

Лабораторная работа №7

Дисциплина: Архитектура компьютера

Комягин Андрей Николаевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
	2.1 Самостоятельная работа	9
3	Вывод	12

Список иллюстраций

2.1	Создание каталога	6
2.2	Работа программы 7.1	6
2.3	Работа программы 7.2.1	6
2.4	Работа программы 7.2.2	7
2.5	Работа программы 7.3	8
2.6	Строки для объяснения	8
2.7	Строки для объяснения	8
2.8	Задание 1	10
2.9	Задание 2	11

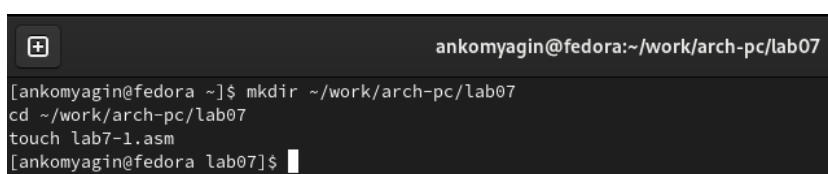
Список таблиц

1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Познакомиться с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

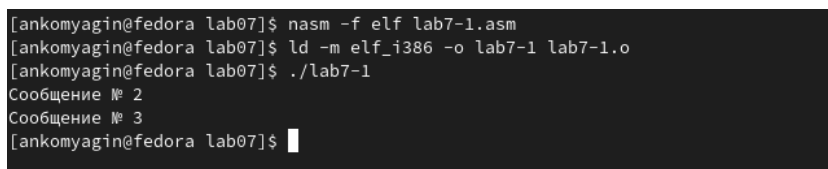
Создадим каталог для программ лабораторной работы 7. Создадим в нем файл **lab7-1.asm**(рис. 2.1).



```
ankomyagin@fedora:~/work/arch-pc/lab07
[ankomyagin@fedora ~]$ mkdir ~/work/arch-pc/lab07
cd ~/work/arch-pc/lab07
touch lab7-1.asm
[ankomyagin@fedora lab07]$
```

Рис. 2.1: Создание каталога

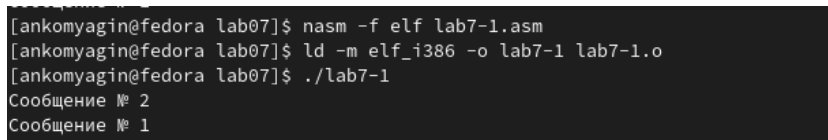
Заполним файл в соответствии с листингом **7.1**. Создадим исполняемый файл и запустим его (рис. 2.2).



```
[ankomyagin@fedora lab07]$ nasm -f elf lab7-1.asm
[ankomyagin@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[ankomyagin@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[ankomyagin@fedora lab07]$
```

Рис. 2.2: Работа программы 7.1

Изменим текст программы в соответствии с листингом **7.2**. Результат работы программы (рис. 2.3).




```
[ankomyagin@fedora lab07]$ nasm -f elf lab7-1.asm
[ankomyagin@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[ankomyagin@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 2.3: Работа программы 7.2.1

Изменим текст программы так, чтобы вывод программы был в порядке: **сообщение №3 - №2 - №1** (рис. 2.4).

```
[ankomyagin@fedora lab07]$ nasm -f elf lab7-1.asm
[ankomyagin@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[ankomyagin@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[ankomyagin@fedora lab07]$
```

Открыть ▾ 

lab7-1.asm
~/work/arch-pc/lab07

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.4: Работа программы 7.2.2

Создадим новый файл. Заполним его в соответствии с листингом 7.3 и проверим работу (рис. 2.5).

```

[ankomyagin@fedora lab07]$ touch lab7-2.asm
[ankomyagin@fedora lab07]$ nasm -f elf lab7-2.asm
[ankomyagin@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[ankomyagin@fedora lab07]$ ./lab7-2
Введите B: 58
Наибольшее число: 58
[ankomyagin@fedora lab07]$ ./lab7-2
Введите B: 4
Наибольшее число: 50
[ankomyagin@fedora lab07]$ ./lab7-2
Введите B: 52
Наибольшее число: 52
[ankomyagin@fedora lab07]$

```

Рис. 2.5: Работа программы 7.3

Создадим файл листинга из файла 7-2. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке. Ознакомимся с форматом.

Объясним 3 строки (рис. 2.6). В строке 14 происходит вычитание значения из регистра `eax` значения из регистра `ebx`. Результат вычитания сохраняется в регистре `eax`. В строке 15 происходит возврат значения из регистра `ebx` в стек, а в строке 16 это значение используется для выполнения команды `pop`, которая извлекает значение из стека и помещает его в регистр `ebx`.

14	<1> finished:
15 0000000B 29D8	<1> sub eax, ebx
16 0000000D 5B	<1> pop ebx

Рис. 2.6: Строки для объяснения

Удалим один операнд. Выполним трансляцию и посмотрим результат (рис. 2.7).

<pre> [ankomyagin@fedora lab07]\$ nasm -f elf -l lab7-2.lst lab7-2.asm lab7-2.asm:38: error: invalid combination of opcode and operands [ankomyagin@fedora lab07]\$ </pre>	<pre> 35 00000135 6A62EEEEEE call a0d1 ; Вызов подпрограммы перевода символа в число 36 0000013A A3[00000000] mov [max], max ; запись преобразованного числа в 'max' 37 ; ----- Сравнение max(A,C) и 'B' (как числа) 38 max max, 38 ;----- 38 ;error: invalid combination of opcode and operands 39 0000014E 7D0D[0A000000] cmp max, [B] ; Сравнение max(A,C) и 'B' </pre>
--	--

Рис. 2.7: Строки для объяснения

2.1 Самостоятельная работа

Вариант 7

1. Напишем программу для нахождения наименьшей целочисленной переменной. Создадим программу и проверим её работу. Тестовые данные 45,67,15 (рис. 2.8).

```
ankomyagin@fedora:~/work/arch-pc/lab07
[ankomyagin@fedora lab07]$ nasm -f elf task7_1.asm
[ankomyagin@fedora lab07]$ ld -m elf_i386 -o task7_1 task7_1.o
[ankomyagin@fedora lab07]$ ./task7_1
Наименьшее число : 45
[ankomyagin@fedora lab07]$
```

```
msg2 db "Наименьшее число : ",0h
A dd '45'
B dd '67'
C dd '15'
section .bss
min resb 90
section .text
global _start
_start:

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [min],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ;
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'

check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в `max`
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Рис. 2.8: Задание 1

2. Напишем программу, определяющая значение функции. $f(x) = 6a$, при $x = a$ и $f(x) = a + x$, при $x \neq a$ (рис. 2.9).

The image shows a terminal window on the left and an IDE window on the right. The terminal shows the compilation and execution of the program. The IDE shows the assembly code for the program.

```
[ankomyagin@fedora lab07]$ nasm -f elf ta
[ankomyagin@fedora lab07]$ ld -m elf_i386
[ankomyagin@fedora lab07]$ ./task7_2
Введите x:
1
Введите a:
1
Результат:
6
[ankomyagin@fedora lab07]$ ./task7_2
Введите x:
2
Введите a:
1
Результат:
3
[ankomyagin@fedora lab07]$
```

```
task7_2.asm
~/.work/arch-pc/lab07

lab7-2.asm  task7_2.asm x  report.md  lab7-2.ls

%include "in_out.asm"

SECTION .data
msg: DB 'Введите x: ',0
msg1: DB 'Введите a: ',0
msg2: DB 'Результат: ',0

SECTION .bss
x: RESB 80
a: RESB 80
otv: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintfLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov [x], eax
mov eax, msg1
call sprintfLF
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov [a], eax
mov eax, [x]
cmp eax, [a]
je EQUAL ; если x = a, переходим к метке EQUAL
ADD:
add eax, [a] ; если x не равен a, прибавляем a к x
jmp ANS ; переходим к метке ANS для вывода результата
EQUAL:
mov ebx, 6 ; ebx = 6
mul ebx ; умножаем x на 6 и сохраняем результат в eax
ANS:
mov [otv], eax ; сохраняем результат в otv
mov eax, msg2 ; вызов подпрограммы печати сообщения
"Результат:"
call sprintfLF
mov eax, [otv] ; вызов подпрограммы печати значения otv
call sprintfLF
```

Рис. 2.9: Задание 2

3 Вывод

В ходе работы я изучил команды условного и безусловного переходов. Приобрел навыки написания программ с использованием переходов. Познакомился с назначением и структурой файла листинга.