

Отчёта по лабораторной работе №8

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений.**

Мошаров Денис Максимович

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Самостоятельная работа	12
4	Выводы	15

Список иллюстраций

3.1	Создание каталога	6
3.2	Запленение 8.1	6
3.3	Проверка	7
3.4	Вносим изменения	8
3.5	Проверка	8
3.6	Редактирование	9
3.7	Проверка работы	9
3.8	Новый файл	9
3.9	Заполняем 8.2	10
3.10	Проверка	10
3.11	Новый файл	11
3.12	Заполнение 8.3	11
3.13	Проверка	11
3.14	Редактирование	12
3.15	Проверка работы	12
3.16	Новый файл	13
3.17	Пишем программу	13
3.18	Проверка	14

1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

3 Выполнение лабораторной работы

Создаем каталог для программ ЛБ8, и в нем создаем файл

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

Рис. 3.1: Создание каталога

Заполняем его в соответствии с листингом 8.1

```
-----
; 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:
; ----- 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
```

Рис. 3.2: Запленение 8.1

Создаем исполняемый файл и запускаем его

```
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
dmmosharov@dmmosharov:~/work/arch-pc/lab08$
```

Рис. 3.3: Проверка

Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле

```

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

```

Рис. 3.4: Вносим изменения

Создаем исполняемый файл и запускаем его

```

dmmosharov@dmmosharov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
Ошибка сегментирования (образ памяти сброшен на диск)

```

Рис. 3.5: Проверка

Регистр ecx принимает значения, в цикле label данный регистр уменьшается

на 2 командой `sub` и `loop`). Число проходов цикла не соответствует числу `N`, так как уменьшается на 2. Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало

```
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
```

Рис. 3.6: Редактирование

Создаем исполняемый файл и запускаем его

```
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рис. 3.7: Проверка работы

В данном случае число проходов цикла равна числу `N`.

Создаем новый файл

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

Рис. 3.8: Новый файл

Заполняем его в соответствии с листингом 8.2

```

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 ; добавляем к промежуточной сумме
mul esi,eax
; 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. 3.9: Заполняем 8.2

Создаем исполняемый файл и проверяем его работу, указав аргументы

```

dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ./lab8-2
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ./lab8-2 2 3 '4'
2
3
4

```

Рис. 3.10: Проверка

Программой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm

```
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ touch lab8-4.asm
```

Рис. 3.11: Новый файл

Открываем файл и заполняем его в соответствии с листингом 8.3

```
;-----
; 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:
; ----- 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
```

Рис. 3.12: Заполнение 8.3

Создаём исполняемый файл и запускаем его, указав аргументы

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

Рис. 3.13: Проверка

Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений

```

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
mov ebx,3
mul ebx
sub eax,1
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`

```

Рис. 3.14: Редактирование

Создаём исполняемый файл и запускаем его, указав аргументы

```

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

```

Рис. 3.15: Проверка работы

3.1 Самостоятельная работа

Вариант 2 Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверь-

те его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$. Создаем новый файл

```
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ touch lab8-4.asm
```

Рис. 3.16: Новый файл

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения $3x-1$

```
;-----  
; 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:  
; ----- 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
```

Рис. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы

```
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4.o
ld: no input files
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lag8-4.o
ld: cannot find lag8-4.o: No such file or directory
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
dmmosharov@dmmosharov:~/work/arch-pc/lab08$ ./lab8-4 4 5 6
Результат: 42
dmmosharov@dmmosharov:~/work/arch-pc/lab08$
```

Рис. 3.18: Проверка

4 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.