

Отчёт по лабораторной работе №8

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

Кочина Дарья Сергеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	22

Список иллюстраций

4.1	Создание файла lab8-1.asm	7
4.2	Текст программы из листинга	8
4.3	Результат работы программы	8
4.4	Текст программы из листинга	9
4.5	Результат работы программы	9
4.6	Текст программы из листинга	10
4.7	Результат работы программы	10
4.8	Создание файла lab8-2.asm	10
4.9	Текст программы из листинга	11
4.10	Текст программы из листинга	12
4.11	Результат работы программы	12
4.12	Создание файла листинга для программы	13
4.13	Открытие файла в редакторе	13
4.14	Удаление одного операнда	14
4.15	Создание файла без одного операнда	14
4.16	Файл листинга без одного операнда	14
4.17	Объяснение первой строки	14
4.18	Объяснение второй строки	14
4.19	Объяснение третьей строки	15
4.20	Текст программы в файле lab8-3.asm	16
4.21	Результат работы программы	17
4.22	Результат работы программы	18
4.23	Текст программы в файле lab8-4.asm	19
4.24	Результат работы программы	20
4.25	Результат работы программы	21

1 Цель работы

Целью данной лабораторной работы является изучение команд условного и безусловного переходов. А также приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

Изучить команды условного и безусловного переходов, приобрести практические навыки написания программ с использованием переходов и ознакомиться с назначением и структурой файла листинга.

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.

безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Структура листинга:

номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);

адрес — это смещение машинного кода от начала текущего сегмента;

машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности;

исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

4 Выполнение лабораторной работы

1. Я создала каталог для программ лабораторной работы №8, перешла в него и создала файл lab8-1.asm. (рис. 4.1)

```
dskochina@dk8n63 ~ $ mkdir ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab08
dskochina@dk8n63 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab08
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ touch lab8-1.asm
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $
```

Рис. 4.1: Создание файла lab8-1.asm

2. Рассмотрела пример программы с использованием инструкции jmp. Для этого я ввела в файл lab8-1.asm текст программы из листинга. (рис. 4.2)

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Текст программы из листинга

3. Я создала исполняемый файл и запустила его. Результат работы данной программы был следующим. (рис. 4.3)

```
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-1.asm
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 3
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $
```

Рис. 4.3: Результат работы программы

4. Я изменила программу таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу. Для этого в текст программы после вывода сообщения No 2 добавила инструкцию `jmp` с меткой `_label1` и после вывода сообщения No 1 добавила инструкцию `jmp` с меткой `_end`. Я изменила текст программы в соответствии с листингом. (рис. 4.4, 4.5)


```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 4.4: Текст программы из листинга

```

dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-1.asm
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 1
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ 

```

Рис. 4.5: Результат работы программы

5. Я изменила текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим. (рис. 4.6, 4.7)

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Текст программы из листинга

```
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-1.asm
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $
```

Рис. 4.7: Результат работы программы

6. Я создала файл lab8-2.asm. Внимательно изучила текст программы из листинга (программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С). Я ввела его в lab8-2.asm. (рис. 4.8, 4.9, 4.10)

```
dskochina@dk8n63 ~ $ cd ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ touch lab8-2.asm
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $
```

Рис. 4.8: Создание файла lab8-2.asm

```

#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

```

Рис. 4.9: Текст программы из листинга

```

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
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Рис. 4.10: Текст программы из листинга

7. Я создала исполняемый файл и проверила его работу для разных значений В. (рис. 4.11)

```

dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-2.asm
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-2
Введите B: 1
Наибольшее число: 50
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-2
Введите B: 5
Наибольшее число: 50
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-2
Введите B: 55
Наибольшее число: 55
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-2
Введите B: 20
Наибольшее число: 50
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-2
Введите B: 21
Наибольшее число: 50
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-2
Введите B: 50
Наибольшее число: 50
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-2
Введите B: 51
Наибольшее число: 51
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ 

```

Рис. 4.11: Результат работы программы

8. Я создала файл листинга для программы из файла lab8-2.asm, указав ключ -l и задав имя файла листинга в командной строке. Затем я открыла файл листинга lab8-2.lst с помощью текстового редактора mcedit. Внимательно ознакомилась с его форматом и содержанием. (рис. 4.12, 4.13)

```

dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf -l lab8-2.lst lab8-2.asm
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ mcedit lab8-2.lst
dskochina@dk8n63 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ 

```

Рис. 4.12: Создание файла листинга для программы

```

lab8-2.lst  [----]  0 L: 1+ 0 1/225] *(0 /14458b) 0032 0x020
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 E8E8FFFF <1>      call    slen
29     <1>.....
30     00000018 89C2     <1>      mov     edx, eax
31     0000001A 58      <1>      pop     eax
32     <1>.....
33     0000001B 89C1     <1>      mov     ecx, eax
34     0000001D BB01000000 <1>      mov     ebx, 1
35     00000022 B804000000 <1>      mov     eax, 4
36     00000027 CD80     <1>      int     80h

```

Рис. 4.13: Открытие файла в редакторе

- Я открыла файл с программой lab8-2.asm и в строке mov eax,тах я убрала тах и попробовала создать файл. Затем я выполнила трансляцию с получением файла листинга. Выдало ошибку, поскольку для программы нужно два операнда. (рис. 4.14, 4.15)

```
check_B:
mov eax,
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
```

Рис. 4.14: Удаление одного операнда

```
dskochina@dk8n70 ~ $ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-pc/lab08
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:34: error: invalid combination of opcode and operands
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $
```

Рис. 4.15: Создание файла без одного операнда

10. В файле листинга можно увидеть, где именно ошибка и с чем она связана. (рис. 4.16)

```
34                                     mov eax
34      *****                      error: invalid combination of opcode and operands
35 00000130 E867FFFFFF                call atoi ; Вызов подпрограммы перевода символа в число
36 00000135 A3[00000000]              mov [max],eax ; запись преобразованного числа в 'max'
```

Рис. 4.16: Файл листинга без одного операнда

11. Эта строка находится на 35 месте, её адрес 00000130, машинный код - E867FFFFFF, исходный текст программы - call atoi. Он означает, что символ, лежащий в строке выше, переводится в число. (рис. 4.17)

```
35 00000130 E867FFFFFF                call atoi ; Вызов подпрограммы перевода символа в число
```

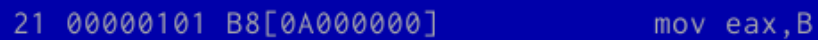
Рис. 4.17: Объяснение первой строки

12. Эта строка находится на 47 месте, её адрес 0000012E, машинный код - A1[00000000], исходный текст программы - mov eax,[max]. Он означает, что число, хранящееся в переменной max, записывается в eax. (рис. 4.18)

```
47 0000012E A1[00000000]              mov eax,[max]
```

Рис. 4.18: Объяснение второй строки

13. Эта строка находится на 21 месте, её адрес 00000101, машинный код - B8[0A000000], исходный текст программы - mov eax,B. Он означает, что в регистр eax мы вносим значения переменной B. (рис. 4.19)



```
21 00000101 B8[0A000000] mov eax,B
```

Рис. 4.19: Объяснение третьей строки

Самостоятельная работа

1. Я написала программу нахождения наименьшей из 3 целочисленных переменных a,b,c. Значения переменных я выбрала из таблицы (81,22,72) в соответствии с вариантом, полученным при выполнении лабораторной работы №7 (вариант 14). Я создала исполняемый файл и проверила его работу. (рис. 4.20, 4.21, 4.22)

```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '81'
C dd '72'

section .bss
min 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,B
call atoi
mov [B],eax

mov ecx,[A]
mov [min],ecx

cmp ecx,[C]
jb check_B
mov ecx,[C]
mov [min],ecx

```

Рис. 4.20: Текст программы в файле lab8-3.asm


```
check_B:
mov  eax,min
call atoi
mov  [min],eax

mov  ecx,[min]
cmp  ecx,[B]
jb  fin
mov  ecx,[B]
mov  [min],ecx

fin:
mov  eax, msg2
call sprint
mov  eax,[min]
call iprintLF
call quit
```



Рис. 4.21: Результат работы программы

```
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-3.asm
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-3
Введите B: 22
Наименьшее число: 22
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $
```

Рис. 4.22: Результат работы программы

2. Я написала программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ я выбрала из таблицы вариантов заданий в соответствии с вариантом, полученным при выполнении работы №7 (вариант 14). Я создала исполняемый файл и проверила его работу для значений x и a . (рис. 4.23, 4.24, 4.25)

```

%include 'in_out.asm'
SECTION .data
msg1 db 'Введите X: ',0h
msg2 db 'Введите a: ',0h
msg3 db 'Ответ: ',0h
SECTION .bss
x resb 10
a resb 10
o resb 10
SECTION .text
global _start
_start:
mov eax,msg1
call sprint

mov ecx,x
mov edx,10
call sread

mov eax,x
call atoi
mov [x],eax

mov eax,msg2
call sprint

mov ecx,a
mov edx,10
call sread

mov eax,a
call atoi
mov [a],eax

```

Рис. 4.23: Текст программы в файле lab8-4.asm

```
mov eax,a
call atoi
mov [a],eax

mov ecx, [x]
mov [o],ecx

mov ebx,3
cmp ecx,[a]
jnl fin
mov eax,[x]
mul ebx
add eax,1
mov [o], eax
jmp otv

fin:
mov eax,[a]
mul ebx
add eax,1
mov [o],eax
otv:
mov eax,msg3
call sprint
mov eax,[o]
call iprintLF
call quit
```

Рис. 4.24: Результат работы программы

```
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-4.asm
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-4
Введите X: 2
Введите a: 3
Ответ: 10
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $ ./lab8-4
Введите X: 4
Введите a: 2
Ответ: 13
dskochina@dk8n70 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab08 $
```

Рис. 4.25: Результат работы программы

5 Выводы

В ходе выполнения данной лабораторной работы я изучила команды условного и безусловного переходов. А также приобрела навыки написания программ с использованием переходов и ознакомилась с назначением и структурой файла листинга.