

# **Лабораторная работа №8. Команды безусловного и условного переходов**

**Архитектура ЭВМ**

Плескачева Елизавета Андреевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация переходов в NASM . . . . .	6
2.2	Программа выводящая наибольшее число . . . . .	9
2.3	Изучение листинга . . . . .	11
2.3.1	Объяснение содержимого файлов листинга . . . . .	12
2.3.2	Создадим ошибку в программе . . . . .	12
<b>3</b>	<b>Задание для самостоятельной работы</b>	<b>14</b>
3.1	Программа, находящая наименьшее . . . . .	14
<b>4</b>	<b>Программа, вычисляющая значение функции</b>	<b>16</b>
<b>5</b>	<b>Выводы</b>	<b>19</b>

# Список иллюстраций

2.1	Создание папки и файла . . . . .	6
2.2	Текст из листинга 8.1 . . . . .	7
2.3	Запуск программы lab8-1 . . . . .	7
2.4	Измененная часть программы lab8-1.asm . . . . .	8
2.5	Вывод измененной программы lab8-1.asm . . . . .	8
2.6	Измененная часть программы lab8-1.asm . . . . .	9
2.7	Вывод измененной программы . . . . .	9
2.8	Содержимое lab8-2.asm . . . . .	10
2.9	Вывод скомпилированной программы . . . . .	10
2.10	Создание листинга для lab8-2 . . . . .	11
2.11	Просмотр файла листинга . . . . .	11
2.12	Файл листинга 2 . . . . .	12
2.13	Изменение файла lab8-2.asm . . . . .	13
2.14	Создание листинга для измененной программы . . . . .	13
2.15	Просмотр листинга . . . . .	13
3.1	Написанная программа по нахождению минимального . . . . .	15
3.2	Запуск программы по вычислению минимума . . . . .	15
4.1	Написанная программа . . . . .	17
4.2	Проверка выполнения программы . . . . .	18

## Список таблиц

# **1 Цель работы**

## 2 Выполнение лабораторной работы

### 2.1 Реализация переходов в NASM

Создадим папку и перейдем в нее.

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

Рис. 2.1: Создание папки и файла

Введем в lab8-1.asm текст из листинга 8.1

```

1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4 div: DB 'Результат: ',0
5 rem: DB 'Остаток от деления: ',0
6
7 SECTION .text
8 GLOBAL _start
9 _start: ; ---- Вычисление выражения
10 mov eax,5 ; EAX=5
11 mov ebx,2 ; EBX=2
12 mul ebx ; EAX=EAX*EBX
13
14
15 add eax,3 ; EAX=EAX+3
16 xor edx,edx ; обнуляем EDX для корректной работы div
17 mov ebx,3 ; EBX=3
18 div ebx ; EAX=EAX/3, EDX=остаток от деления
19
20 mov edi,eax ; запись результата вычисления в 'edi'
21
22 ; ---- Вывод результата на экран
23
24 mov eax,div ; вызов подпрограммы печати
25 call sprint ; сообщения 'Результат: '
26 mov eax,edi ; вызов подпрограммы печати значения
27 call iprintLF ; из 'edi' в виде символов
28
29 mov eax,rem ; вызов подпрограммы печати
30 call sprint ; сообщения 'Остаток от деления: '
31 mov eax,edx ; вызов подпрограммы печати значения
32 call iprintLF ; из 'edx' (остаток) в виде символов
33
34 call quit ; вызов подпрограммы завершения
35
36

```

Рис. 2.2: Текст из листинга 8.1

Скомпилируем и запустим программу

```

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

```

Рис. 2.3: Запуск программы lab8-1

Программа вывела сообщение 2 и 3

Изменим программу так, что бы она выводила сообщения 2 и 1

```

11 _start:
12     jmp _label2
13 _label1:
14     mov eax, msg1 ; Вывод на экран строки
15     call sprintf ; 'Сообщение № 1'
16
17     jmp _end
18
19 _label2:
20     mov eax, msg2 ; Вывод на экран строки
21     call sprintf ; 'Сообщение № 2'
22
23     jmp _label1
24
25 _label3:
26     mov eax, msg3 ; Вывод на экран строки
27     call sprintf ; 'Сообщение № 3'
28

```

Рис. 2.4: Измененная часть программы lab8-1.asm

Запустим измененную программу

```

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

```

Рис. 2.5: Вывод измененной программы lab8-1.asm

Программа вывела вначале 2 потом 1

Теперь изменим программу так, что бы она выводила сообщения 3 2 1

Для этого в самом начале программа переносится на вывод сообщения 3, из сообщения 3 к сообщению 2 оттуда к сообщению 1, а сообщение 1 переносит нас на выход



```

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
18
19 _label2:
20     mov eax, msg2 ; Вывод на экран строки
21     call sprintf ; 'Сообщение № 2'
22
23     jmp _label1
24
25 _label3:
26     mov eax, msg3 ; Вывод на экран строки
27     call sprintf ; 'Сообщение № 3'
28
29     jmp _label2
30 end.

```

Рис. 2.6: Измененная часть программы lab8-1.asm

Запустим программу

```

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

```

Рис. 2.7: Вывод изменной программы

Программа выводит сообщения в обратном порядке

## 2.2 Программа выводящая наибольшее число

Скопируем код из листинга 8.3 в lab8-2.asm

```

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:
46     mov eax,max
47     call atoi ; Вызов подпрограммы перевода символа в число
48     mov [max],eax ; запись преобразованного числа в 'max'
49 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
50     mov ecx,[max]
51     cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
52     jg fin ; если 'max(A,C)>B', то переход на 'fin',
53     mov ecx,[B] ; иначе 'ecx = B'
54     mov [max],ecx
55 ; ----- Вывод результата
56 fin:
57     mov eax, msg2
58     call sprint ; Вывод сообщения 'Наибольшее число: '
59     mov eax,[max]
60     call iprintLF ; Вывод 'max(A,B,C)'

```

Рис. 2.8: Содержимое lab8-2.asm

Запустим программу

```

[eapleskacheva@localhost lab08]$ nasm -f elf ./lab8-2.asm
ld -m elf_i386 -o ./lab8-2 ./lab8-2.o
./lab8-2
Введите B: 4
Наибольшее число: 50
[eapleskacheva@localhost lab08]$ ./lab8-2
Введите B: 23213
Наибольшее число: 23213
[eapleskacheva@localhost lab08]$ 

```

Рис. 2.9: Вывод сокмпилированной программы

Программа верно выводит наибольшие числа

## 2.3 Изучение листинга

Создадим листинг программы lab8-2.asm

```
apple@kali:~/lab8$ nasm -f elf -l lab8-2.lst lab8-2.asm
apple@kali:~/lab8$
```

Рис. 2.10: Создание листинга для lab8-2

Откроем файл листинга в gedit

```
193 18 ; ----- Вывод сообщения 'Введите B: '
194 19 ; -----
195 20 000000E8 B8[00000000] mov eax,msg1
196 21 000000ED E81DFFFFFF call sprint
197 22
198 23 ; ----- Ввод 'B'
199 24 000000F2 B9[0A000000] mov ecx,B
200 25 000000F7 BA0A000000 mov edx,10
201 26 000000FC E842FFFFFF call sread
202 27
203 28 ; ----- Преобразование 'B' из символа в число
204 29 00000101 B8[0A000000] mov eax,B
205 30 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
206 31 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
207 32
208 33 ; ----- Записываем 'A' в переменную 'max'
209 34 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
210 35
211 36 00000116 890D[00000000] mov [max],ecx ; 'max = A'
212 37 ; ----- Сравниваем 'A' и 'C' (как символы)
213 38
214 39 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
215 40 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
216 41 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
217 42 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
218 43
219 44 ; ----- Преобразование 'max(A,C)' из символа в число
220 45 check_B:
221 46 00000130 B8[00000000] mov eax,max
222 47 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
223 48 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
224 49 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
225 50 0000013F 8B0D[00000000] mov ecx,[max]
226 51 00000145 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
227 52 0000014B 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
228 53 0000014D 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
229 54 00000153 890D[00000000] mov [max],ecx
230 55 ; ----- Вывод результата
231 56 fin:
232 57 00000159 B8[13000000] mov eax,msg2
233 58 0000015E E8ACFFFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
234 59 00000163 A1[00000000] mov eax,[max]
235 60 00000168 E819FFFFFF call iprintLF ; Вывод 'max(A,B,C)'
236 61 0000016D E869FFFFFF call quit
237 62 ; Выход
238 63
```

Рис. 2.11: Просмотр файла листинга

```

1      1      %include 'in_out.asm'
2      2
3      3      <1> ;----- slen -----
4      4      <1> ; Функция вычисления длины сообщения
5      5      <1> slen:
6      6      00000000 53      <1> push    ebx
7      7      00000001 89C3    <1> mov     ebx, eax
8      8
9      9      00000003 803800    <1> nextchar:
10     10      00000006 7403      <1> cmp     byte [eax], 0
11     11      00000008 40        <1> jz      finished
12     12      00000009 EBF8      <1> inc     eax
13     13      0000000A 00        <1> jmp     nextchar
14     14
15     15      0000000B 29D8      <1> finished:
16     16      0000000D 5B        <1> sub     eax, ebx
17     17      0000000E C3        <1> pop     ebx
18     18      0000000F 00        <1> ret
19     19
20     20      00000010 00        <1> ;----- sprint -----
21     21      00000011 00        <1> ; Функция печати сообщения
22     22      00000012 00        <1> ; входные данные: mov eax,<message>
23     23      00000013 00        <1> sprint:
24     24      00000014 52        <1> push    edx
25     25      00000015 51        <1> push    ecx
26     26      00000016 53        <1> push    ebx
27     27      00000017 50        <1> push    eax
28     28      00000018 E8E8FFFF    <1> call    slen
29     29
30     30      00000019 89C2      <1> mov     edx, eax
31     31      0000001A 58        <1> pop     eax
32     32
33     33      0000001B 89C1      <1> mov     ecx, eax
34     34      0000001C B801000000 <1> mov     ebx, 1
35     35      0000001D B804000000 <1> mov     eax, 4
36     36      0000001E CD80      <1> int     80h
37     37
38     38      0000001F 5B        <1> pop     ebx
39     39      00000020 59        <1> pop     ecx
40     40      00000021 5A        <1> pop     edx
41     41      00000022 C3        <1> ret
42     42
43     43
44     44      00000023 00        <1> ;----- sprintf -----
45     45      00000024 00        <1> ; Функция печати сообщения с переводом строки
46     46      00000025 00        <1> ; входные данные: mov eax,<message>
47     47      00000026 00        <1> sprintf:
48     48      00000027 E8DDFFFF    <1> call    sprint
49     49
50     50      00000028 50        <1> push    eax
51     51      00000029 B80A000000 <1> mov     eax, 0Ah
52     52      0000002A 50        <1> push    eax
53     53      0000002B 89E0      <1> mov     eax, esp
54     54      0000002C E8CFFFFF    <1> call    sprintf
55     55      0000002D 58        <1> pop     eax
56     56      0000002E 58        <1> pop     eax
57     57      0000002F C3        <1> ret
58     58

```

Рис. 2.12: Файл листинга 2

### 2.3.1 Объяснение содержимого файлов листинга

1. На первой строчке находится подключение внешнего файла. Дальше в листинге содержится все содержимое внешнего файла.
2. Наша программа начинается со 192 строчки, там находится `_start`
3. Уа сьолчке 201 находится вызов функции `sread`, справа написан номер строки в файле, адрес в памяти и код команды

### 2.3.2 Создадим ошибку в программе

На 57 строчке уберем один регистр

```

56  fin:
57      mov eax,
58      call sprint : Вь

```

Рис. 2.13: Изменение файла lab8-2.asm

Теперь создадим листинг для этой программы.

```

eapleskacheva@localhost lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:57: error: invalid combination of opcode and operands
eapleskacheva@localhost lab08]$ 

```

Рис. 2.14: Создание листинга для измененной программы

При создании листинга выводится ошибка,но листинг создается

Посмотрим листинг

```

231  56      fin:
232  57      mov eax,
233  57      ***** error: invalid combination of opcode and operands
234  58      call sprint : Вь

```

Рис. 2.15: Просмотр листинга

После строчки, на которой мы сделали ошибку появляется сообщение об ошибке

## **3 Задание для самостоятельной работы**

### **3.1 Программа, находящая наименьшее**

Напишем программу, которая будет выводить наименьшее число из чисел 82 59 61 (потому что мой вариант 2)

```

1 %include "in_out.asm"
2
3 section .data
4 msg1 db "Введите a: ",0h
5 msg2 db "Введите b: ",0h
6 msg3 db "Введите c: ",0h
7 ans db "Наименьшее число: ",0h
8 num1 db '82'
9 num2 db '59'
10 num3 db '61'
11 section .bss
12 min resb 50
13 section .text
14
15 GLOBAL _start
16
17 _start:
18
19     mov eax, num1
20     call atoi
21     mov [min], eax        ; min = num1
22
23
24 .comparing2:
25     mov eax, num2
26     call atoi
27     cmp [min], eax        ; cmp num1, num2
28     jl .comparing3        ; if num1 < num2 jmp to .comparing
29     mov [min], eax        ; min = eax = num2
30
31
32
33 .comparing3:
34
35     mov eax, num3
36     call atoi
37     cmp [min], eax        ; cmp num2, num3
38     jl .final            ; if num2 < num3 jmp to .comparing
39     mov [min], eax        ; min = eax = num3
40
41
42 .final:
43     mov eax, ans
44     call sprintf
45
46     mov eax, [min]
47     call iprintLF
48
49
50 call quit                ;Выход
51

```

Рис. 3.1: Написанная программа по нахождению минимального

Скомпилируем и запустим ее

```

[eapleskacheva@localhost lab08]$ ./lab8-3
Наименьшее число:
59
[eapleskacheva@localhost lab08]$

```

Рис. 3.2: Запуск программы по вычислению минимума

Программа верно вывела наименьшее число 59

## 4 Программа, вычисляющая значение функции

Напишем программу которая будет вчислять функцию:

$a - 1$ , если  $x < a$   $x - 1$ , если  $x \geq a$



```

15
16 _start:
17
18
19     ;; PRINT msg1
20     push eax
21     mov eax, msg1
22     call sprintf
23     pop eax
24
25     mov ecx, x
26     mov edx, 200
27     call sread
28     mov eax, x
29     call atoi
30     mov [x], eax           ; max = num1
31
32 ;
33     ;; PRINT msg1
34     push eax
35     mov eax, msg2
36     call sprintf
37     pop eax; --- READ A ---
38
39     mov ecx, a
40     mov edx, 200
41     call sread
42     mov eax, a
43     call atoi
44     mov [a], eax           ; max = num1
45
46
47 .var1:
48
49     xor ecx, ecx
50     mov ecx, [x]
51     cmp ecx, [a]
52     jge .var2               ; if x >= a, jump to var2
53
54     push eax
55     mov eax, debug
56     call sprintf
57     pop eax
58
59     xor eax, eax
60     mov eax, [a]
61     dec eax                 ; eax = a - 1
62
63     jp .final
64
65 .var2:
66     xor eax, eax
67     mov eax, [x]
68     dec eax                 ; eax = x - 1
69
70 .final:
71
72     ;; PRINT msg1
73     push eax

```

Рис. 4.1: Написанная программа

Скомпилируем ее и запустим. Проверим на значениях (5;7) и (6;4)

```
[eapleskacheva@localhost lab08]$ nasm -f elf ./lab8-4.asm
ld -m elf_i386 -o ./lab8-4 ./lab8-4.o
./lab8-4
Введите x:
5
Введите a:
7
debug
Ответ:
6
[eapleskacheva@localhost lab08]$ ./lab8-4
Введите x:
6
Введите a:
4
Ответ:
5
[eapleskacheva@localhost lab08]$
```

Рис. 4.2: Проверка выполнения программы

Программа вычисляет функцию верно.

## 5 Выводы

Мы изучили условные переходы в языке Ассемблера NASM и научились писать программы с их использованием и ознакомились со структурой файлов листинга.