

Лабораторная работа №8. Команды безусловного и условного переходов в Nasm. Программирование ветвлений.

Дисциплина: Архитектура компьютера

Алексей Назаров НММбд-02-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Создадим каталог для лабораторной и файл lab8-1.asm	6
2.2	Введем в файл lab8-1.asm текст из листинга	7
2.3	Изменим код, как показано в листинге 8.2	8
2.4	Изменим код, что бы программа выводила 3, 2, 1	8
2.5	Создадим lab8-2 и откроем в Gedit	9
2.6	Изучим листинг	11
2.6.1	Описание трех строчек листинга	12
2.7	Изменим строчку в коде и создадим новый листинг	12
3	Задания для самостоятельной работы	13
4	Выводы	17
	Список литературы	18

Список иллюстраций

2.1	Создадим файл и папку	6
2.2	Код lab8-1	7
2.3	Запуск lab8-1	7
2.4	Измененный lab8-1	8
2.5	Запуск lab8-1	8
2.6	Измененный lab8-1	9
2.7	Вывод измененного lab8-1	9
2.8	Создание и открытие lab8-2 в текстовом редакторе	9
2.9	Код lab8-2	10
2.10	Исполним программу несколько раз	11
2.11	Создание и открытие листинга в тестовом редакторе	11
2.12	Просмотр листинга в GEDIT	11
2.13	Изменение lab8-2	12
2.14	Вывод при генерации листинга из кода с ошибкой	12
2.15	Просмотр листинга	12
3.1	Код lab8-3.asm	13
3.2	Вывод lab8-3.asm	14
3.3	Функция 13	14
3.4	Код lab8-4.asm	15
3.5	Проверка lab8-4.asm	16

Список таблиц

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файлов листинга

2 Выполнение лабораторной работы

2.1 Создадим каталог для лабораторной и файл lab8-1.asm

```
[amnazarov@localhost ~]$ mkdir ~/work/arch-pc/lab08  
cd ~/work/arch-pc/lab08  
touch lab8-1.asm  
[amnazarov@localhost lab08]$ ls  
lab8-1.asm  
[amnazarov@localhost lab08]$
```

Рис. 2.1: Создадим файл и папку

2.2 Введем в файл lab8-1.asm текст из листинга

```
1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3 DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10
11 _start:
12     jmp _label3
13 _label1:
14     mov eax, msg1 ; Вывод на экран строки
15     call sprintf ; 'Сообщение № 1'
16
17     jmp _end ;Прыжок к выходу
18
19 _label2:
20     mov eax, msg2 ; Вывод на экран строки
21     call sprintf ; 'Сообщение № 2'
22
23     jmp _label1 ; прыжок к выводу первого сообщения
24 _label3:
25     mov eax, msg3 ; Вывод на экран строки
26     call sprintf ; 'Сообщение № 3'
27
28     jmp _label2
29 _end:
30     call quit ; вызов подпрограммы завершения
31
32
```

Рис. 2.2: Код lab8-1

Запустим программу и посмотрим результат

```
[amnazarov@localhost lab08]$ nasm -f elf lab8-1.asm
[amnazarov@localhost lab08]$ ld -m elf_i386 lab8-1.o -o lab8-1
[amnazarov@localhost lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[amnazarov@localhost lab08]$
```

Рис. 2.3: Запуск lab8-1

2.3 Изменим код, как показано в листинге 8.2

```
1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3 DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10
11 _start:
12     jmp _label2
13 _label1:
14     mov eax, msg1 ; Вывод на экран строки
15     call sprintf ; 'Сообщение № 1'
16
17     jmp _end ;Прыжок к выходу
18
19 _label2:
20     mov eax, msg2 ; Вывод на экран строки
21     call sprintf ; 'Сообщение № 2'
22     jmp _label1
23
24     jmp _label1 ; прыжок к выводу первого сообщения
25
26 _label3:
27     mov eax, msg3 ; Вывод на экран строки
28     call sprintf ; 'Сообщение № 3'
29
30     jmp _label2
31 _end:
32     call quit ; вызов подпрограммы завершения
33
```

Рис. 2.4: Измененный lab8-1

Запустим программу и посмотрим на измененный результат

```
[amnazarov@localhost lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[amnazarov@localhost lab08]$
```

Рис. 2.5: Запуск lab8-1

2.4 Изменим код, что бы программа выводила 3, 2, 1

Нам достаточно перепрыгивать на label3 в начале, и в конце вывода “3” прыгать на label2, а дальше программа выполняется так как и раньше


```

11 _start:
12         jmp _label3
13 label1:

```

Рис. 2.6: Измененный lab8-1

Запустим программу и посмотрим на результат

```

[amnazarov@localhost lab08]$ nasm -f elf lab8-1.asm
[amnazarov@localhost lab08]$ ld -m elf_i386 lab8-1.o -o lab8-1
[amnazarov@localhost lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[amnazarov@localhost lab08]$

```

Рис. 2.7: Вывод измененного lab8-1

2.5 Создадим lab8-2 и откроем в Gedit

```

[amnazarov@localhost lab08]$ touch lab8-2.asm
[amnazarov@localhost lab08]$ gedit lab8-2.asm

```

Рис. 2.8: Создание и открытие lab8-2 в текстовом редакторе

Введем код в lab8-2

```

1 %include 'in_out.asm'
2 section .data
3
4     msg1 db 'Введите B: ',0h
5     msg2 db "Наибольшее число: ",0h
6     A dd '20'
7     C dd '50'
8
9 section .bss
10    max resb 10
11    B resb 10
12
13 section .text
14
15
16 global _start
17 _start:
18
19 ; ----- Вывод сообщения 'Введите B: '
20     mov eax,msg1
21     call sprint
22
23 ; ----- Ввод 'B'
24     mov ecx,B
25     mov edx,10
26     call sread
27
28 ; ----- Преобразование 'B' из символа в число
29     mov eax,B
30     call atoi ; Вызов подпрограммы перевода символа в число
31     mov [B],eax ; запись преобразованного числа в 'B'
32
33 ; ----- Записываем 'A' в переменную 'max'
34     mov ecx,[A] ; 'ecx = A'
35
36     mov [max],ecx ; 'max = A'
37 ; ----- Сравниваем 'A' и 'C' (как символы)
38
39     cmp ecx,[C] ; Сравниваем 'A' и 'C'
40     jg check_B ; если 'A>C', то переход на метку 'check_B',
41     mov ecx,[C] ; иначе 'ecx = C'
42     mov [max],ecx ; 'max = C'
43
44 ; ----- Преобразование 'max(A,C)' из символа в число
45 check_B:

```

Рис. 2.9: Код lab8-2

Скомпилируем и исполним программу несколько раз

```

[amnazarov@localhost lab08]$ nasm -f elf lab8-1.asm
[amnazarov@localhost lab08]$ nasm -f elf lab8-2.asm
[amnazarov@localhost lab08]$ ld -m elf_i386 lab8-2.o -o lab8-2
[amnazarov@localhost lab08]$ ./lab8-2
Введите B: 33
Наибольшее число: 50
[amnazarov@localhost lab08]$ ./lab8-2
Введите B: 9
Наибольшее число: 50
[amnazarov@localhost lab08]$ ./lab8-2
Введите B: 434
Наибольшее число: 434
[amnazarov@localhost lab08]$ ./lab8-2
Введите B: 11111
Наибольшее число: 11111
[amnazarov@localhost lab08]$

```

Рис. 2.10: Исполним программу несколько раз

2.6 Изучим листинг

Сгенерируем листинг

```

[amnazarov@localhost lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
[amnazarov@localhost lab08]$ gedit lab8-2.lst

```

Рис. 2.11: Создание и открытие листинга в тестовом редакторе

```

1 1
2 2
3 3
4 4
5 5 00000000 53
6 6 00000001 89C3
7 7
8 8
9 9 00000003 803800
10 10 00000006 7403
11 11 00000008 40
12 12 00000009 EBF8
13 13
14 14
15 15 0000000B 2908
16 16 0000000D 5B
17 17 0000000E C3
18 18
19 19
20 20
21 21
22 22
23 23
24 24 0000000F 52
25 25 00000010 51
26 26 00000011 53
27 27 00000012 50
28 28 00000013 E8E8FFFFFF
29 29
30 30 00000018 89C2
31 31 0000001A 5B
32 32
33 33 0000001B 89C1
34 34 0000001D 8B01000000
35 35 00000022 B804000000
36 36 00000027 CD00
37 37
38 38 00000029 5B
39 39 0000002A 59
40 40 0000002B 5A
41 41 0000002C C3
42 42
43 43
44 44
45 45
46 46
47 47
48 48 0000002D E8D0FFFFFF
49 49
50 50 00000032 50
51 51 00000033 B80A000000
52 52 00000038 50
53 53 00000039 89E0

```

Рис. 2.12: Просмотр листинга в GEDIT

2.6.1 Описание трех строчек листинга

- На строке 25 видим номер строки (25), адрес инструкции в памяти (00000010) и код команды (51)
- Сравним строкит 5 и 16
- Адрес инструкции на строке 5 - 00000000
- Адрес инструкции на строке 16 000000D
- Код инструкции на строке 5 - 53
- Код инструкции на строке 16 5B
- Из этого можем сделать вывод, что код для ebx - 3, потому что $50 + 3 = 53$, а $58 + 3 = 5B$, так как код push 50, а pop 58

2.7 Изменим строчку в коде и создадим новый листинг

вместо `cmp ecx, [C]` оставим `cmp ecx,`

```
38
39      cmp ecx, ; Сравниваем 'A' и 'C'
```

Рис. 2.13: Изменение lab8-2

Сгенерируем листинг

```
[amnazarov@localhost lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:39: error: invalid combination of opcode and operands
[amnazarov@localhost lab08]$ gedit lab8-2.lst
```

Рис. 2.14: Вывод при генерации листинга из кода с ошибкой

Откроем созданный файл листинга

```
213      38      cmp ecx, ; Сравниваем 'A' и 'C'
214      39
215      39      ***** error: invalid combination of opcode and operands
216      40 0000011C 7F0C      jg check_B ; если 'A>C', то переход на метку 'check_B',
```

Рис. 2.15: Просмотр листинга

Видим, что выведенная при генерации ошибка находится в файле листинга, после измененной строчки

3 Задания для самостоятельной работы

Напишем код, который будет выводить наименьшее число из 84, 32, 77 (Так как мой номер 13)

```
1 include 'in_out.asm'
2 section .data
3
4     msg1 db 'Введите B: ',0h
5     msg2 db "Наименьшее число: ",0h
6     A dd '84'
7     C dd '32'
8     B dd '77'
9
10 section .bss
11     min resb 10
12
13 section .text
14
15 global _start
16 _start:
17
18 ; ----- Преобразование 'B' из символа в число
19     mov eax,B
20     call atoi ; Вызов подпрограммы перевода символа в число
21     mov [B],eax ; запись преобразованного числа в 'B'
22
23 ; ----- Записываем 'A' в переменную 'min'
24     mov ecx,[A] ; 'ecx = A'
25
26     mov [min],ecx ; 'min = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28
29     cmp ecx, [C] ; Сравниваем 'A' и 'C'
30     jl check_B ; если 'A < C', то переход на метку 'check_B',
31     mov ecx,[C] ; иначе 'ecx = C'
32     mov [min],ecx ; 'min = C'
33
34 ; ----- Преобразование 'min(A,C)' из символа в число
35 check_B:
36     mov eax,min
37     call atoi ; Вызов подпрограммы перевода символа в число
38     mov [min],eax ; запись преобразованного числа в 'min'
39 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
40     mov ecx,[min]
41     cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
42     jl fin ; если 'min(A,C) < B', то переход на 'fin',
43     mov ecx,[B] ; иначе 'ecx = B'
44     mov [min],ecx
45 ; ----- Вывод результата
46 fin:
47     mov eax, msg2
48     call sprint ; Вывод сообщения 'Наименьшее число: '
49     mov eax,[min]
50     call iprintf ; Вывод 'min(A,B,C)'
51     call quit
52 ; Выход
53
```

Рис. 3.1: Код lab8-3.asm

Запустим программу и убедимся в правильности результата

```
[amnazarov@localhost lab08]$ ./lab8-3
Наименьшее число: 32
[amnazarov@localhost lab08]$
```

Рис. 3.2: Вывод lab8-3.asm

Напишем программу, которая будет вычислять функцию

$$13 \quad \begin{cases} a - 7, & a \geq 7 \\ ax, & a < 7 \end{cases} \quad (3;9) \quad (6;4)$$

Рис. 3.3: Функция 13

```

2
3 section .data
4 msg1 db "x: ",0h
5 msg2 db "a: ",0h
6 msg3 db "Ответ: ",0h
7 section .bss
8 A resb 20
9 X resb 20
10 ans resb 20
11 section .text
12
13 GLOBAL _start
14
15 _start:
16
17     mov eax, msg1
18     call sprintf ; выводим msg1
19
20     mov ecx, X
21     mov edx, 20
22     call sread
23     mov eax, X
24     call atoi
25     mov [X], eax ; Записываем ввод в виде числа в X
26
27     mov eax, msg2
28     call sprintf ; Выводим msg2
29
30     mov ecx, A
31     mov edx, 20
32     call sread
33     mov eax, A
34     call atoi
35     mov [A], eax ; Записываем ввод в виде числа в A
36
37 .else:
38
39     xor ecx, ecx
40     mov ecx, [A]
41     cmp ecx, 7
42     jge .if ; Если A больше 7 переходим на .if
43
44     xor eax, eax
45     mov eax, [A]
46
47     mul byte [X] ; Умножаем A на X
48
49     je .final
50
51 .if:
52     xor eax, eax
53     mov eax, [A]
54     sub eax, 7 ; A - 7
55
56 .final:
57
58     push eax
59     mov eax, msg3
60     call sprintf

```

Рис. 3.4: Код lab8-4.asm

Запустим ее и проверим на данных значениях

```
[amnazarov@localhost lab08]$ nasm -f elf lab8-4.asm
[amnazarov@localhost lab08]$ ld -m elf_i386 lab8-4.o -o lab8-4
[amnazarov@localhost lab08]$ ./lab8-4
x:
3
a:
9
Ответ:
2
[amnazarov@localhost lab08]$ ./lab8-4
x:
6
a:
4
Ответ:
24
[amnazarov@localhost lab08]$ gedit lab8-4.asm
```

Рис. 3.5: Проверка lab8-4.asm

4 Выводы

Мы изучили команды условного и безусловного переходов в языке асембела NASM, научились писать программы с использованием переходов и познакомились с назначением и структурой файлов листинга

Список литературы