

Лабораторной работа №7

Воинов Кирилл Викторович

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задание для самостоятельной работы	13
4	Выводы	18

Список иллюстраций

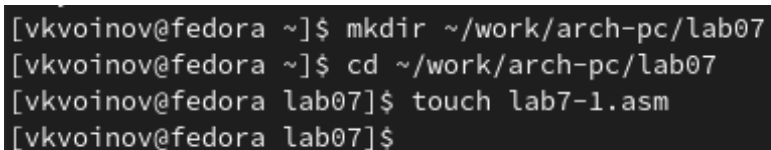
2.1	Создание каталога, переход в него и создание файла	5
2.2	Текст программы	6
2.3	Результат работы программы	6
2.4	Изменённый текст программы	7
2.5	Результат работы изменённой программы	7
2.6	Повторно изменённый текст программы	8
2.7	Результат работы повторно изменённой программы	8
2.8	Создание файла	9
2.9	Текст программы	9
2.10	Создание исполняемого файла	10
2.11	Проверка работы программы	10
2.12	Создание файла листинга	10
2.13	Открытие файла листинга	11
2.14	Три строки листинга	11
2.15	Удаление операнда	12
2.16	Трансляция	12
2.17	Выход	12
3.1	Текст программы	14
3.2	Результат работы программы	15
3.3	Текст программы	16
3.4	Результат работы программы	17

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

1. Создаю каталог для программ лабораторной работы No 7, перехожу в него и создаю файл lab7-1.asm. (рис. 2.1).



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

Рис. 2.1: Создание каталога, переход в него и создание файла

2. Ввожу в файл lab7-1.asm текст программы из листинга 7.1. (рис. 2.2).

```

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

```

Рис. 2.2: Текст программы

Создаю исполняемый файл и проверяю его работу. (рис. 2.3).

```

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

```

Рис. 2.3: Результат работы программы

Изменяю текст программы в соответствии с листингом 7.2. (рис. 2.4).

```

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

```

Рис. 2.4: Изменённый текст программы

Создаю исполняемый файл и проверяю его работу. (рис. 2.5).

```

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

```

Рис. 2.5: Результат работы изменённой программы

Изменяю текст программы, чтобы вывод сообщений был следующим: Сообщение No 3, Сообщение No 2, Сообщение No 1. (рис. 2.6).

```

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

```

Рис. 2.6: Повторно изменённый текст программы

Создаю исполняемый файл и проверяю его работу. (рис. 2.7).

```

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

```

Рис. 2.7: Результат работы повторно изменённой программы

3. Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. 2.8).


```
[vkvoinov@fedora lab07]$ touch lab7-2.asm
[vkvoinov@fedora lab07]$
```

Рис. 2.8: Создание файла

Ввожу текст программы из листинга 7.3 в lab7-2.asm. (рис. 2.9).

```
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 2.9: Текст программы

Создаю исполняемый файл и проверяю его работу для разных значений B. (рис.

2.10) и (рис. 2.11).

```
[vkvoinov@fedora lab07]$ nasm -f elf lab7-2.asm
[vkvoinov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[vkvoinov@fedora lab07]$ ./lab7-2
Введите B: 68
Наибольшее число: 68
```

Рис. 2.10: Создание исполняемого файла

```
[vkvoinov@fedora lab07]$ ./lab7-2
Введите B: 49
Наибольшее число: 50
[vkvoinov@fedora lab07]$ ./lab7-2
Введите B: 51
Наибольшее число: 51
[vkvoinov@fedora lab07]$
```

Рис. 2.11: Проверка работы программы

4. Создаю файл листинга для программы из файла lab7-2.asm. (рис. 2.12).

```
[vkvoinov@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[vkvoinov@fedora lab07]$
```

Рис. 2.12: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора mcedit. (рис. 2.13).

```

lab7-2.lst      [----] 45 L: [ 1+ 0 1/225] *(45 /14458b) 0117 0x075 [*][X]
1      %include 'in_out.asm'
1      <1> ;----- slen -----
2      <1> ; Функция вычисления длины сообщения
3      <1> slen:.....
4      00000000 53      <1> push    ebx.....
5      00000001 89C3    <1> mov     ebx, eax.....
6      <1> .....
7      <1> nextchar:.....
8      00000003 803800   <1> cmp     byte [eax], 0...
9      00000006 7403     <1> jz      finished.....
10     00000008 40       <1> inc     eax.....
11     00000009 EBF8     <1> jmp     nextchar.....
12     <1> .....
13     <1> finished:
14     0000000B 29D8     <1> sub     eax, ebx
15     0000000D 5B       <1> pop     ebx.....
16     0000000E C3       <1> ret.....
17     <1> .
18     <1> .
19     <1> ;----- sprint -----
20     <1> ; Функция печати сообщения
21     <1> ; входные данные: mov eax,<message>

```

Рис. 2.13: Открытие файла листинга

Три строки листинга(рис. 2.14).

```

33      check_B:
34      0000012A B8[00000000]    mov eax, max
35      0000012F E868FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число

```

Рис. 2.14: Три строки листинга

33 строка: check_B - указатель перехода в исходном тексте программы. 34 строка: 0000012A - адрес(смещение машинного кода от начала сегмента), B8[00000000] - машинный код, mov eax, max - исходный текст программы. 35 строка: 0000012F - адрес(смещение машинного кода от начала сегмента), E868FFFFFF - машинный код, call atoi ; Вызов подпрограммы перевода символа в число - исходный текст программы

Открываю файл с программой lab7-2.asm и в одной инструкции с двумя операндами удаляю один операнд. (рис. 2.15).

```

; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx ; Сравниваем 'A' и 'C' (удален операнд)
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'

```

Рис. 2.15: Удаление операнда

Выполняю трансляцию с получением файла листинга. (рис. 2.16).

```

[vkvoinov@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
[vkvoinov@fedora lab07]$

```

Рис. 2.16: Трансляция

Листинг создаётся. В него добавляется уведомление об ошибке (рис. 2.17).

```

27                                     ; ----- Сравниваем 'A' и 'C' (как символы)
28                                     cmp ecx ; Сравниваем 'A' и 'C' (удален операнд)
28          *****
28          error: invalid combination of opcode and operands
29 0000011C 7F0C                     jg check_B ; если 'A>C', то переход на метку 'check_B',
30 0000011E 8B0D[39000000]          mov ecx,[C] ; иначе 'ecx = C'
31 00000124 890D[00000000]          mov [max],ecx ; 'max = C'

```

Рис. 2.17: Выход

3 Задание для самостоятельной работы

1. Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c для варианта 18. (рис. 3.1).

```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее| число: ",0h
A dd 83
B dd 73
C dd 30
section .bss
max resb 10
section .text
global _start
_start:
mov ecx,[A]
mov [max],ecx
cmp [B],ecx
jg check_C
mov ecx,[B]
mov [max],ecx

check_C:
mov ecx,[max]
cmp [C],ecx
jg fin
mov ecx,[C]
mov [max],ecx
fin:
mov eax, msg2
call sprint
mov eax,[max]
call iprintLF
call quit

```

Рис. 3.1: Текст программы

Создаю исполняемый файл и проверяю его работу. (рис. 3.2).

```
[vkvoinov@fedora lab07]$ nasm -f elf lab7-3-18.asm
[vkvoinov@fedora lab07]$ ld -m elf_i386 -o lab7-3-18 lab7-3-18.o
[vkvoinov@fedora lab07]$ ./lab7-3-18
Наибольшее число: 30
[vkvoinov@fedora lab07]$
```

Рис. 3.2: Результат работы программы

2. Пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений для варианта 18. (рис. 3.3).

```

#include 'in_out.asm'
SECTION .data
msgx: DB 'Введите x: ',0
msga: DB 'Введите a: ',0
div: DB 'Результат: ',0
SECTION .bss
x: RESB 80
a: RESB 80
max resb 10
SECTION .text
GLOBAL _start
_start:
; ----- Вывод сообщения 'Введите x: '
mov eax,msgx
call sprint
; ----- Ввод 'x'
mov ecx,x
mov edx,10
call sread
; ----- Преобразование 'x' из символа в число
mov eax,x
call atoi
mov [x],eax
; ----- Вывод сообщения 'Введите a: '
mov eax,msga
call sprint
; ----- Ввод 'a'
mov ecx,a
mov edx,10
call sread
; ----- Преобразование 'a' из символа в число
mov eax,a
call atoi |
mov [a],eax
; ----- Сравниваем 'A' и 1
mov ecx, [a]
cmp ecx,1
je check_x
mov eax, [a]
mul ecx
mov ecx, eax
jmp fin
check_x:
mov ecx, [x]
add ecx,10
; ----- Вывод результата
fin:
mov eax, div
call sprint
mov eax,ecx
call iprintLF
call quit

```

Рис. 3.3: Текст программы

Создаю исполняемый файл и проверяю его работу для значений x и a из таблицы 7.6. (рис. 3.4).

```
[vkvoinov@fedora lab07]$ nasm -f elf lab7-4-18.asm
[vkvoinov@fedora lab07]$ ld -m elf_i386 -o lab7-4-18 lab7-4-18.o
[vkvoinov@fedora lab07]$ ./lab7-4-18
Введите x: 1
Введите a: 2
Результат: 4
[vkvoinov@fedora lab07]$ ./lab7-4-18
Введите x: 2
Введите a: 1
Результат: 12
[vkvoinov@fedora lab07]$
```

Рис. 3.4: Результат работы программы

4 Выводы

На этой лабораторной работе я изучил команды условного и безусловного переходов, приобрел навыки написания программ с использованием переходов и ознакомился с назначением и структурой файла листинга.