

Лабораторная работа №8

рограммирование цикла. Обработка аргументов командной строки.

Виеру Женифер

Содержание

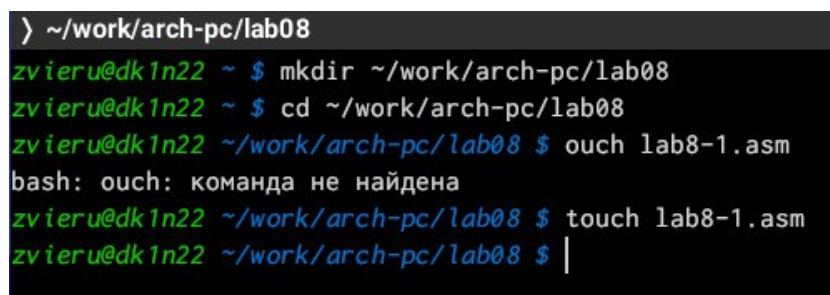
1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Самостоятельная работа	11
4	Выводы	14

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки..

2 Выполнение лабораторной работы

Создала каталог для программ лабораторной работы № 8, перешла в него и создала файл lab8-1.asm. Потом открыла его с помощью команды `те` и горячей клавиши F4 и написала в нем следующий текст(рис. 2.1).



```
> ~/work/arch-pc/lab08
zvieru@dk1n22 ~ $ mkdir ~/work/arch-pc/lab08
zvieru@dk1n22 ~ $ cd ~/work/arch-pc/lab08
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ouch lab8-1.asm
bash: ouch: команда не найдена
zvieru@dk1n22 ~/work/arch-pc/lab08 $ touch lab8-1.asm
zvieru@dk1n22 ~/work/arch-pc/lab08 $ |
```

Рис. 2.1: Создание lab8-1.asm

В lab8-1.asm написала программу вывода значений регистра `есх` (рис. 2.2).

```

> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru:~/work/arch-pc/lab08
lab8-1.asm [----] 11 L:[ 1+ 1 2/ 28] *(33 / 636b) 0116 0x
#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: Текст программы

Создала исполняемый файл запустила его и проверила его работу (рис. 2.3).

```

zvieru@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 2
2
1

```

Рис. 2.3: Запуск lab8-1.asm

Изменила текст программы добавив изменение значение регистра ecx в цикле: (рис. 2.4).

```

> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru:~/work/arch-pc/lab08
lab8-1.asm [----] 11 L:[ 1+ 1 2/ 28] *(33 / 636b) 0116 0x
#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. 2.4: Текст программы

Создала исполняемый файл запустила его и проверила его работу (рис. 2.5).

```

zvieru@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 2
1
zvieru@dk1n22 ~/work/arch-pc/lab08 $

```

Рис. 2.5: Запуск lab8-1.asm

Внесила изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop: (рис. 2.6).

```

> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab08
lab8-1.asm          [-M--]  9 L:[ 1+28  29/ 30] *(663 / 673b) 0010 0x00A
#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 labe
call quit

```

Рис. 2.6: Текст программы

Создала исполняемый файл и запустила его и по итоге число проходко циклов стало соответствовать числу введеному с клавиатуры (рис. 2.7).

```

zvieru@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 2
1
0

```

Рис. 2.7: Запуск lab8-1.asm

Создала lab8-2.asm и написала программу выводящую на экран аргументы командной строки(рис. 2.8).

```

) mc [zvieru@dk1n22.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab08
lab8-2.asm      [-M--]  9 L:[ 1+19  20/ 20] *(943 / 943b) <EOF>
#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

Создала исполняемый файл и запустила его и проверила его работу вводя /lab8-2 аргумент1 аргумент 2 'аргумент 3(рис. 2.9).

```

zvieru@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-2
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
zvieru@dk1n22 ~/work/arch-pc/lab08 $

```

Рис. 2.9: Запуск lab8-2.asm

Создала lab8-3.asm и написала программу вычисления суммы аргументов командной строки(рис. 2.10).


```

> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab08
lab8-3.asm      [-M--] 16 L:[ 1+17 18/ 29] *(795 /1428b) 1090 0x442
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
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)
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')
por 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. 2.10: Текст программы

Создала исполняемый файл и запустила его и проверила его работу вводя /lab8-3 12 13 7 10 5 (рис. 2.11).

```

zvieru@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
bash: ./main: Нет такого файла или каталога
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
zvieru@dk1n22 ~/work/arch-pc/lab08 $ |

```

Рис. 2.11: Запуск lab8-3.asm

Изменила текст программы для вычисления произведения аргументов командной строки. (рис. 2.12).

```

> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru:~/work/arch-pc/lab08
lab8-3.asm [-M--] 0 L:[ 8+23 31/ 38] *(255 / 322b) 0010 0x00A

_start:

pop ecx

pop edx

sub ecx,1

mov esi,1
mov eax,1

next:
cmp ecx,0
jz _end

pop eax
call atoi
mov ebx,eax
mov eax,esi
mul ebx
mov esi,eax
loop next

_end:
mov eax,msg
call sprint
mov eax,esi
call iprintLF

call quit

```

Рис. 2.12: Текст программы

Создала исполняемый файл и запустила его и проверила его работу вводя /lab8-3 1 2 3 (рис. 2.13).

```

zvieru@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-3
Результат: 1
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3
Результат: 6
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-3 4 5 6 7
Результат: 840

```

Рис. 2.13: Запуск lab8-3.asm

3 Самостоятельная работа

Создала файл lab8-4.asm в каталоге ~/work/arch-pc/lab06 и написала программу которая находит сумму значений функции $f(x)$ (рис. 3.1).

```

> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab08
lab8-4.asm      [----]  0 L:[ 2+26 28/ 40] *(256 / 360b) 0010 0x00

SECTION .data
prim DB 'f(x)=4x-3',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

mov eax,prim
call sprintf
next:
cmp ecx,0
jz _end

mov ebx,4
pop eax
call atoi
mul ebx

add eax,(-3)

add esi,eax

loop next

_end:
mov eax,otv
call sprintf
mov eax,esi
call iprintLF
call quit

```

Рис. 3.1: Программа вычисления суммы $f(x)$

Создала исполняемый файл, запустила его и проверила если программы работает (рис. 3.2).

```
zvieru@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3
f(x)=4x-3
Результат: 15
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-4 5 7 8
f(x)=4x-3
Результат: 71
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-4 5 7 8
f(x)=4x-3
Результат: 71
zvieru@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-4 4 7 5 9 8
f(x)=4x-3
Результат: 117
```

Рис. 3.2: Запуск lab8-4.asm

4 Выводы

Выполнив данную лабораторную работу я обрела теоретические и практические знания в организации стека, добавлении элемента в стеке, извлечении элемента из стека и в организации циклов