

Отчёт по лабораторной работе №7

Дисциплина: Архитектура Компьютера

Иванов Сергей Владимирович

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выводы	14

Список иллюстраций

2.1	Создание lab7-1.asm	5
2.2	Текст из листинга 7.1	5
2.3	Исполняемый файл	6
2.4	Программа из листинга 7.2	6
2.5	Изменение программы	7
2.6	Запуск исполняемого файла	7
2.7	Программа из листинга 7.3	8
2.8	Файл листинга	8
2.9	Удаление операнда	9
2.10	Листинг с ошибкой	9
2.11	Программа lab7-3.asm	9
2.12	Программа lab7-4.asm	11

1 Цель работы

Целью лабораторной работы является изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов и знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы № 7, перейдём в него и создадим файл lab7-1.asm.(Рис. 2.1)

```
svivanov1@svivanov1:~$ mkdir ~/work/arch-pc/lab07
svivanov1@svivanov1:~$ cd ~/work/arch-pc/lab07
svivanov1@svivanov1:~/work/arch-pc/lab07$ touch lab7-1.asm
svivanov1@svivanov1:~/work/arch-pc/lab07$
```

Рис. 2.1: Создание lab7-1.asm

Введем в файл lab7-1.asm текст программы из листинга 7.1 (Рис. 2.2)

```
GNU nano 7.2 /home/svivanov1/work/arch-pc/lab07/lab7-1.asm
#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 _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
[ Прочитано 20 строк ]
```

Рис. 2.2: Текст из листинга 7.1

Создадим исполняемый файл и запустим его. (Рис. 2.3)

```

svivanov1@svivanov1:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
svivanov1@svivanov1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
svivanov1@svivanov1:~/work/arch-pc/lab07$ █

```

Рис. 2.3: Исполняемый файл

Изменим текст программы в соответствии с листингом 7.2, создадим исполняемый файл и проверим его работу. (Рис. 2.4)

```

svivanov1@svivanov1:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
svivanov1@svivanov1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
svivanov1@svivanov1:~/work/arch-pc/lab07$ █

```

Рис. 2.4: Программа из листинга 7.2

Изменим текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим: Сообщение №3, Сообщение №2, Сообщение №1. (Рис. 2.5)

```

%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. 2.5: Изменение программы

Создадим исполняемый файл и запустим его. (Рис. 2.6)

```

svivanov1@svivanov1:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
svivanov1@svivanov1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 2.6: Запуск исполняемого файла

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Введём в lab7-2.asm текст из листинга 7.3. Создадим исполняемый файл и проверим его работу для разных значений В. (Рис. 2.7)

```

svivanov1@svivanov1:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
svivanov1@svivanov1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 7
Наибольшее число: 50
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 89
Наибольшее число: 89
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 52
Наибольшее число: 52
svivanov1@svivanov1:~/work/arch-pc/lab07$

```

Рис. 2.7: Программа из листинга 7.3

Создадим файл листинга для программы из файла lab7-2.asm и откроем его с помощью mcedit. (Рис. 2.8)

```

/home/sv-7-2.lst  [----]  8 L: [ 1+ 0 1/225] *(8 /13933b) 0032 0x020 [*][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>
22 <1> sprint:
23 0000000F 52      <1> push    edx
24 00000010 51      <1> push    ecx

```

Рис. 2.8: Файл листинга

Объясним содержание строк 4,5 и 8:

Строка 4: 4-номер строки, 00000000-адрес строки, 53-машинный код, push ebx-исходный текст программы.

Строка 5: 5-номер строки, 00000001-адрес строки, 89C3-машинный код, mov ebx,eax-исходный текст программы

Строка 8: 8-номер строки, 00000003-адрес строки, 803800-машинный код, str byte[eax],0-исходный текст программы

Откроем файл с программой lab7-2.asm и в строке mov edx,10 удалим 10 (Рис. 2.9)

```
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx, 10
call sread
; ----- Преобразование 'B' из символа в число
```

Рис. 2.9: Удаление операнда

Создадим файл листинга и откроем его. Как мы видим, в строке в которой мы удалили операнд, появилось сообщение об ошибке, а в первой части листинга появились *. (Рис. 2.10)

```
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 mov edx
18 ***** error: invalid combination of opcode and opera
19 000000F7 E847FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в чи
21 000000FC B8[0A000000] mov eax,B
22 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода символа
23 00000106 A3[0A000000] mov [B],eax ; запись преобразованного числа в '
```

Рис. 2.10: Листинг с ошибкой

Напишем программу нахождения наименьшей из 3 целочисленных переменных a,b и c в соответствии со своим вариантом(8). Создадим исполняемый файл и проверим его работу. Программа работает правильно. (Рис. 2.11)

```
svivanov1@svivanov1:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
svivanov1@svivanov1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 33
svivanov1@svivanov1:~/work/arch-pc/lab07$
```

Рис. 2.11: Программа lab7-3.asm

Листинг программы для самостоятельной работы 1(lab7-3.asm):

```

%include 'in_out.asm'

section .data
msg2 db "Наименьшее число: ",0h
A dd '52'
B dd '33'
C dd '40'

section .bss
min resb 10

section .text
global _start
_start:
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jnl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в `min`
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]

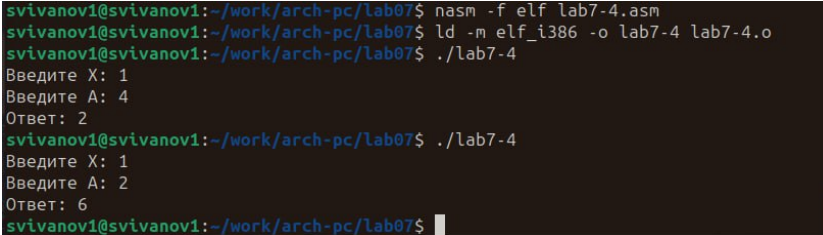
```

```

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 ;
mov eax,[min]
call iprintLF ;
call quit ; Выход

```

Напишем программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Создадим исполняемый файл и проверим его работу для значений x и a из 7.6.(Вариант 8). (Рис. 2.12)



```

svivanov1@svivanov1:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
svivanov1@svivanov1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-4
Введите X: 1
Введите A: 4
Ответ: 2
svivanov1@svivanov1:~/work/arch-pc/lab07$ ./lab7-4
Введите X: 1
Введите A: 2
Ответ: 6
svivanov1@svivanov1:~/work/arch-pc/lab07$

```

Рис. 2.12: Программа lab7-4.asm

Листинг программы для самостоятельной работы 2(lab7-4.asm):

```

%include 'in_out.asm'
section .data
msg1 DB "Введите X: ",0h
msg2 DB "Введите A: ",0h
msg3 DB "Ответ: ",0h

```

```

section .bss
x: RESB 80
a: RESB 80
ans: RESB 80
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
mov [x],eax
mov eax,msg2
call sprint
mov ecx,a
mov edx, 80
call sread
mov eax,a
call atoi
mov [a],eax

mov eax,[a]
cmp eax,3
jl xsa

mov eax,[x]

```

```
add eax,1
```

```
jmp ansv
```

```
xsa:
```

```
mov ebx,3
```

```
mov eax,[a]
```

```
mul ebx
```

```
ansv:
```

```
mov [ans],eax
```

```
mov eax,msg3
```

```
call sprint
```

```
mov eax,[ans]
```

```
call iprintLF
```

```
call quit
```

3 Выводы

В результате выполнения лабораторной работы мы изучили команды условного и безусловного переходов, приобрели навыки написания программ с использованием переходов и познакомились с назначением и структурой файла листинга.