

Лабораторная работа №7

Архитектура компьютера

Казначеева Кристина Никитична

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
4	Вывод	13

1 Цель работы

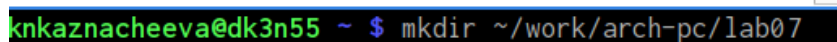
Лабораторная работа направлена на рассмотрение команды условного и безусловного перехода, формирование навыков написания программ с использованием переходов, а также на знакомство с назначением и структурой файла листинга.

2 Задание

В данной лабораторной работе мы изучим безусловные переходы (jmp) и программу поиска наибольшего из трёх целых чисел. Также мы рассмотрим структуру файлов листинга и реализацию переходов в NASM.

3 Выполнение лабораторной работы

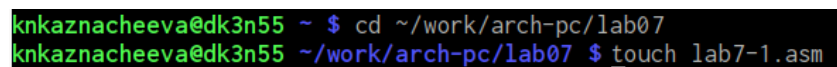
Создадим каталог lab07 и перейдём в него (рис. 3.1).

A terminal window with a black background and green text. The prompt is 'knkaznacheeva@dk3n55 ~ \$' and the command entered is 'mkdir ~/work/arch-pc/lab07'.

```
knkaznacheeva@dk3n55 ~ $ mkdir ~/work/arch-pc/lab07
```

Рис. 3.1: Создание каталога

Затем создадим файл lab7-1.asm (рис. 3.2).

A terminal window with a black background and green text. The first line shows the prompt 'knkaznacheeva@dk3n55 ~ \$' and the command 'cd ~/work/arch-pc/lab07'. The second line shows the prompt 'knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 \$' and the command 'touch lab7-1.asm'.

```
knkaznacheeva@dk3n55 ~ $ cd ~/work/arch-pc/lab07
knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 3.2: Создание файла

Введём в файл lab7-1.asm текст программы с использованием инструкции jmp (рис. 3.3).

```

lab7-1.asm      [-M--] 41 L:[ 1+19 20/ 20] *(6
%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 sprintLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 3.3: Ввод текста программы

Создадим исполняемый файл и запустим его (рис. 3.4).

```

knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3

```

Рис. 3.4: Проверка работы исходного файла

Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим текст программы таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу (рис. 3.5).

```

lab7-1.asm      [-M--] 41 L:[ 1+21 22/ 22] *(
%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 sprintfLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 3.5: Замена текста программы

Запустим исполняемый файл его (рис. 3.6).

```

knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1

```

Рис. 3.6: Проверка работы исходного файла

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. 3.7).

```

knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-2.asm

```

Рис. 3.7: Создание файла

Введём в файл lab7-2.asm программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С (рис. 3.8):

```

lab7-2.asm      [-M--] 50 L:[ 1+19 20/ 49] *(432 /17
%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

```

Рис. 3.8: Ввод текста программы

Создадим исполняемый файл и проверим его работу, например, для значения $B = 98$ (рис. 3.9).

```

knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 98
Наибольшее число: 98

```

Рис. 3.9: Проверка работы исходного файла

Чтобы получить файл листинга, создадим файл листинга для программы из файла lab7-2.asm, указав ключ -l и задав имя файла листинга в командной строке (рис. 3.10).

```

knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm

```

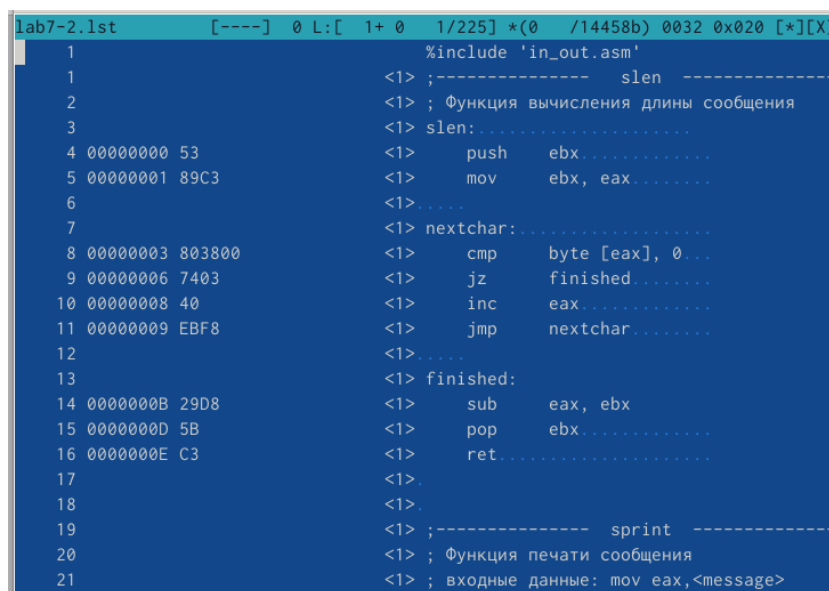
Рис. 3.10: Создание файл листинга для программы из файла

Откроем файл листинга lab7-2.lst с помощью mcedit (рис. 3.11).

```
knkaznacheeva@dk3n55 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
```

Рис. 3.11: Открытие файла листинга

В содержимом листинга можно увидеть построчное отображение ассемблерного кода и соответствующего машинного кода, сгенерированного ассемблером. Файл листинга содержит: адреса (адреса памяти, где располагается каждая инструкция), машинный код, шестнадцатеричный (фактические машинные инструкции, сгенерированные ассемблером, отображаются в шестнадцатеричном формате) и символьную информацию (информация о символах (метках, переменных), используемых в вашем коде) (рис. 3.12).



```
lab7-2.lst  [----]  0 L: 1+ 0 1/225 *(0 /14458b) 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>
```

Рис. 3.12: Содержимое листинга

Откроем файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалим один операнд, затем выполним трансляцию с получением файла листинга. Удаление операнда из инструкции с двумя операндами создаёт синтаксическую ошибку, которую ассемблер не может разрешить. Процесс сборки завершится неудачей, что предотвратит создание объектного файла, но файл

листинга всё равно будет сгенерирован с сообщениями об ошибках, указывающими на проблему (рис. 3.13).

```
knkznacheeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 3.13: Трансляция с получением файла листинга после удаления одной из операнд

Создадим файл lab7-3.asm (рис. 3.14).

```
knkznacheeva@dk2n21 ~ $ touch lab7-3.asm
```

Рис. 3.14: Создание файла

Напишем в полученном файле программу нахождения наименьшей из 3 целочисленных переменных a, b и c для варианта 14 (рис. 3.15).

```
/afs/.dk.sci.~7/lab7-3.asm 582/1538 3
%include 'in_out.asm'

section .data
    msg1 db 'Введите B: ',0h
    msg2 db "Наименьшее число: ",0h
    A dd 81 ; Now numeric values
    C dd 72
section .bss
    min resd 1 ; Reserve a doubleword for the minimum
    B resd 1 ; Reserve a doubleword for B

section .text
    global _start

_start:
    ; Get input for B (using read_int from previous response
    for efficiency and correctness)
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, 19
    int 80h

    mov eax, 3
    mov ebx, 0
    mov ecx, B
    mov edx, 10
    int 80h
    call read_int
```

Рис. 3.15: Ввод текста программы

Создадим исполняемый файл и проверим его работу (рис. 3.16).

```
knkaznacheeva@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-3
Введите В: 22
Наименьшее число: 22
knkaznacheeva@dk2n21 ~/work/arch-pc/lab07 $
```

Рис. 3.16: Проверка работы исходного файла

Создадим файл lab7-4.asm (рис. 3.17).

```
knkaznacheeva@dk2n21 ~/work/arch-pc/lab07 $ touch lab7-4.asm
```

Рис. 3.17: Создание файла

Напишем в полученном файле программу (вариант 14), которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений (рис. 3.18).

```
lab7-4.asm      [----]  0 L:[ 56+ 6  62/104[*][X
mov ebx, [three]
imul ebx, ; 3 * X
add eax, [one]
mov [fin], eax
jmp output_result

equal_case:
; X < A: Calculate 3 * A + 1
mov eax, [A]
mov ebx, [three]
imul ebx, ; 3 * X
add eax, [one]
mov [fin], eax

output_result:
; Output the result
mov eax, 4
mov ebx, 1
mov ecx, msg3
mov edx, 20
int 80h

mov eax, [fin]
call iprintLF ; Assuming iprintLF works correctly
1Почсть 2Соан 3Блок 4За~на 5Копия 6Петь 7Поиск
```

Рис. 3.18: Ввод текста программы

Создадим исполняемый файл и проверим его работу для значений x и a , сначала для $x=2$ и $a=3$ (рис. 3.19).

```
knkaznacheeva@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
knkaznacheeva@dk2n21 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
knkaznacheeva@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-4
Введите A: 3
Введите X: 2
Результат: 7
```

Рис. 3.19: Проверка работы исходного файла при $x=2$ и $a=3$

Затем проверим работу исполняемого файла для значений $x=4$ и $a=2$ (рис. 3.20).

```
knkaznacheeva@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-4
Введите A: 2
Введите X: 4
Результат: 13
```

Рис. 3.20: Проверка работы исходного файла при $x=4$ и $a=2$

4 Вывод

В рамках лабораторной работы были изучены команды условного и безусловного перехода, структура файлов листинга и особенности реализации переходов в среде NASM. Были получены практические навыки написания программ с использованием переходов.