

Лабораторная работа №7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений.**

Виеру Женифер

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Самостоятельная работа	9
4	Выводы	12

1 Цель работы

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

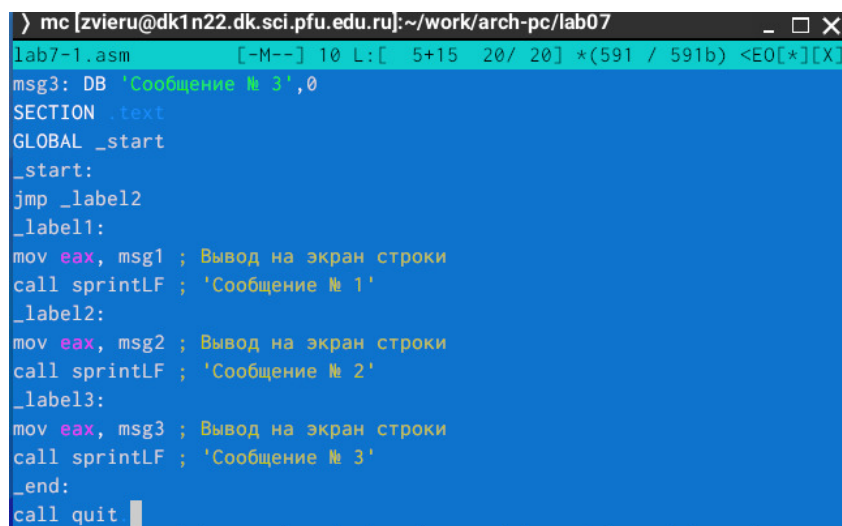
2 Выполнение лабораторной работы

Создала каталог для программ лабораторной работы № 7, перешла в него и создала файл lab6-1.asm. (рис. 2.1).

```
zvieru@dk1n22 ~ $ mkdir ~/work/arch-pc/lab07
zvieru@dk1n22 ~ $ cd ~/work/arch-pc/lab07
zvieru@dk1n22 ~/work/arch-pc/lab07 $ touch lab7-1.asm
zvieru@dk1n22 ~/work/arch-pc/lab07 $
```

Рис. 2.1: Создание файла для выполнения лабораторной работы

Потом открыла его в mc и написала пример программы для использования инструкции jmp (рис. 2.2).



```
> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab07
lab7-1.asm [-M--] 10 L:[ 5+15 20/ 20] *(591 / 591b) <E0[*][X]
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
```

Рис. 2.2: Программа с использованием инструкции jmp

Создала исполняемый файл и запустила его (рис. 2.3).

```

zvieru@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
zvieru@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
zvieru@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
zvieru@dk1n22 ~/work/arch-pc/lab07 $ |

```

Рис. 2.3: Запуск lab7-1.asm

Потом изменю текст программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу (рис. 2.4).

```

> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab07
lab7-1.asm [----] 11 L:[ 1+20 21/ 23] *(606 / 623b) 0010 0x0[*][X
%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

```

Рис. 2.4: Новая программа

Создала исполняемый файл и запустила его (рис. 2.5).

```

zvieru@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
zvieru@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
zvieru@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 2.5: Запуск lab7-1.asm

Потом создала файл lab7-1.asm и открыла его и написала программу которая выбирает наибольшее число из 3-х (рис. 2.6).

```
> mc [zvieru@dk1n22.dk.sci.pfu.edu.ru:~/work/arch-pc/lab07]
lab7-2.asm [----] 11 L: [ 1+ 2 3/ 49] *(49 /1743b) 1077 0x43[*][X
#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
```

Рис. 2.6: Код программы lab7-2.asm

Создала исполняемый файл и запустила его (рис. 2.7).

```

zvieru@dk1n22 ~/work/arch-pc/lab07 $ touch lab7-2.asm
zvieru@dk1n22 ~/work/arch-pc/lab07 $ mc

zvieru@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
zvieru@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
zvieru@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 23
Наибольшее число: 50

```

Рис. 2.7: Запуск lab7-2.asm

Создала файл листинга для программы из файла lab7-2.asm и открыла файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit. Объяснение одной строки: Эта строка находится на 21 месте, ее адрес “00000101”, Машинный код - B8 [00000000], а mov eax,B - исходный текст программы, означающий что в регистр eax мы вносим значения переменной B.(рис. 2.8).

```

21 00000101 B8[0A000000]          mov eax,B

```

Рис. 2.8: Объяснение первой строки

Объяснение другой строки:Эта строка находится на 35 месте, ее адрес “00000135”, *, Машинный код - E862FFFFFF, а call atoi - исходный текст программы, означающий что символ лежащий в строке выше переводится в число.(рис. 2.9).

```

35 00000135 E862FFFFFF          call atoi

```

Рис. 2.9: Объяснение второй строки

Объяснение другой строки:Эта строка находится на 47 месте, ее адрес “00000163”, Машинный код - A1[00000000], а mov eax,[max] - исходный текст программы, означающий что число хранившееся в переменной тах записывается в регистр eax. (рис. 2.10).

```

47 00000163 A1[00000000]          mov eax,[max]

```

Рис. 2.10: Объяснение третьей строки

Открыла файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалила один операнд.(рис. 2.11).

```
check_B:
mov  eax,
call atoi ; Вызов подпрограммы перевода символа в число
```

Рис. 2.11: Удаление одной из друкя команд

Выполнила трансляцию с получением файла листинга (рис. 2.12).

```
zvieru@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
zvieru@dk1n22 ~/work/arch-pc/lab07 $
```

Рис. 2.12: Запуск lab7-2.asm

Зметим что в самой программе написано чтотшибка (рис. 2.13).

```
34                                     mov  eax,
34      *****                      error: invalid combination of opcode
```

Рис. 2.13: Программа

3 Самостоятельная работа

Написала программу нахождения наименьшей из 3 целочисленных переменных (рис. 3.1).

```

> mc [zvieu@dk1n22.dk.sci.pfu.edu.ru:~/work/arch-pc/lab07
ex1.asm [----] 0 L:[ 1+18 19/ 81] *(312 / 844b) 0099 0x06[*][X
%include 'in_out.asm'

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
call atoi
mov [B],eax

xor eax,eax

```

Рис. 3.1: Текст программы

Создала исполняемый файл, запустила его и проверила если программы работает (рис. 3.2).

```
zvieru@dk1n22 ~/work/arch-pc/lab07 $ touch ex1.asm
zvieru@dk1n22 ~/work/arch-pc/lab07 $ mc

zvieru@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf ex1.asm
zvieru@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o ex1 ex1.o
zvieru@dk1n22 ~/work/arch-pc/lab07 $ ./ex1
Введите число A: 79
Введите число B: 83
Введите число C: 41
Наименьшее число: 41
```

Рис. 3.2: Запуск ex1.asm

4 Выводы

Выполнив данную лабораторную работу я обрела теоретические и практические знания в NASM. Я научилась использовать команд условного и безусловного переходов, написать программ с использованием переходов и ознакомилась с назначением и структурой файла листинга.