Лабораторная работа №8.

Команды безусловного и условного переходов в Nasm.

Кучеренко София.

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить команды условного и безусловного переходов и научиться писать программы с использованием этих переходов.

2 Выполнение лабораторной работы

1. Создадим файл lab8-1.asm, запишем код программы и проверим его работу:

```
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ touch lab8-1.asm
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ nasm -f elf lab8-1.asm
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 3
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $
```

Рис. 2.1: Работа программы lab8-1

2. Создадим файл lab8-2.asm, запишем код программы и также проверим его работу:

```
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ touch lab8-2.asm
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ nasm -f elf lab8-2.asm
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ./lab8-2
Сообщение No 2
Сообщение No 1
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $
```

Рис. 2.2: Работа программы lab8-2

3. Изменим текст программы, изменив инструкции jmp и получим следующее:

```
1   %include 'in_out.asm' ; подключение внешнего файла
2   SECTION .data
3   msg1: DB 'Сообщение No 1',0
4   msg2: DB 'Сообщение No 2',0
5   msg3: DB 'Сообщение No 3',0
6   SECTION .text
7   GLOBAL _start
8   _start:
9   jmp _label3
10  _label1:
11  mov eax, msg1 ; Вывод на экран строки
12  call sprintLF ; 'Сообщение No 1'
13  jmp _end
14  _label2:
15  mov eax, msg2 ; Вывод на экран строки
16  call sprintLF ; 'Сообщение No 2'
17  jmp _label1
18  _label3:
19  mov eax, msg3 ; Вывод на экран строки
20  call sprintLF ; 'Сообщение No 3'
21  jmp _label2
22  _end:
23  call quit ; вызов подпрограммы завершения
```

Рис. 2.3: Измененный код

```
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ nasm -f elf lab8-2.asm
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ./lab8-2
Сообщение No 3
Сообщение No 2
Сообщение No 1
```

Рис. 2.4: Работа измененной программы

4. Откроем файл с программой lab8-1.asm и в любой инструкции с двумя операндами удалим один, выполним трансляцию с получением файла листинга (nasm -f elf -l lab8-2.lst lab8-2.asm):

```
 6                                  SECTION .text
 7                                  GLOBAL _start
 8                                  _start:
 9 000000E8 EB05                    jmp _label2
10                                  _label1:
11                                  mov eax, ; Вывод на экран строки
11                  ******************  error: invalid combination of opcode and operands
12 000000EA E83EFFFFFF              call sprintLF ; 'Сообщение No 1'
13                                  _label2:
14 000000EF B8[18000000]            mov eax, msg2 ; Вывод на экран строки
15 000000F4 E834FFFFFF              call sprintLF ; 'Сообщение No 2'
16                                  _label3:
17 000000F9 B8[30000000]            mov eax, msg3 ; Вывод на экран строки
18 000000FE E82AFFFFFF              call sprintLF ; 'Сообщение No 3'
19                                  _end:
20 00000103 E8D3FFFFFF              call quit ; вызов подпрограммы завершения
```

Рис. 2.5: Ошибка в программе

5. Создадим файл lab8-3.asm, запишем код программы и проверим его работу на разных значениях B:

```
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ touch lab8-3.asm
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ nasm -f elf lab8-3.asm
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ./lab8-3
Введите B: 3
Наибольшее число: 50
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ./lab8-3
Введите B: 5
Наибольшее число: 50
smkucherenko@dk3n66 ~/work/koshechki/study_2022-2023_arh-pc/labs/lab08 $ ./lab8-3
Введите B: 55
Наибольшее число: 55
```

Рис. 2.6: Результат выполнения программы

3 Самостоятельная работа

1. Напишем программу нахождения наименьшей из 3 целочисленных переменных и запустим её:

```
%include 'in_out.asm'

section .data
    msg1 db "Наименьшее число:"
    a dd 32
    b dd 46
    c dd 74

section .bss
    min resb 10

section .text
global _start

_start:
    mov eax, msg1
    call sprint

    mov ecx, [a]
    mov [min], ecx ; 'min = A'
    ; ---------- Сравниваем 'A' и 'C' (как числа)
    cmp ecx, [c] ; Сравниваем 'A' и 'C'
    jl 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:
    ; ---------- 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, [min]
    call iprintLF ; Вывод 'min(A,B,C)'
    call quit ; Выход
```

Рис. 3.1: Код программы

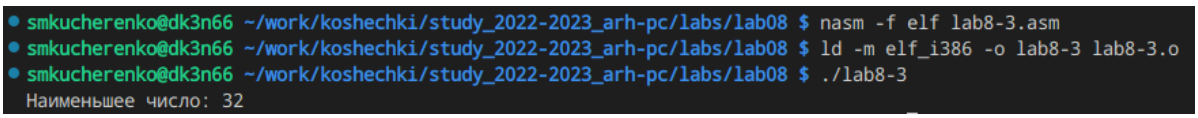

Рис. 3.2: Результат выполнения программы

2. Напишем программу для нахождения значения заданной функции из введенных с клавиатуры значений:

```
_start:
    ; ---------- Ввод 'X'
    mov eax, msgX
    call sprint
    mov ecx, x
    mov edx,10
    call sread

    ; ---------- Ввод 'A'
    mov eax, msgA
    call sprint
    mov ecx, a
    mov edx,10
    call sread

    ; ---------- Преобразование 'x' из символа в число
    mov eax, x
    call atoi
    mov [x], eax

    ; ---------- Преобразование 'a' из символа в число
    mov eax, a
    call atoi
    mov [a], eax


    mov ecx, [x]
    cmp ecx, [a]

    ja newfunc

    mov eax, [x]
    jmp fin

newfunc:
    mov eax, [a]
    mov ebx, [x]
    add eax, ebx

fin:
    call iprintLF
    call quit
```

Рис. 3.3: Код программы

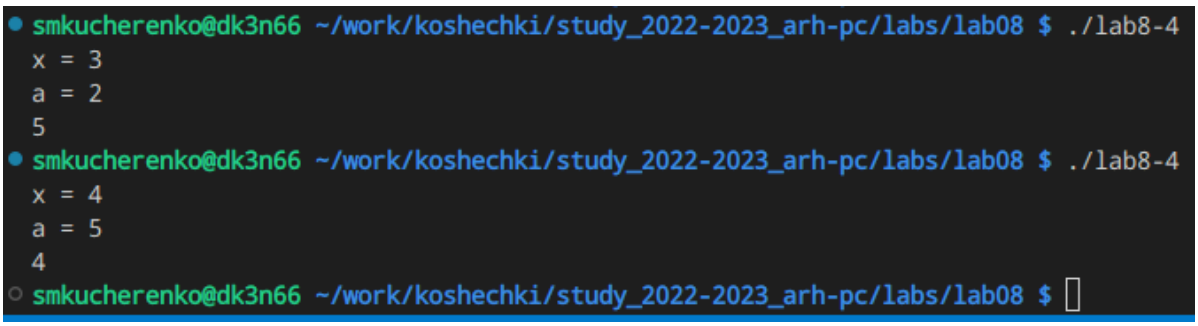

Рис. 3.4: Результат выполнения программы

4 Выводы

Я изучила команды условного и безусловного переходов и научилась писать программы с использованием этих переходов.