РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Архитектура компьютера

Студент: Семенченко Т. С.

Группа: НКАбд-05-25

МОСКВА

2025г.

Оглавление

1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Познакомиться с назначением и структурой файла листинга.

2 Задачи

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Задание для самостоятельной работы

3 Порядок выполнения лабораторной работы

3.1 Реализация переходов в NASM

Создала каталог для программам лабораторной работы № 7, перешла в него и создала файл lab7-1.asm (рис. 3.1.1)

```
tssemenchenko@dk6n18 ~ $ mkdir ~/work/arch-pc/lab07
tssemenchenko@dk6n18 ~ $ cd  ~/work/arch-pc/lab07
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ touch lab7-1.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ mc
```

Рис. 3.1.1 Демонстрация написанных команд

Ввела в файл lab7-1.asm текст программы из первого листинга. Создала исполняемый файл и запустила его (рис. 3.1.2)

```
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $
```

Рис. 3.1.2 Демонстрация работы программы

Изменила текст программы в соответствии со вторым листингом. Создала исполняемый файл и запустила его (рис. 3.1.3)

```
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.1.3 Демонстрация работы команды

Изменяю текст программы, чтобы результат работы программы выводил на экран все три сообщения в обратном порядке (рис. 3.1.4, 3.1.5)

```
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $
```

Рис. 3.1.4 Демонстрация работы программы

```
/afs/.dk.sci.pfu.edu.ru/ho~/arch-pc/lab07/l
%include 'in_out.asm'
 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
    call sprintLF
    jmp _end
  _label2:
    mov eax, msg2
    call sprintLF
    jmp _label1
  _label3:
    mov eax, msg3
    call sprintLF
    jmp _label2
  _end:
    call quit
```

Рис. 3.1.5 Измененный текст программы

Создала файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучила текст программы из третьего листинга и ввела его файл lab7-2.asm. Создала исполняемый файл и запустила его. Проверила работу программы на некотором значении B (рис. 3.1.6)

```
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ touch lab7-2.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 3
Наибольшее число: 50
```

Рис. 3.1.6 Демонстрация работы программы

3.2 Изучение структуры файлы листинга

Создала файл листинга для программы из файла lab7-2.asm (рис. 3.2.1)

```
menchenko@dk6n18 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 3.2.1 Создание файла

Открыла файл листинга lab7-2.lst с помощью текстового редактора mcedit (рис. 3.2.2)

```
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
25 00000011 53                    <1>     push    ebx
26 00000012 50                    <1>     push    eax
27 00000013 E8E8FFFFFF            <1>     call    slen
28                                <1>
29 00000018 89C2                  <1>     mov     edx, eax
30 0000001A 58                    <1>     pop     eax
31                                <1>
32 0000001B 89C1                  <1>     mov     ecx, eax
33 0000001D BB01000000            <1>     mov     ebx, 1
34 00000022 B804000000            <1>     mov     eax, 4
35 00000027 CD80                  <1>     int     80h
```

Рис. 3.2.2 Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удалила один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 3.2.3)

```
lab7-2.asm          [-M--] 12 L:[  1+17  1
%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:
    mov eax,msg1
    call sprint
    mov ecx,B
    mov edx,10
    call sread
    mov eax,
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [max],ecx
    cmp ecx,[C]
```

Рис. 3.2.3 Удаление одного операнда

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 3.2.4)

```
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands
tssemenchenko@dk6n18 ~/work/arch-pc/lab07 $
```

Рис. 3.2.4 Демонстрация ошибки

4 Задания для самостоятельной работы

4.1 т. к. в ходе 7 лабораторной не было получено нового варианта, я

использовала вариант, полученных в 6 лабораторной работе – 10.

Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 4.1.1)

```
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $ ./lab7-3
Введите A: 41
Введите B: 62
Введите C: 35
Наименьшее число: 35
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $
```

Рис. 4.1.1 Демонстрация работы программы

Прикладываю код программы:

```
%include 'in_out.asm'
section .data
    msg_b db 'Введите B: ',0h
    msg_a db 'Введите A: ',0h
    msg_c db 'Введите C: ',0h
    msg_result db "Наименьшее число: ",0h
section .bss
    min resb 10
    B    resb 10
    A    resb 10
    C    resb 10
section .text
    global _start
_start:
    mov eax,msg_a
    call sprint
    mov ecx,A
    mov edx,10
    call sread
    mov eax, A
    call atoi
    mov [A], eax
```

```
    mov eax, msg_b
    call sprint
    mov ecx, B
    mov edx, 10
    call sread
    mov eax, B
    call atoi
    mov [B], eax
    mov eax, msg_c
    call sprint
    mov ecx, C
    mov edx, 10
    call sread
    mov eax, C
    call atoi
    mov [C], eax
    mov eax, [A]
    cmp eax, [B]
    jl check_c
    mov eax, [B]
check_c:
    cmp eax, [C]
    jl print_min
    mov eax, [C]
print_min:
    mov [min], eax
    mov eax, msg_result
    call sprint
    mov eax, [min]
    call iprintLF
    call quit
```

4.2 Написала программу, которая для введенных с клавиатуры значений x и a

вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Создала файл lab7-4.asm для работы над вторым заданием.

Создала исполняемый файл и проверила его работу для значений x и a из 6 листинга. (рис. 4.2.1)

```
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $ touch lab7-4.asm

tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 3
Введите a: 0
Результат: 1
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 1
Введите a: 2
Результат: 6
tssemenchenko@dk7n07 ~/work/arch-pc/lab07 $
```

Рис. 4.2.1 Демонстрация выполненный команд и проверка работы программы

Прикладываю код программы:

```
%include 'in_out.asm'
section .data
   msg_x db "Введите x: ",0
   msg_a db "Введите a: ",0
   msg_res db "Результат: ",0
section .bss
   x resb 10
   a resb 10
section .text
global _start
 _start:
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 10
    call sread
    mov eax, x
```

```
    call atoi
    mov [x], eax
    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 10
    call sread
    mov eax, a
    call atoi
    mov [a], eax
    mov ebx, [x]
    cmp ebx, 2
    jg branch_x_gt_2
    mov eax, [a]
    mov ebx, 3
    mul ebx
    jmp print_result
branch_x_gt_2:
   mov eax, [x]
   sub eax, 2
print_result:
   mov edi, eax
   mov eax, msg_res
   call sprint
   mov eax, edi
   call iprintLF
   call quit
```

5 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.