

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Ким Денис Вячеславович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Создание файла в новом каталоге	8
4.2	Ввод текста программы из листинга	9
4.3	Запуск программы	9
4.4	Изменение текста программы	10
4.5	Пример некорректной работы программы	10
4.6	Внесение изменений в текст программы	11
4.7	Запуск программы с изменённым текстом	11
4.8	Ввод текста программы из листинга	12
4.9	Запуск программы	12
4.10	Ввод текста программы из листинга	13
4.11	Запуск программы	13
4.12	Запуск программы с изменённым текстом	13
4.13	Ввод текста программы для вычисления суммы значений заданной функции	14
4.14	Запуск программы	14

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . . .	7
-----	---	---

1 Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

В ходе работы мне предстоит ознакомиться с организацией стека, инструкцией организации циклов, реализовывать их в NASM, а также создавать собственные программы для выполнения математических заданий.

3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы. Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux	
Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

4 Выполнение лабораторной работы

Создаём каталог для программ лабораторной работы № 8, переходим в него и создаём файл lab8-1.asm: (рис. 4.1).



```
dvkim@dk8n76 ~ $ mkdir ~/work/arch-pc/lab08
dvkim@dk8n76 ~ $ cd ~/work/arch-pc/lab08
dvkim@dk8n76 ~/work/arch-pc/lab08 $ touch lab8-1.asm
dvkim@dk8n76 ~/work/arch-pc/lab08 $
```

Рис. 4.1: Создание файла в новом каталоге

Вводим в файл lab8-1.asm текст программы из листинга 8.1: (рис. 4.2).


```

#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:

```

Рис. 4.2: Ввод текста программы из листинга

Создаём исполняемый файл и проверяем его работу: (рис. 4.3).

```

dvkim@dk8n76 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 3
3
2
1
dvkim@dk8n76 ~/work/arch-pc/lab08 $

```

Рис. 4.3: Запуск программы

Далее изменяем текст программы, добавив изменение значение регистра ecx в цикле: (рис. 4.4).

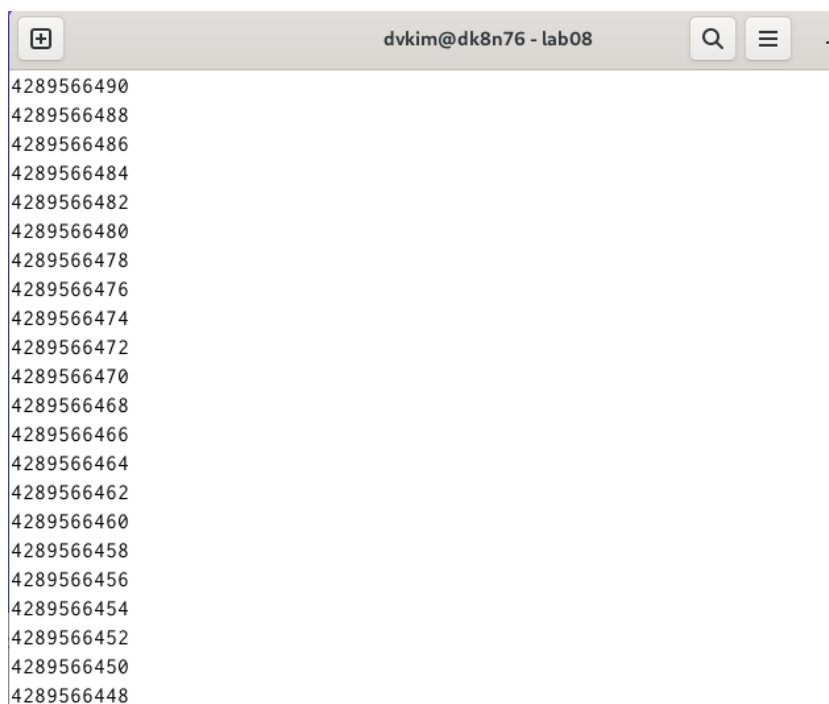
```

label:
sub  ecx,1
mov  [N],ecx
mov  eax,[N]
call iprintLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'

```

Рис. 4.4: Изменение текста программы

Создаём исполняемый файл и проверяем его работу. На выходе получаем множество больших чисел. Число проходов цикла не соответствует значению N: (рис. 4.5).



```

dvkim@dk8n76 - lab08
4289566490
4289566488
4289566486
4289566484
4289566482
4289566480
4289566478
4289566476
4289566474
4289566472
4289566470
4289566468
4289566466
4289566464
4289566462
4289566460
4289566458
4289566456
4289566454
4289566452
4289566450
4289566448

```

Рис. 4.5: Пример некорректной работы программы

Вносим изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop: (рис. 4.6).

```

label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения 'N'
pop ecx
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'

```

Рис. 4.6: Внесение изменений в текст программы

Создаём исполняемый файл и проверяем его работу. В данном случае число проходов цикла значению N соответствует: (рис. 4.7).

```

dvkim@dk8n76 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 3
2
1
0
dvkim@dk8n76 ~/work/arch-pc/lab08 $ 

```

Рис. 4.7: Запуск программы с изменённым текстом

Создаём файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и вводим в него текст программы из листинга 8.2: (рис. 4.8).

```

#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. 4.8: Ввод текста программы из листинга

Создаём исполняемый файл и запускаем его, указав заданные аргументы. Программа обработала 4 аргумента: (рис. 4.9).

```

dvkim@dk8n76 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
dvkim@dk8n76 ~/work/arch-pc/lab08 $

```

Рис. 4.9: Запуск программы

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создаём файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и вводим в него текст программы из листинга 8.3: (рис. 4.10).

```

%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'

```

Рис. 4.10: Ввод текста программы из листинга

Создайте исполняемый файл и запустите его, указав аргументы, как указано в примере: (рис. 4.11).

```

dvkim@dk8n76 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
dvkim@dk8n76 ~/work/arch-pc/lab08 $ 

```

Рис. 4.11: Запуск программы

Изменяем текст так, чтобы программа вычисляла не сумму аргументов, а их произведение. Запускаем программу: (рис. 4.12).

```

dvkim@dk8n76 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 54600
dvkim@dk8n76 ~/work/arch-pc/lab08 $ 

```

Рис. 4.12: Запуск программы с измененным текстом

Выполняем задания для самостоятельной работы. Напишем 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 $f(x)$ выбираем из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6 (у нас был 11 вариант): (рис. 4.13).

```
%include 'in_out.asm'
SECTION .data
func DB 'f(x)=15x+2',0
a DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,0
mov eax,func
call sprintf@PLT
next:
cmp ecx,0
jz _end
mov ebx,15
pop eax
call atoi
mul ebx
add eax,2
add esi,eax
```

Рис. 4.13: Ввод текста программы для вычисления суммы значений заданной функции

Создаём исполняемый файл и проверяем его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$: (рис. 4.14).

```
dvkim@dk8n76 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
dvkim@dk8n76 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4
f(x)=15x+2
Результат: 158
dvkim@dk8n76 ~/work/arch-pc/lab08 $
```

Рис. 4.14: Запуск программы

5 Выводы

В ходе данной работы я приобрёл навыки написания программ с использованием циклов и обработкой аргументов командной строки. Я также научился работать с циклами и писать собственные программы для выполнения задач.

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.