

Лабораторная работа №7

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

Приспешкин Андрей Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
	Список литературы	16

Список иллюстраций

3.1	Создание рабочего каталога и файла lab7-1.asm	7
3.2	Код из листинга 7.1	8
3.3	Создание исполняемого файла и результат его работы	8
3.4	Результат после внесения изменений в код	9
3.5	Код после внесения изменений	9
3.6	Проверка результата работы исполняемого файла	10
3.7	Создание файла lab7-2.asm	10
3.8	Код из листинга 7.3	10
3.9	Проверка работы программы	11
3.10	Создание файла листинга	11
3.11	Открытие файла листинга в текстовом редакторе	11
3.12	Вид файла листинга	12
3.13	Строка 45 после удаления операнда msg2	13
3.14	Ошибка при создании объектного файла	13

Список таблиц

1 Цель работы

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

2 Задание

1. Команды безусловного перехода
2. Команды условного перехода
3. Файлы листинга
4. Задания для самостоятельной работы

3 Выполнение лабораторной работы

Создадим новый каталог в котором будем делать лабораторную работу, в рабочем каталоге arch-рс, там-же создадим файл lab7-1.asm(Рис.1).

```
aaprispeshkin:[aaprispeshkin]:~$ mkdir ~/work/arch-pc/lab07  
aaprispeshkin:[aaprispeshkin]:~$ cd ~/work/arch-pc/lab07  
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 3.1: Создание рабочего каталога и файла lab7-1.asm

В файл lab7-1.asm вставим код из листинга 7.1(Рис.2).

```
lab07 : bash — Konsole
New Tab ▾ Split View ▾

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

Рис. 3.2: Код из листинга 7.1

Создадим исполняемый файл и проверим результат работы кода(Рис.3).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 3.3: Создание исполняемого файла и результат его работы

Изменим код в соответствии с листингом 7.2 и проверим результат(Рис.4).


```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.4: Результат после внесения изменений в код

Поменяем код так, чтобы порядок выводимых на экран сообщений был “3, 2, 1”(Рис.5).

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

Рис. 3.5: Код после внесения изменений

Создадим исполняемый файл и проверим результат(Рис.6).

```

aapripeshkin:[aapripeshkin]:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aapripeshkin:[aapripeshkin]:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aapripeshkin:[aapripeshkin]:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 3.6: Проверка результата работы исполняемого файла

Создадим файл lab7-2.asm(Рис.7).

```

aapripeshkin:[aapripeshkin]:~/work/arch-pc/lab07$ touch lab7-2.asm

```

Рис. 3.7: Создание файла lab7-2.asm

Вставим туда код в соответствии с листингом 7.3(Рис.8).

```

include "in_out.asm"
section .data
msg1 db "Введите B: ",0h
msg2 db "Наибольшее число: ",0h
A dd '20'
B dd '50'
section .bss
max resb 10
section .text
global _start
_start:
    ; ----- Вывод сообщения "Введите B: "
    mov eax,msg1
    call sprintf
    ; ----- Ввод 'B'
    mov ecx,B
    mov edx,10
    call sread
    ; ----- Преобразование 'B' из символа в число
    mov eax,B
    call atoi ; Выход 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
    call atoi ; Выход подпрограммы перевода символа в число
    mov [max],eax ; запись преобразованного числа в 'max'
    ; ----- Сравниваем 'max(A,C)' и 'B' (как число)
    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:

```

Рис. 3.8: Код из листинга 7.3

Проверим правильность работы программы с разными входными данными(Рис.9).

```

aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 52
Наибольшее число: 52
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 23
Наибольшее число: 50

```

Рис. 3.9: Проверка работы программы

Создадим файла листинга для файла lab7-2.asm(Рис.10).

```

aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm

```

Рис. 3.10: Создание файла листинга

Откроем его текстовым редактором neovim(Рис.11).

```

aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nvim lab7-2.lst

```

Рис. 3.11: Открытие файла листинга в текстовом редакторе

Заметим, что файлы листинга устроены таким образом, что сначала там пишется номер строки, а затем адреса в памяти(Рис.12).

```

185 10 section .text
186 11 global _start
187 12 _start:
188 13 ; ----- Вывод сообщения 'Введите B: '
189 14 000000E8 B8[00000000] mov eax,msg1
190 15 000000ED E81DFFFFFF call sprint
191 16 ; ----- Ввод 'B'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call sread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
198 23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[39000000] mov ecx,[A] ; 'ecx = A'
201 26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
204 29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
205 30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
206 31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
207 32 ; ----- Преобразование 'max(A,C)' из символа в число
208 33 check_B:
209 34 00000130 B8[00000000] mov eax,max
210 35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
211 36 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
212 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
213 38 0000013F 8B0D[00000000] mov ecx,[max]
214 39 00000145 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
215 40 0000014B 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
216 41 0000014D 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
217 42 00000153 890D[00000000] mov [max],ecx
218 43 ; ----- Вывод результата
219 44 fin:
220 45 00000159 B8[13000000] mov eax,msg2
221 46 0000015E E8ACFFFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
222 47 00000163 A1[00000000] mov eax,[max]
223 48 00000168 E819FFFFFF call iprintLF ; Вывод 'max(A,B,C)'
224 49 0000016D E869FFFFFF call quit ; Выход

```

Рис. 3.12: Вид файла листинга

Попробуем удалить из файла lab7-2.asm операнд из строки 45(Рис.13).

```

43 ; ----- вывод результата
44 fin:
45 mov eax,
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax, [max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход

```

Рис. 3.13: Строка 45 после удаления операнда msg2

Заметим, что в результате при попытке создать объектный файл мы получаем ошибку(Рис.14).

```

aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:45: error: invalid combination of opcode and operands

```

Рис. 3.14: Ошибка при создании объектного файла

Напишем код, который будет вычислять выражение $2a - x$ в случае если $x < a$ или выводить 8 в противном случае(Рис.15 и 16).

```

1 %include "in_out.asm"
2
3 section .data
4 msg1 db "Введите x: ", 0h
5 msg2 db "Введите a: ", 0h
6 msg3 db "Ответ: ", 0h
7
8 section .bss
9 X resb 10
10 A resb 10
11
12 section .text
13 global _start
14
15 _start:
16 mov eax, msg1
17 call sprint
18 mov ecx, X
19 mov edx, 10
20 call sread
21 mov eax, X
22 call stoi
23 mov [X], eax
24
25 mov eax, msg2
26 call sprint
27 mov ecx, A
28 mov edx, 10
29 call sread
30 mov eax, A
31 call stoi
32 mov [A], eax
33
34 cmp [X], eax
35 jl Lesser
36 mov eax, msg3
37 call sprint
38 mov eax, 8
39 call iprintLF
40 call quit
41
42 Lesser:
43 mov eax, [A]
44 mov ebx, 2

```

```

22  call atoi
23  mov [X], eax
24
25  mov eax, msg2
26  call sprint
27  mov ecx, A
28  mov edx, 10
29  call sread
30  mov eax, A
31  call atoi
32  mov [A], eax
33
34  cmp [X], eax
35  jl Lesser
36  mov eax, msg3
37  call sprint
38  mov eax, 8
39  call iprintLF
40  call quit
41
42  Lesser:
43  mov eax, [A]
44  mov ebx, 2
45  mul ebx
46  mov ebx, [X]
47  sub eax, ebx
48  mov ecx, eax
49  mov eax, msg3
50  call sprint
51  mov eax, ecx
52  call iprintLF
53  call quit
54

```

Создадим исполняемый файл и проверим результат его работы при $x(1,2)$ и $a(2,1)$ (Рис.17)

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Ввведита a: 2
Ответ: 3
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Ввведита a: 1
Ответ: 8
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$
```

Выводы

Я изучил команды безусловного и условного переходов и приобрёл навыки написания программ с их использованием.

Список литературы

Лабораторная работа №7