

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Краснопер Данила Олегович

Содержание

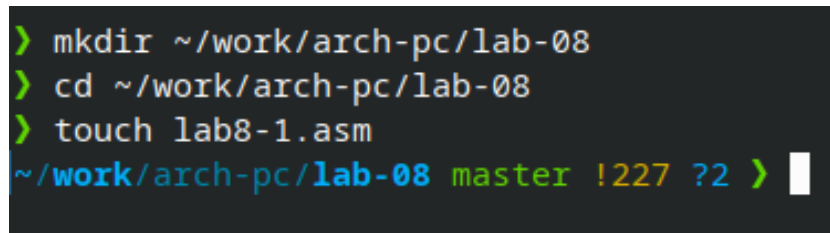
1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Самостоятельная работа	13
4	Вывод	16

1 Цель работы

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

2 Выполнение лабораторной работы

1) Я создала каталог lab8 и файл lab8-1.asm

A terminal window with a dark background and light-colored text. It shows three commands being executed: 'mkdir ~/work/arch-pc/lab-08', 'cd ~/work/arch-pc/lab-08', and 'touch lab8-1.asm'. The prompt for the first command is a green '>'. The prompt for the second command is a green '>'. The prompt for the third command is a green '>'. The output of the third command is a blue prompt '~/' followed by the path 'work/arch-pc/lab-08' in blue, the word 'master' in green, and a red '!227 ?2' followed by a green '>' and a white cursor bar.

```
> mkdir ~/work/arch-pc/lab-08
> cd ~/work/arch-pc/lab-08
> touch lab8-1.asm
~/work/arch-pc/lab-08 master !227 ?2 > |
```

Рис. 2.1: Создание файла и каталога

2) В файл я ввела текст первой программы и создала исполняемый файл.

```

#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
loop label
call quit

```

Рис. 2.2: Текст программы

```
> nasm -f elf lab8-1.asm
> ld -m elf_i386 -o lab8-1 lab8-1.o
> ./lab8-1
Введите N: 5
5
4
3
2
1
~/work/a/lab-08 master !227 ?2 >
```

Рис. 2.3: Запуск программы и проверка результата

3) Я изменила текст программы, в теле цикла `label` добавила строку `sub eax,1`. Цикл закольцевался и стал бесконечным.

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

Рис. 2.4: Измененный текст программы

294578188
294578186
294578184
294578182
294578180
294578178
294578176
294578174
294578172
294578170
294578168
294578166
294578164
294578162
294578160

Рис. 2.5: Запуск программы

4)Я изменила текст программы так, чтобы цикл и счетчик работал правильно. По итогу после изменения программы, число проходимости циклов стало соответствовать числу введенному с клавиатуры.

```
%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: Редактирование текста программы


```

> nasm -f elf lab8-1.asm
> ld -m elf_i386 -o lab8-1 lab8-1.o
> ./lab8-1
Введите N: 5
4
3
2
1
0
~/work/a/lab-08 master !227 ?2 >

```

Рис. 2.7: Запуск измененной программы

5) Я создала файл lab8-2.asm и ввела туда программу, которая выводит все аргументы, которые ввели. Программа выводит все 3 аргумента которые ввели, но в разной вариации.

```

%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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку `next`)
_end:
    call quit

```

Рис. 2.8: Текст программы для вывода аргументов

```
> nasm -f elf lab8-2.asm
> ld -m elf_i386 -o lab8-2 lab8-2.o
> ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
~/work/arch-pc/lab-08 master !227 ?2 > |
```

Рис. 2.9: Результаты работы программы

- 6) Я создала файл lab8-3.asm. Ввела текст программы и запустила ее.
Программа вывела сумму чисел, которые я ввела.

```

#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

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. 2.10: Текст программы lab9-3

```

> nasm -f elf lab8-3.asm
> ld -m elf_i386 -o lab8-3 lab8-3.o
> ./lab8-3 12 13 7 10 5
Результат: 47
~/work/arch-pc/lab-08 master !227 ?2 >

```

Рис. 2.11: Результат работы программы

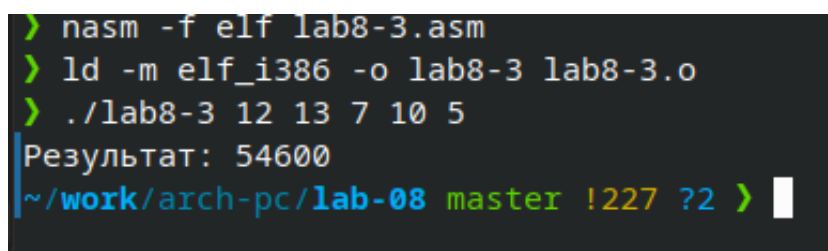
- 7) Я изменила программу, чтобы она выводила произведение введенных чисел.

```
%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. 2.12: Текст программы с произведением чисел



```
> nasm -f elf lab8-3.asm
> ld -m elf_i386 -o lab8-3 lab8-3.o
> ./lab8-3 12 13 7 10 5
Результат: 54600
~/work/arch-pc/lab-08 master !227 ?2 > |
```

Рис. 2.13: Результаты работы программы с произведением

3 Самостотельная работа

Я написала программу, которая выводит сумму всех решений примера. В лабораторной работе №6, я получила 3 вариант, поэтому я писала программу для 3 варианта. Введенные числа я придумала сам, и посчитала их, чтобы проверить работу программы.

```

#include 'in_out.asm'

SECTION .data
msg_func db "Функция:  $f(x) = 10x - 5$ ", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprintLF
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
mov ebx, 10
mul ebx
sub eax, 5
add esi, eax
loop next
_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit

```

Рис. 3.1: Текст программы в самостоятельной работе

```
> touch lab8-4.asm
> nasm -f elf lab8-4.asm
> ld -m elf_i386 -o lab8-4 lab8-4.o
> ./lab8-4 1 2 3 4
Функция:  $f(x) = 10x - 5$ 
Результат: 80
> ./lab8-4 0 0 0 0
Функция:  $f(x) = 10x - 5$ 
Результат: 4294967276
> ./lab8-4 1 2 0 9 0 9 8
Функция:  $f(x) = 10x - 5$ 
Результат: 255
~/work/arch-pc/lab-08 master !227 ?2 > |
```

Рис. 3.2: Результаты работы программы

4 Вывод

Я приобрела навыки написания программы с использованием цикла.