

# **Лабораторная работа №8**

Руслан Шухратович Исмаилов

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>4</b>	<b>Задание для самостоятельной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>17</b>

## Список иллюстраций

4.1	работа lab8-1.asm . . . . .	9
4.2	работа lab8-1.asm (2) . . . . .	9
4.3	код lab8-2 . . . . .	9
4.4	проверка lab8-2 . . . . .	10
4.5	разные В . . . . .	10
4.6	Файл листинга . . . . .	11
4.7	Ошибка в файле листинга . . . . .	12
4.8	код min.asm . . . . .	13
4.9	код min.asm . . . . .	14
4.10	проверка min.asm . . . . .	15
4.11	код var14.asm . . . . .	15
4.12	код var14.asm . . . . .	16
4.13	проверка var14.asm . . . . .	16

# 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

Написать программу нахождения наименьшей из 3-х переменных и вычисления уравнения в зависимости от размера введенных переменных (по вариантам)

## 3 Выполнение лабораторной работы

### Шаг 1

Создадим файл lab8-1.asm в каталоге для лабораторной работы 8:

введём в него код из Листинга 8.1

запустим lab8-1:

(рис. 4.1)

Сообщение 1 отсутствует, хотя оно и есть в тексте файла. Использование инструкции `jmp` меняет порядок исполнения инструкций, позволяет исполнить `_label2`, потом `_label3` пропустив инструкцию `_label1`.

введём код из Листинга 8.2, для того чтобы программа выводила ‘Сообщение No 2’, потом ‘Сообщение No 1’, проверим работу файла:

(рис. 4.2)

### Шаг 2

Изменим код, чтобы сообщения выводились в порядке 3-2-1; добавим `jmp _label3` в начале, после вывода 3 сообщения переходим в `_label2`, аналогично переходим в `_label1`, и оттуда переходим в подпрограмму завершения

(рис. 4.3)

Проверим

(рис. 4.4)

### Шаг 3

Создадим lab8-2.asm, и введем в него код из Листинга 8.3, для нахождения меньшего из 3 чисел с помощью инструкции `cmp`(сравнение) и `jg`(переход если больше),  $A = 20$ ,  $C = 50$ , и переменной  $B$

Проверим его работу для различных В:

(рис. 4.5)

#### **Шаг 4**

Создадим файл листинга для lab8-2.asm, используя ключ -l в команде nasm, и откроем его, чтобы ознакомиться с содержимым. Рассмотрим 3 строчки для примера структуры листинга:

(рис. 4.6)

```
45 00000153 B8[13000000] mov eax, msg2
```

45 (номер строки) 00000153 (адрес, для того чтобы инструкции по порядку выполнялись) B8 (инструкция на машинном языке) [13000000] (переменная) mov eax, msg2 (сам текст файла) 46 00000158 E8B2FEFFFF call sprint ; Вывод сообщения 'Наибольшее число:' (комментарий) 47 0000015D A1[00000000] mov eax,[max] 46,47 - номер строки;

00000158, 0000015D - адрес

E8B2FEFFFF, A1 - машинный код, инструкция на машинном языке, отвечающая за исполнение команды sprint (вывод сообщения на экран) и перемещение переменной max в eax

[00000000] - переменная

Справа находится исходный код нашей программы и комментарии

Удалим один операнд в операции, требующей два и создадим файл листинга, lab8-2.lst и посмотрим как он изменился:

(рис. 4.7)

Как мы видим, в файле листинга около строки кода находится предупреждение об ошибке.

## 4 Задание для самостоятельной работы

### Шаг 1

Создадим файл `min.asm` для создания программы для нахождения наименьшей из 3 переменных за основу взяв код `lab8-2.asm`

(рис. 4.8)

(рис. 4.9)

Принцип работы:

Мы записываем с клавиатуры 3 переменные, переводим их в числа

Далее как в коде Листинга 8.3, заменим `jl` (переход если больше) на `jl` (переход если меньше) (Также См комментарии в коде)

Проверяем:

(рис. 4.10)

Создадим файл `var14.asm` для создания программы вычисления ответа на систему уравнений из двух уравнений с использованием 2 переменных `a, x` за основу взяв код `lab8-2.asm`

Как и указано в комментариях, мы записываем введенные значения `a` и `x` в переменные `A` и `X`, преобразуем их в числа для работы с операциями сложения и умножения, и сравниваем `X` и `A`.

Если  $a < x$  то выполняется программа вычисления  $3a+1$ , выводим результат, в противном случае ищем  $3x+1$ , с помощью `jmp` переходим в конец, где мы выводим сообщение 'Ответ:' на экран

Код (рис. 4.11)

(рис. 4.12)



Проверим

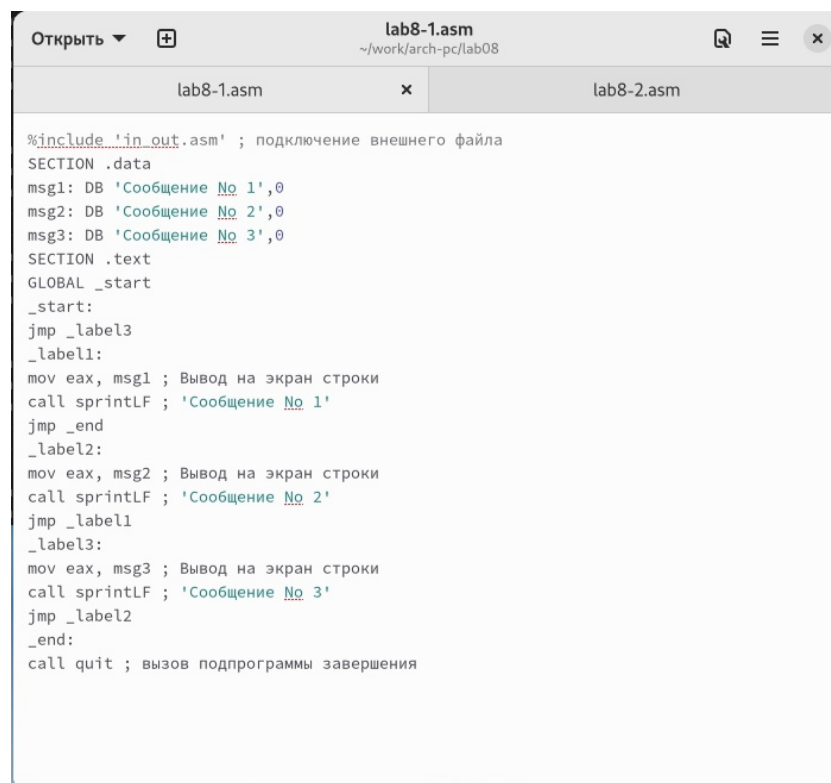
(рис. 4.13)

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

Рис. 4.1: работа lab8-1.asm

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

Рис. 4.2: работа lab8-1.asm (2)



```
lab8-1.asm
~/work/arch-pc/lab08

lab8-1.asm x lab8-2.asm

%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 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'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.3: код lab8-2

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

Рис. 4.4: проверка lab8-2

```
[rsismailov@fedora lab08]$ nasm -f elf lab8-2.asm
[rsismailov@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[rsismailov@fedora lab08]$ ./lab8-2
Введите B: 32
Наибольшее число: 50
[rsismailov@fedora lab08]$ ./lab8-2
Введите B: 51
Наибольшее число: 51
[rsismailov@fedora lab08]$ ./lab8-2
Введите B: -55
Наибольшее число: 50
[rsismailov@fedora lab08]$
```

Рис. 4.5: разные B

```

5 00000035 32300000      A dd '20'
6 00000039 35300000      C dd '50'
7                               section .bss
8 00000000 <res Ah>      max resb 10
9 0000000A <res Ah>      B resb 10
10                               section .text
11                               global start
12                               _start:
13                               ; ----- Вывод сообщения 'Введите B: '
14 000000E8 B8[00000000]      mov eax,msg1
15 000000ED E81DFFFFFF      call sprint
16                               ; ----- Ввод 'B'
17 000000F2 B9[0A000000]      mov ecx,B
18 000000F7 BA0A000000      mov edx,10
19 000000FC F842FFFFFF      call read
20                               ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]      mov eax,B
22 00000106 E891FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
23 0000010B A3[0A000000]      mov [B],eax ; запись преобразованного числа в 'B'
24                               ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]      mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000]      mov [max],ecx ; 'max = A'
27                               ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]      cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C              ig check_B ; если 'A>C', то переход на метку 'check_B',
30 00000124 8B0D[39000000]      mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000]      mov [max],ecx ; 'max = C'
32                               ; ----- Преобразование 'max(A,C)' из символа в число
33                               check_B:
34 00000130 B8[00000000]      mov eax,max
35 00000135 E862FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
36 0000013A A3[00000000]      mov [max],eax ; запись преобразованного числа в 'max'
37                               ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000]      mov ecx,[max]
39 00000145 3B0D[0A000000]      cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 0000014B 7F0C              ig fin ; если 'max(A,C)>B', то переход на 'fin',
41 0000014D 8B0D[0A000000]      mov ecx,[B] ; иначе 'ecx = B'
42 00000153 890D[00000000]      mov [max],ecx
43                               ; ----- Вывод результата
44                               fin:
45 00000159 B8[13000000]      mov eax, msg2
46 0000015E E8ACFFFFFF      call sprint ; Вывод сообщения 'Наибольшее число: '
47 00000163 A1[00000000]      mov eax,[max]
48 00000168 E819FFFFFF      call iprintf ; Вывод 'max(A,B,C)'
49 0000016D E869FFFFFF      call quit ; Выход

```

Рис. 4.6: Файл листинга

20		; ----- Преобразование 'B' из символа в число
21	00000101 B8[0A000000]	mov eax,B
22	00000106 E891FFFFFF	call atoi ; Вызов подпрограммы перевода символа в число
23	0000010B A3[0A000000]	mov [B],eax ; запись преобразованного числа в 'B'
24		; ----- Записываем 'A' в переменную 'max'
25	00000110 8B0D[35000000]	mov ecx,[A]; 'ecx = A'
26	00000116 890D[00000000]	mov [max],ecx ; 'max = A'
27		; ----- Сравниваем 'A' и 'C' (как символы)
28		cmp ecx, ; Сравниваем 'A' и 'C'
28	*****	error: invalid combination of opcode and operands
29	0000011C 7F0C	jg check_B ; если 'A>C', то переход на метку 'check_B',
30	0000011E 8B0D[39000000]	mov ecx,[C] ; иначе 'ecx = C'
31	00000124 890D[00000000]	mov [max],ecx ; 'max = C'
32		; ----- Преобразование 'max(A,C)' из символа в число
33		check_B:
34	0000012A B8[00000000]	mov eax,max
35	0000012F E868FFFFFF	call atoi ; Вызов подпрограммы перевода символа в число
36	00000134 A3[00000000]	mov [max],eax ; запись преобразованного числа в 'max'

Рис. 4.7: Ошибка в файле листинга

```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "меньшее число: ",0h
msg3 db 'Введите A: ',0h
msg4 db 'Введите C: ',0h
section .bss
max resb 10
A resb 10
B resb 10
C resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите A: '
mov eax,msg3
call sprint
; ----- Ввод 'A'
mov ecx,A
mov edx,10
call sread
; ----- Преобразование 'A' из символа в число
mov eax,A
call atoi ; Вызов подпрограммы перевода символа в число
mov [A],eax ; запись преобразованного числа в 'B'
; ----- 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'
; ----- Вывод сообщения 'Введите C: '
mov eax,msg4
call sprint

```

Рис. 4.8: код min.asm

```

; ----- Ввод 'C'
mov ecx,C
mov edx,10
call sread
; ----- Преобразование 'C' из символа в число
mov eax,C
call atoi ; Вызов подпрограммы перевода символа в число
mov [C],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'
cmp ecx,[C] ; Сравниваем 'A' и 'C'
j< check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Сравниваем 'max(A,C)' и 'B'
check_B:
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
j< 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
mov eax,[max]
call iprintLF ; Вывод
call quit ; Выход

```

Рис. 4.9: код min.asm

```

[rsismaiлов@fedora lab08]$ ./min
Введите A: 81
Введите B: 22
Введите C: 72
меньшее число: 22
[rsismaiлов@fedora lab08]$

```

Рис. 4.10: проверка min.asm

```

#include 'in_out.asm'
section .data
msg1 db 'Введите x: ',0h
msg2 db "Ответ: ",0h
msg3 db 'Введите a: ',0h

section .bss
max resb 10
x resb 10
a resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите x: '
mov eax,msg1
call sprint
; ----- Ввод 'x'
mov ecx,x
mov edx,10
call sread
; ----- Преобразование 'a' из символа в число
mov eax,x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x],eax ; запись преобразованного числа в 'a'

; ----- Вывод сообщения 'Введите a: '
mov eax,msg3
call sprint
; ----- Ввод 'a'
mov ecx,a
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,a
call atoi ; Вызов подпрограммы перевода символа в число
mov [a],eax ; запись преобразованного числа в 'B'

; ----- Записываем 'A' в переменную 'max'
mov ecx,[a]; 'ecx' = A
mov [max],ecx; 'max' = A

```

Рис. 4.11: код var14.asm

```

; ----- Записываем 'A' в переменную 'max'
mov ecx,[a]; 'ecx = A'
mov [max],ecx; 'max = A'

; ----- Сравниваем 'a' и 'x'
cmp ecx,[x]; Сравниваем 'a' и 'x'
jg check_B; если 'x<a', то переход на метку 'check_B',
;3x+1
mov eax,[x]
mov ecx,3
mul ecx
add eax,1
mov [max],eax
jmp fin
; ----- ;3a+1 x<a
check_B:
mov eax,[a]
mov ecx,3
mul ecx
add eax,1
mov [max],eax
; ----- 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. 4.12: код var14.asm

```

[rsismailov@fedora lab08]$ nasm -f elf var14.asm
[rsismailov@fedora lab08]$ ld -m elf_i386 -o var14 var14.o
[rsismailov@fedora lab08]$ ./var14
Введите x: 2
Введите a: 3
Ответ: 10
[rsismailov@fedora lab08]$ ./var14
Введите x: 4
Введите a: 2
Ответ: 13
[rsismailov@fedora lab08]$

```

Рис. 4.13: проверка var14.asm



## 5 Выводы

Я Изучил команды условного и безусловного переходов, приобрёл навыки написания программ с использованием переходов, познакомился с назначением и структурой файла листинга