабочую папку и файл lab8-1.asm (рис. 2.1)

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

Рис. 2.1: Создание папки и файла lab8-1.asm

Вставим в файл предложенный код (рис. 2.2)

```
GNU nano 6.2 /home/irkolontyrskiy/work/arch-pc/lab08/lab8-1.asm
#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 в файл

Скомпилируем файл и посмотрим результат (рис. 2.3)

```

irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1

```

Рис. 2.3: Компиляция и запуск программы

Изменим код программы следующим образом (рис. 2.4)

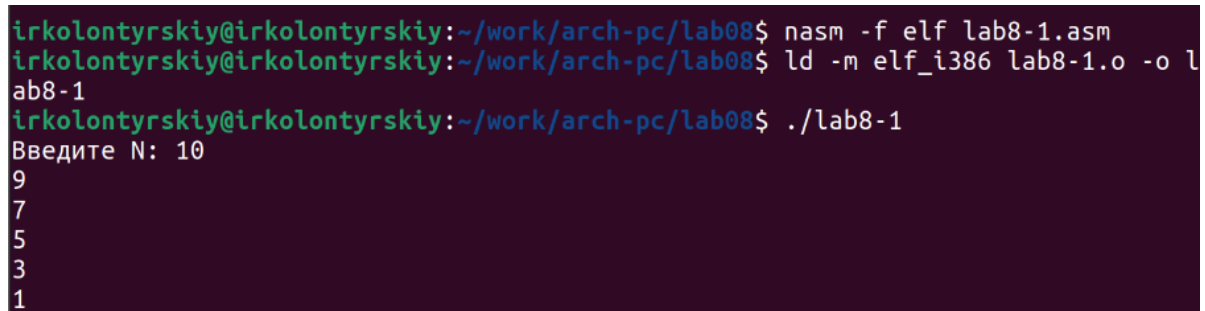
```

GNU nano 6.2 /home/irkolontyrskiy/work/arch-pc/lab08/lab8-1.asm *
#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:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit

```

Рис. 2.4: Изменение кода в файле lab8-1.asm

Скомпилируем и запустим программу. Подадим на вход число 10 и посмотрим на результат (рис. 2.5)



```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
```

Рис. 2.5: Компиляция и запуск программы

Как мы видим, в цикле значение регистра `ecx` меняется 2 раза, поэтому нам выводятся числа через один. Соответственно, число проходов цикла значению `N` не соответствует.

Теперь изменим программу для её правильной работы (рис. 2.6)

```

GNU nano 6.2 /home/irkolontyrskiy/work/arch-pc/lab08/lab8-1.asm *
#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:
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
call quit

```

Рис. 2.6: Изменение программы lab8-1.asm

Скомпилируем и запустим программу (рис. 2.7)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

Рис. 2.7: Компиляция и запуск программы

Сейчас количество прогонов соответствует числу N. Теперь создадим второй файл (рис. 2.8)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ touch lab8-2.asm
```

Рис. 2.8: Создание файла lab8-2.asm

Вставим в него следующий код (рис. 2.9)

```
GNU nano 6.2 /home/irkolontyrskiy/work/arch-pc/lab08/lab8-2.asm *
#include 'in_out.asm'
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)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop 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. 2.9: Вставка кода в файл

Скомпилируем его и запустим, указав предложенные аргументы (рис. 2.10)

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

Рис. 2.10: Компиляция и запуск

Программа обработала 4 аргумента, разделённых пробелами. Пробелы в кавычках не считаются разделителями. Создадим третий файл (рис. 2.11)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ touch lab8-3.asm
```

Рис. 2.11: Создание файла lab8-3.asm

Вставим в него код для нахождения суммы аргументов (рис. 2.12)

```

GNU nano 6.2 /home/irkolontyrskiy/work/arch-pc/lab08/lab8-3.asm *
#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. 2.12: Вставка кода в файл

Скомпилируем код и запустим его (рис. 2.13)

```

irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47

```

Рис. 2.13: Компиляция и запуск

Программа работает правильно. Сделаем так, чтобы она искала не сумму, а произведение (рис. 2.14):

```

GNU nano 6.2 /home/irkolontyrskiy/work/arch-pc/lab08/lab8-3.asm
#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 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: Изменение файла

Скомпилируем и запустим программу (рис. 2.15)

```

irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600

```

Рис. 2.15: Компиляция и запуск

Ответ верный

Самостоятельная работа

Создадим файл программы для самостоятельной работы (рис. 2.16)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ touch v19.asm
```

Рис. 2.16: Создание файла

Напишем программу для 19 варианта, где нужно сложить все $f(x)=8x-3$ (рис. 2.17)

```
GNU nano 6.2 /home/irkolontyrskiy/work/arch-pc/lab08/v19.asm *
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg2 db "Функция: f(x)=8x-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, 8
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. 2.17: Код для самостоятельной работы

Скомпилируем и проверим работу программы (рис. 2.18)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ nasm -f elf v19.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ld -m elf_i386 v19.o -o v19
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ./v19 1 2 3 4
Функция:  $f(x)=8x-3$ 
Результат: 68
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab08$ ./v19 1 3 5 7
Функция:  $f(x)=8x-3$ 
Результат: 116
```

Рис. 2.18: Компиляция и запуск программы

Программа работает верно

3 Выводы

Были получены навыки работы с циклами, а также с аргументами командной строки