

Отчёт по лабораторной работе №7

Архитектура компьютера

Андреева Софья Владимировна

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задание для самостоятельной работы.	9
4	Выводы	13

Список иллюстраций

2.1	Результат работы файла lab7-1.asm	5
2.2	Запуск измененного файла.	6
2.3	Измененный текст файла	6
2.4	Запуск измененного файла.	6
2.5	Работа файла lab7-2.asm	7
2.6	Файл листинга lab7-2.lst.	7
2.7	Удаление одного операнда	8
2.8	Ошибка	8
2.9	Файл листинга	8
3.1	Программа вычисления функции	10
3.2	Запуск файла	11
3.3	Программа вычисления функции.Вариант 12	11
3.4	Запуск файла	12

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

Создадим каталог для программ лабораторной работы № 7, перейдем в него и создадим файл lab6-7.asm. Введем в файл lab7-1.asm текст программы из листинга 7.1. Создадим исполняемый файл и запустим его. Использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. (рис. 2.1).

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

svandreeva@svandreeva-VirtualBox: ~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
lab7-1.asm:1: error: unable to open include file 'in_out.asm': No such file or directory
svandreeva@svandreeva-VirtualBox: ~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
svandreeva@svandreeva-VirtualBox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
svandreeva@svandreeva-VirtualBox: ~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
svandreeva@svandreeva-VirtualBox: ~/work/arch-pc/lab07$ █
```

Рис. 2.1: Результат работы файла lab7-1.asm

Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end`. Создадим исполняемый файл и проверим его работу (рис. 2.2).

```

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

```

Рис. 2.2: Запуск измененного файла.

Изменим текст программы (рис. 2.3), чтобы вывод программы был следующим:
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

#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. 2.3: Измененный текст файла

Создадим исполняемый файл и запустим его. Всё получилось (рис. 2.4).

```

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

```

Рис. 2.4: Запуск измененного файла.

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучим текст программы из листинга 7.3 и введем в lab7-2.asm. Создадим исполняемый файл и проверим его работу для разных значений В, я ввела сначала 5, а затем 70. (рис. 2.5).

```
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ mc
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 5
Наибольшее число: 50
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 70
Наибольшее число: 70
```

Рис. 2.5: Работа файла lab7-2.asm

Создадим файл листинга для программы из файла lab7-2.asm: `nasm -f elf -l lab7-2.lst lab7-2.asm`. Откроем файл листинга lab7-2.lst с помощью текстового редактора `mcedit`. Внимательно ознакомимся с его форматом и содержанием. (рис. 2.6).

```
home/svandreeva/work/arch-pc/lab07/lab7-2.lst [27 L: 1+0 1/225] * (27 / 14474)
1      %include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:-----
5      00000000 53      <1> push    ebx
6      00000001 89C3    <1> mov     ebx, eax
7      <1> .....
8      <1> nextchar:-----
9      00000003 803800    <1> cmp     byte [eax], 0
10     00000006 7403     <1> jz      finished
11     00000008 40      <1> inc     eax
12     00000009 EBF8     <1> jmp     nextchar
13     <1> .....
14     <1> finished:-----
15     0000000B 29D8     <1> sub     eax, ebx
16     0000000D 5B      <1> pop     ebx
17     0000000E C3      <1> ret
18     <1> .....
19     <1> .....
20     <1> ;----- sprint -----
21     <1> ; Функция печати сообщения
22     <1> ; входные данные: mov eax,<message>
23     <1> sprint:-----
24     0000000F 52      <1> push    edx
25     00000010 51      <1> push    ecx
26     00000011 53      <1> push    ebx
27     00000012 50      <1> push    eax
28     00000013 E8E8FFFFFF <1> call    slen
29     <1> .....
30     00000018 89C2     <1> mov     edx, eax
31     0000001A 5B      <1> pop     eax
32     <1> .....
33     0000001B 89C1     <1> mov     ecx, eax
34     0000001D B801000000 <1> mov     ebx, 1
35     00000022 B804000000 <1> mov     eax, 4
36     00000027 CD80     <1> int     80h
37     <1> -----
```

Рис. 2.6: Файл листинга lab7-2.lst.

Рассмотрим 24 строку: “00000101 B8 [0A000000] mov eax,B”. Ее адрес “00000101”, Машинный код - “B8 [0A000000]”, а mov eax,B - исходный текст программы, означающий что в регистр eax мы вносим значения переменной B. Рассмотрим 38 строку: “00000134 E863FFFFFF call atoi”. Ее адрес “00000134”, Машинный код - E863FFFFFF, а call atoi - исходный текст программы, означающий что символ лежащий в строке выше переводится в число. Рассмотрим 50 строку: “00000162 A1[00000000] mov eax,[max]”. Ее адрес “00000162”, Машинный код - A1[00000000], а mov eax,[max] - исходный текст программы, означающий что число хранившееся в переменной max записывается в регистр eax.

Откроем файл с программой lab7-2.asm и в инструкции с двумя операндами удалим один операнд (вместо cmp ecx, [C] оставим cmp ecx)(рис. 2.7).

```
mov [max],ecx ; max = C
; ----- 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:
```

Рис. 2.7: Удаление одного операнда

Выполним трансляцию с получением файла листинга. Нам выдало ошибку, так как для программы нужно два операнда(рис. 2.8).

```
svandreeva@svandreeva-VirtualBox: ~/work/arch-pc/lab07$ mc
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.8: Ошибка

В файле листинга нам показывает где именно ошибка и с чем она связана.(рис. 2.9).

```
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx ; Сравниваем 'A' и 'C'
28 *****
29 0000011C 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B'
30 0000011F 0000[00000000] mov ecx,[C] ; иначе 'ecx = C'
```

Рис. 2.9: Файл листинга

3 Задание для самостоятельной работы.

Напишем программу в файле samr12.1.asm для нахождения наименьшей из 3 целочисленных переменных A, B и C (рис. 3.1).

```

SECTION .data
A1 DB 'Введите число A: ',0h
B1 DB 'Введите число B: ',0h
C1 DB 'Введите число C: ',0h
otv DB 'Наименьшее число: ',0h
SECTION .bss
min RESB 20
A RESB 20
B RESB 20
C RESB 20

SECTION .text
GLOBAL _start
_start:

mov eax,A1
call sprint

mov ecx,A
mov edx,20
call sread

mov eax, A
call atoi
mov [A],eax

xor eax,eax

mov eax,B1
call sprint

mov ecx,B
mov edx,20
call sread
mov eax,B

```

Рис. 3.1: Программа вычисления функции

Создадим исполняемый файл и запустим его. Мой вариант 12, поэтому проверим программу для значений 99, 29 и 26. Все исполнилось корректно (рис. 3.2).

```
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/Lab07$ nasm -f elf samr12.1.asm
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/Lab07$ ld -m elf_i386 -o samr12.1 samr12.1.o
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/Lab07$ ./samr12.1
Введите число A: 99
Введите число B: 29
Введите число C: 26
Наименьшее число: 26
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/Lab07$
```

Рис. 3.2: Запуск файла

Напишем программу в файле samr12.2.asm, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции и выводит результат вычисления. Мой вариант-12. Составим программу для функции $f(x)=ax$ при $x < 5$ и $f(x)=x-5$ при $x \geq 5$ (рис. 3.3).

```
GNU nano 6.2 /home/svandreeva/work/arch-pc/Lab07
#include 'in_out.asm'

section .data
    msgf db 'f(x) = a*x, x<5',10, 9, 'x-5, ','x>=5',0h
    msgx db 'Введите x: ',0h
    msga db 'Введите a: ',0h
    msg2 db 'f(x) = ',0h

section .bss
    res resb 10
    x resb 10
    a resb 10

section .text
    global _start

_start:
; ----- Вывод функции
    mov eax, msgf
    call sprintLF

; ----- Получение переменной x
    mov eax, msgx
    call sprint

    mov ecx, x
    mov edx, 10
    call sread

    mov eax, x
    call atoi ; Вызов подпрограммы перевода символа в число
    mov [x], eax ; запись преобразованного числа в 'x'

; ----- Получение переменной a
    mov eax, msga
    call sprint
```

Рис. 3.3: Программа вычисления функции. Вариант 12

Создадим исполняемый файл и запустим его. Проверим его для значений (3;7)

и (6;4). Все исполнилось корректно (рис. 3.4).

```
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf samr12.2.asm
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o samr12.2 samr12.2.o
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ ./samr12.2
f(x) = a*x, x<5
      x-5, x>=5
Введите x: 3
Введите a: 7
f(x) = 21
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$ ./samr12.2
f(x) = a*x, x<5
      x-5, x>=5
Введите x: 6
Введите a: 4
f(x) = 1
svandreeva@svandreeva-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.4: Запуск файла

4 Выводы

Я изучила команды условного и безусловного переходов и приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.