

Лабораторная работа №7

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

Славинский Владислав Вадимович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение задания для самостоятельной работы	14
4	Вывод	19

Список иллюстраций

2.1	Создание lab7-1	6
2.2	Код программы	7
2.3	Запуск	7
2.4	Замена	8
2.5	Запуск	8
2.6	Создание файла lab7-2.asm	9
2.7	Программа	10
2.8	Запуск	11
2.9	Получение файла листинга	11
2.10	Содержимое файла	11
2.11	Вычисление выражения	12
2.12	Запуск вычисления выражения	13
2.13	Запуск вычисления выражения	13
3.1	Создание laba7.asm	14
3.2	Код программы для нахождения наименьшего	15
3.3	Запуск программы	16
3.4	Создание laba72.asm	16
3.5	Код программы	17
3.6	Запуск программы	18

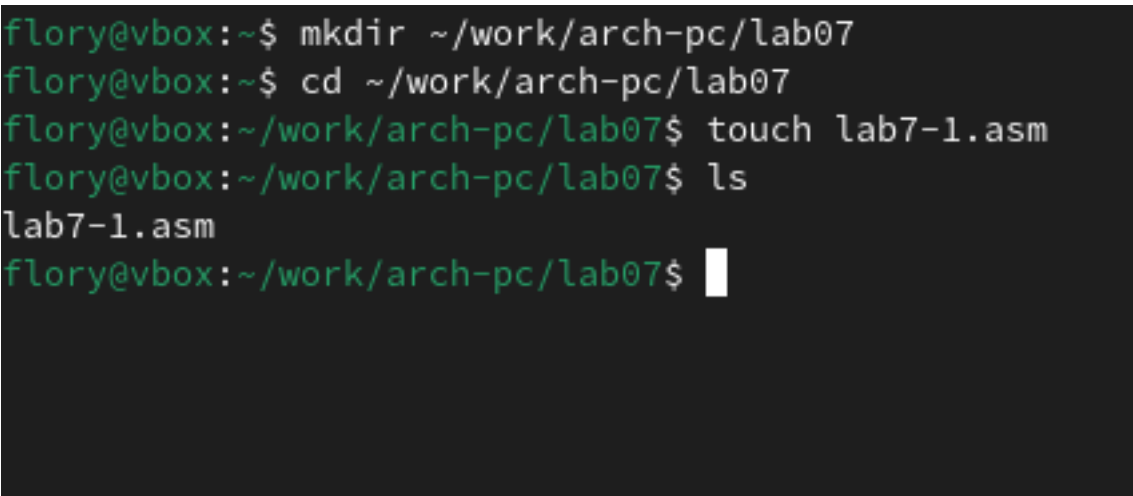
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1) Создал каталог для лабораторной работы №7 и создал файл lab7-1.asm: (Рис. 2.1)



```
flory@vbox:~$ mkdir ~/work/arch-pc/lab07
flory@vbox:~$ cd ~/work/arch-pc/lab07
flory@vbox:~/work/arch-pc/lab07$ touch lab7-1.asm
flory@vbox:~/work/arch-pc/lab07$ ls
lab7-1.asm
flory@vbox:~/work/arch-pc/lab07$
```

Рис. 2.1: Создание lab7-1

2) Ввел код программы с использованием конструкции jmp: (Рис. 2.2)

```

lab7-1.asm      [----] 41 L:[ 6+21 27/ 27] *(656 / 656b) <EOF>
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения

```

Рис. 2.2: Код программы

3) Создал исполняемый файл и запустил его: (Рис. 2.3)

```

flory@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
flory@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
flory@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
flory@vbox:~/work/arch-pc/lab07$ █

```

Рис. 2.3: Запуск

4) Теперь изменим программу, чтобы она выводил сначала сообщение № 2, а потом сообщение № 1: (Рис. 2.4)

```

lab7-1.asm      [----]  4 L:[  8+ 9 17/ 29] *(367 / 677b) 0095 0x05F
SECTION .text
GLOBAL _start
_start:

jmp _label2

_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'

_end:
call quit ; вызов подпрограммы завершения

```

Рис. 2.4: Замена

5) Запустил измененный файл: (Рис. 2.5)

```

flory@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
flory@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
flory@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
flory@vbox:~/work/arch-pc/lab07$ █

```

Рис. 2.5: Запуск

6) Создал файл lab7-2.asm: (Рис. 2.6)


```
flory@vbox:~/work/arch-pc/lab07$ touch lab7-2.asm
flory@vbox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm
flory@vbox:~/work/arch-pc/lab07$
```

Рис. 2.6: Создание файла lab7-2.asm

7) Ввел код программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C: (Рис. 2.7)

```

lab7-2.asm      [-M--] 29 L:[ 1+30 31/ 50] *(961 /1744b) 0010 0x00A
#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
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. 2.7: Программа

8)Запустил его и убедился в работоспособности:(Рис. 2.8)

```
flory@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 32
Наибольшее число: 50
flory@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 51
Наибольшее число: 51
flory@vbox:~/work/arch-pc/lab07$
```

Рис. 2.8: Запуск

9)Получаю файл листинга, указав ключ -l и задав имя файла листинга в командной строке, и открыл файл с помощью редактора:(Рис. 2.9)

```
flory@vbox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
flory@vbox:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 2.9: Получение файла листинга

10)Внимательно ознакомился с его форматом и содержимым:(Рис. 2.10)

```
1 1: function: main
2 1:
3 1:
4 00000000 01: silen; "silence"
5 00000001 02: mov     eax, 0
6 1:
7 1:
8 00000003 03: mov     ecx, 0
9 00000004 04: mov     ecx, 0
10 00000005 05: mov     ecx, 0
11 00000006 06: mov     ecx, 0
12 1:
13 1:
14 00000008 07: sub     eax, 0
15 00000009 08: mov     eax, 0
16 0000000A 09: ret
17 1:
18 1:
19 1:
20 1: function: print
21 1: function: print
22 1: function: print
23 0000000F 10: push    ecx
24 00000010 11: push    ecx
25 00000011 12: push    ecx
26 00000012 13: push    ecx
27 00000013 14: call    silen
28 1:
29 00000014 15: mov     ecx, 0
30 00000015 16: mov     ecx, 0
31 1:
32 00000016 17: mov     ecx, 0
33 00000017 18: mov     ecx, 0
34 00000018 19: mov     ecx, 0
35 00000019 20: mov     ecx, 0
36 00000020 21: mov     ecx, 0
37 00000021 22: mov     ecx, 0
38 00000022 23: mov     ecx, 0
39 00000023 24: mov     ecx, 0
40 00000024 25: mov     ecx, 0
41 1:
42 1:
43 1:
44 1: function: print
45 1: function: print
46 1: function: print
47 0000002D 26: call    print
48 1:
49 1:
50 0000002F 27: push    ecx
51 00000030 28: mov     ecx, 0
52 00000031 29: mov     ecx, 0
53 00000032 30: mov     ecx, 0
54 00000033 31: mov     ecx, 0
55 00000034 32: mov     ecx, 0
56 00000035 33: mov     ecx, 0
57 1:
58 1:
59 1:
60 1: function: print
61 1: function: print
62 00000037 34: push    ecx
63 00000038 35: push    ecx
64 1:
65 00000039 36: mov     ecx, 0
66 00000040 37: mov     ecx, 0
```

Рис. 2.10: Содержимое файла

Объясню что делают 8, 9 и 10 строки. Строка 8 `cmp byte [eax], 0`. Эта инструкция сравнивает байт, находящийся по адресу, на который указывает регистр `eax`, с нулем. `cmp` - команда, которая выполняет вычитание, но не сохраняет результат. `[eax]` указывает на значение по адресу, хранящемуся в `eax`. Если значение равно нулю, то флаг нуля будет установлен. Строка 9 `jz finished`. Эта инструкция выполняет переход к метке `finished`, если установлен флаг нуля (если результат предыдущего сравнения был бы равен нулю). `jz` - условный переход, который срабатывает, если результат предыдущей операции (в данном случае `cmp`) указывает на то, что сравниваемые значения равны (т.е. `byte [eax]` равно 0). Если байт по адресу `eax` равен 0, выполнение кода переходит к метке `finished`, что означает конец строки или конец строки в строке. Строка 10: `inc eax`. Инструкция увеличивает значение в регистре `eax` на 1. `inc` увеличивает значение в указанном регистре на 1, `eax` указывает на текущий символ в строке, и эта команда перемещает указатель к следующему символу. Если текущий символ не равен 0, то `eax` будет увеличен, чтобы указать на следующий символ в строке.

11) Удалю одну операнду `ecx` в строке `mov [max], ecx`: (Рис. 2.11)

```
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:
```

Рис. 2.11: Вычисление выражения

12) Создал исполняемый файл и запустил его. Вывод: ошибка: (Рис. 2.12)

```
flory@vbox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:27: error: invalid combination of opcode and operands
flory@vbox:~/work/arch-pc/lab07$
```

Рис. 2.12: Запуск вычисления выражения

13)Открою теперь файл листинга и проверим его на ошибку:(Рис. 2.13)

```
mov [B],ecx ; Записываем преобразованное значение в B
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max] ; 'max = A'
error: invalid combination of opcode and operands
; ----- 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',
```

Рис. 2.13: Запуск вычисления выражения

В листинге отображается ошибка

3 Выполнение задания для самостоятельной работы

1)Создам первый файл laba7.asm для самостоятельной работы. Вариант 10.
:(Рис. 3.1)

```
flory@vbox:~/work/arch-pc/lab07$ touch laba7.asm
flory@vbox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst  laba7.asm
flory@vbox:~/work/arch-pc/lab07$
```

Рис. 3.1: Создание laba7.asm

2)Написал код программы для нахождения наименьшей из 3 целочисленных переменных.(Рис. 3.2)

```

laba7.asm      [----]  4 L:[  1+35  36/ 49] *(372 / 511b) 0107 0x06B
%include 'in_out.asm'

section .data
a dd '41'
b dd '62'
c dd '35'
msg2 db "Минимальное число: ", 0h

section .bss
min resb 10

section .text
global _start
_start:

mov eax, b
call atoi
mov [b], eax

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

mov eax, c
call atoi
mov [c], eax

mov ecx, [a]
mov [min], ecx

cmp ecx, [c]
jl check_b
mov ecx, [c]
mov [min], ecx

check_b:
mov ecx, [min]
cmp ecx, [b]
jl fin
mov ecx, [b]
mov [min], ecx

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

```

Рис. 3.2: Код программы для нахождения наименьшего

3)Создал исполняемый файл и запустил его.(Рис. 3.3)

```
flory@vbox:~/work/arch-pc/lab07$ nasm -f elf laba7.asm
flory@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o laba7 laba7.o
flory@vbox:~/work/arch-pc/lab07$ ./laba7
Минимальное число: 35
```

Рис. 3.3: Запуск программы

- 4) Создам второй файл laba72.asm для второго задания. Рис. 3.4)

```
flory@vbox:~/work/arch-pc/lab07$ touch laba72.asm
flory@vbox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm  laba7-2      lab7-2.lst  laba72.asm  laba7.o
laba7-1     lab7-1.o    lab7-2.asm  laba7       laba7.asm
```

Рис. 3.4: Создание laba72.asm

- 5) Написал код программы для вычисления значения заданной функции $f(x)$.
(Рис. 3.5)


```

laba72.asm      [-M--]  0 L:[ 1+45  46/ 53] *(657 / 775b) 0010 0x00A
#include 'in_out.asm'

section .data
    msg1 DB "Введите X: ", 0h
    msg2 DB "Введите A: ", 0h
    msg3 DB "Ответ = ", 0h

section .bss
    x: RESB 80
    a: RESB 80
    ans: RESB 80

section .text
    global _start

_start:
    mov eax, msg1
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov [x], eax

    mov eax, msg2
    call sprint
    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atoi
    mov [a], eax

    mov eax, [x]
    cmp eax, 2
    jg greater_than_2

    mov eax, [a]
    shl eax, 1
    add eax, [a]
    jmp store_result

greater_than_2:
    sub eax, 2

store_result:
    mov [ans], eax
    mov eax, msg3
    call sprint
    mov eax, [ans]
    call iprintLF
    call quit

```

Рис. 3.5: Код программы

6) Создал исполняемый файл и запустил его. (Рис. 3.6)

```
flory@vbox:~/work/arch-pc/lab07$ nasm -f elf laba72.asm
flory@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o laba72 laba72.o
flory@vbox:~/work/arch-pc/lab07$ ./laba72
Введите X: 3
Введите A: 0
Ответ = 1
flory@vbox:~/work/arch-pc/lab07$ ./laba72
Введите X: 1
Введите A: 2
Ответ = 6
flory@vbox:~/work/arch-pc/lab07$
```

Рис. 3.6: Запуск программы

4 Вывод

В ходе выполнения лабораторной работы я изучил команды условного и безусловного переходов, также приобрел навыки написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.