

Отчёта по лабораторной работе 8

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

Ела Абого Мигель Анхель Ндонг

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	21

Список иллюстраций

2.1	Файл lab8-1.asm:	7
2.2	Программа lab8-1.asm:	7
2.3	Файл lab8-1.asm:	8
2.4	Программа lab8-1.asm:	9
2.5	Файл lab8-1.asm	10
2.6	Программа lab8-1.asm	11
2.7	Файл lab8-2.asm	12
2.8	Программа lab8-2.asm	13
2.9	Файл листинга lab8-2	14
2.10	ошибка трансляции lab8-2	15
2.11	файл листинга с ошибкой lab8-2	16
2.12	Файл lab8-3.asm	17
2.13	Программа lab8-3.asm	18
2.14	Файл lab8-4.asm	19
2.15	Программа lab8-4.asm	20

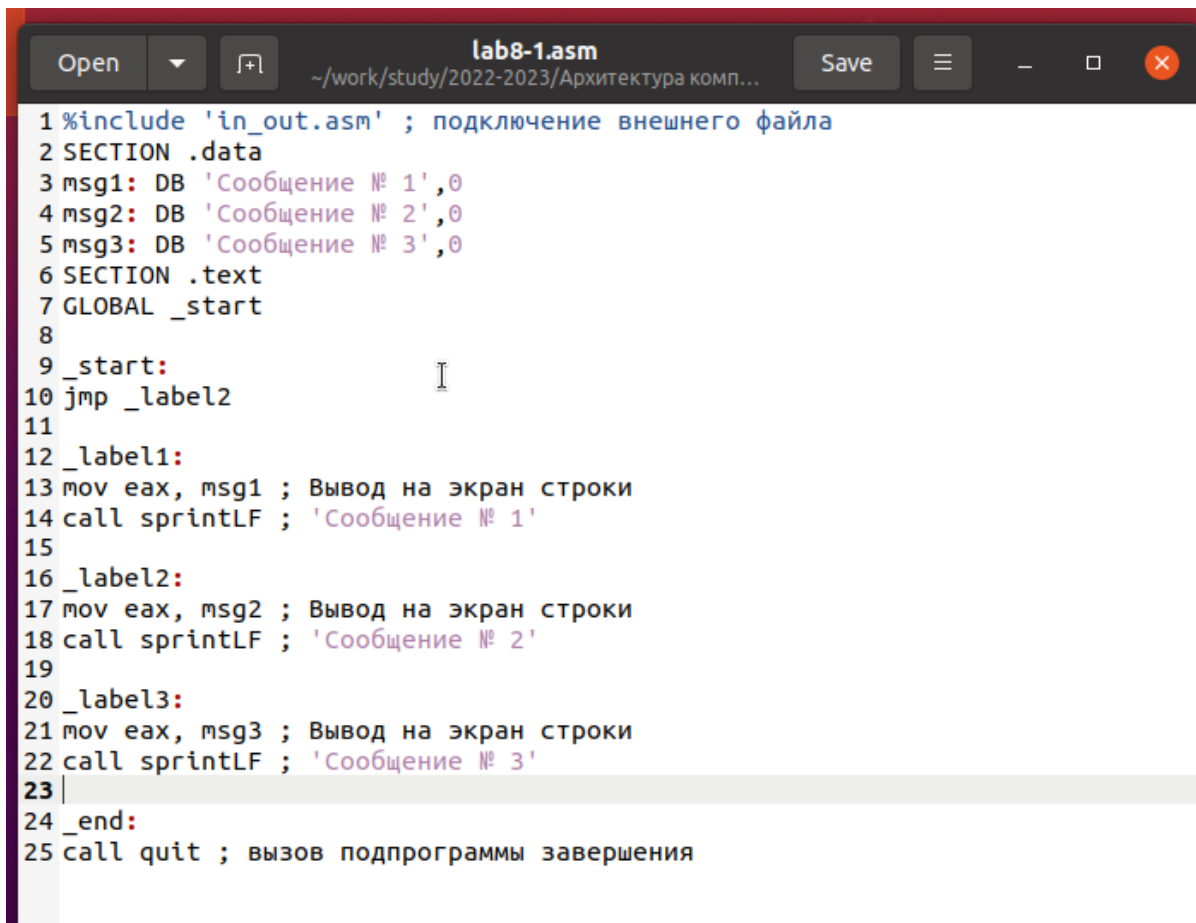
Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

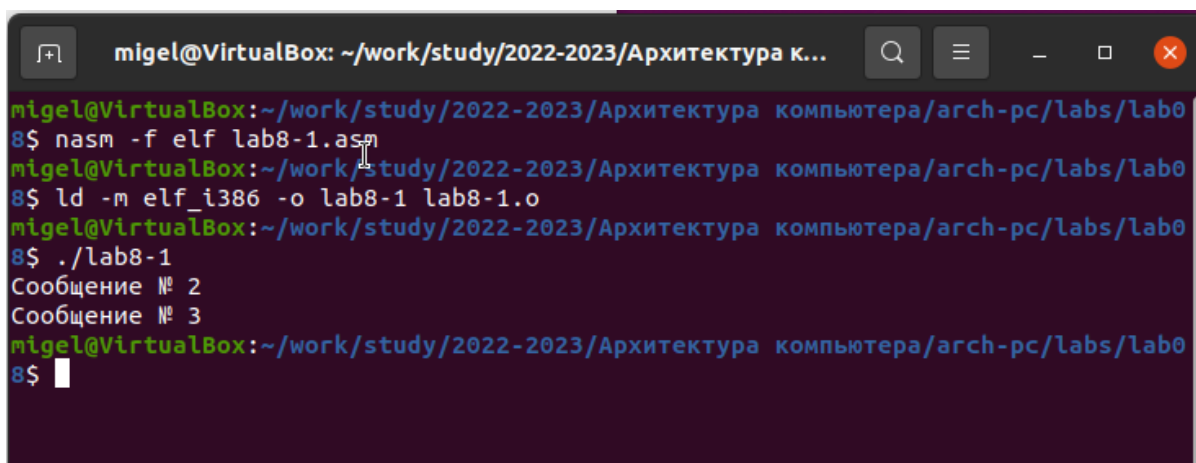
1. Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введите в файл lab8-1.asm текст программы из листинга 8.1. (рис. 2.1)



```
lab8-1.asm
~/work/study/2022-2023/Архитектура комп...
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
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1 ; Вывод на экран строки
14 call sprintf ; 'Сообщение № 1'
15
16 _label2:
17 mov eax, msg2 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 2'
19
20 _label3:
21 mov eax, msg3 ; Вывод на экран строки
22 call sprintf ; 'Сообщение № 3'
23
24 _end:
25 call quit ; вызов подпрограммы завершения
```

Рис. 2.1: Файл lab8-1.asm:

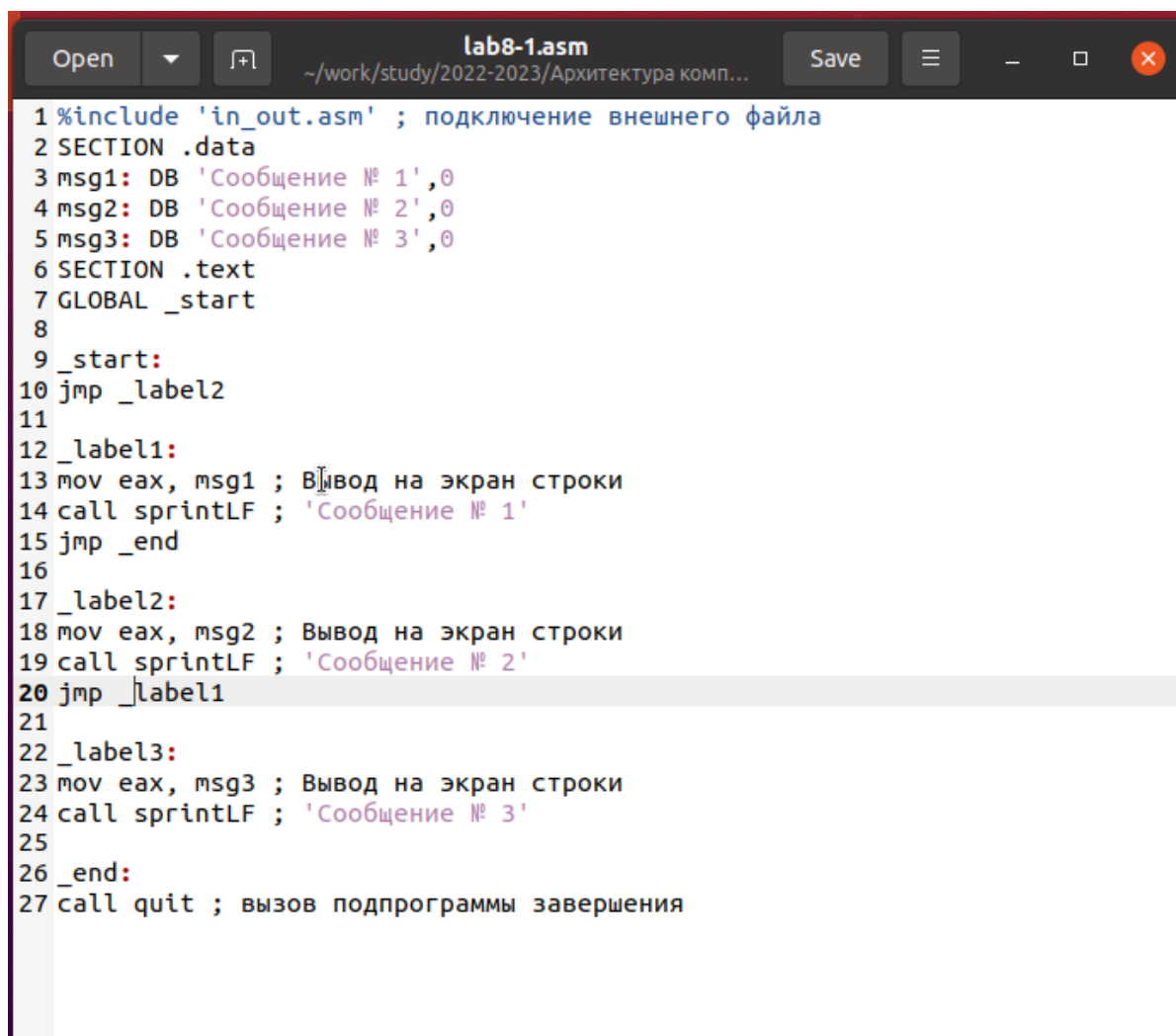
Создайте исполняемый файл и запустите его. (рис. 2.2)



```
migel@VirtualBox: ~/work/study/2022-2023/Архитектура к...
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-1.asm
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ld -m elf_i386 -o lab8-1 lab8-1.o
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-1
Сообщение № 2
Сообщение № 3
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$
```

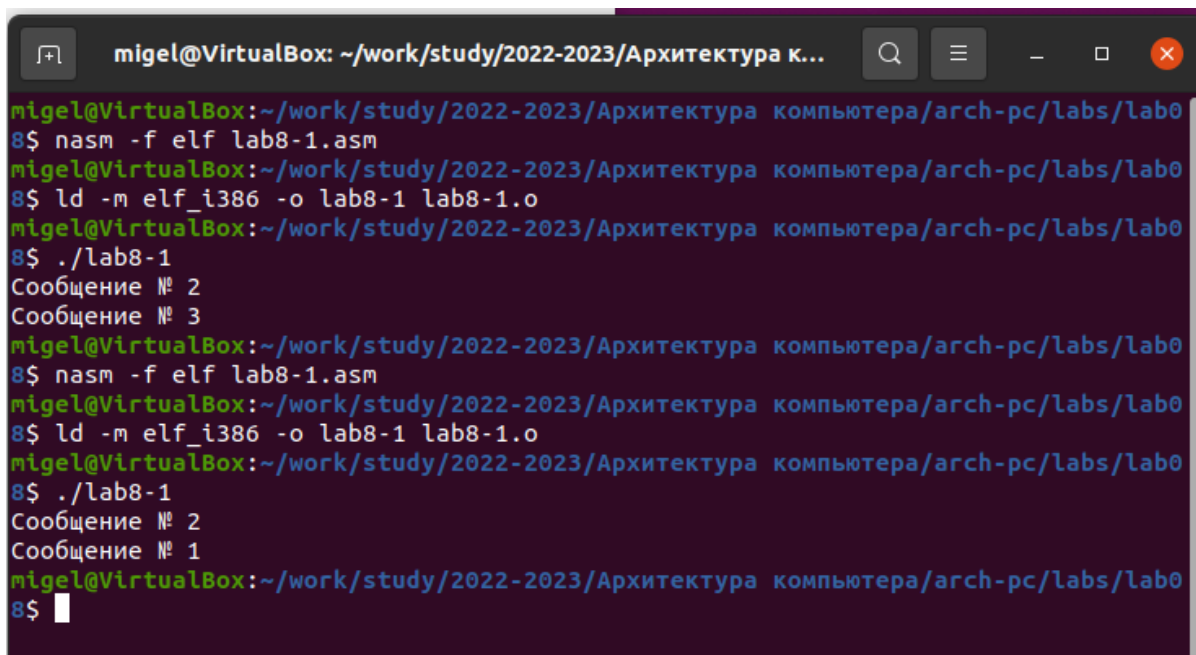
Рис. 2.2: Программа lab8-1.asm:

Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом 8.2. (рис. 2.3, 2.4)



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

Рис. 2.3: Файл lab8-1.asm:

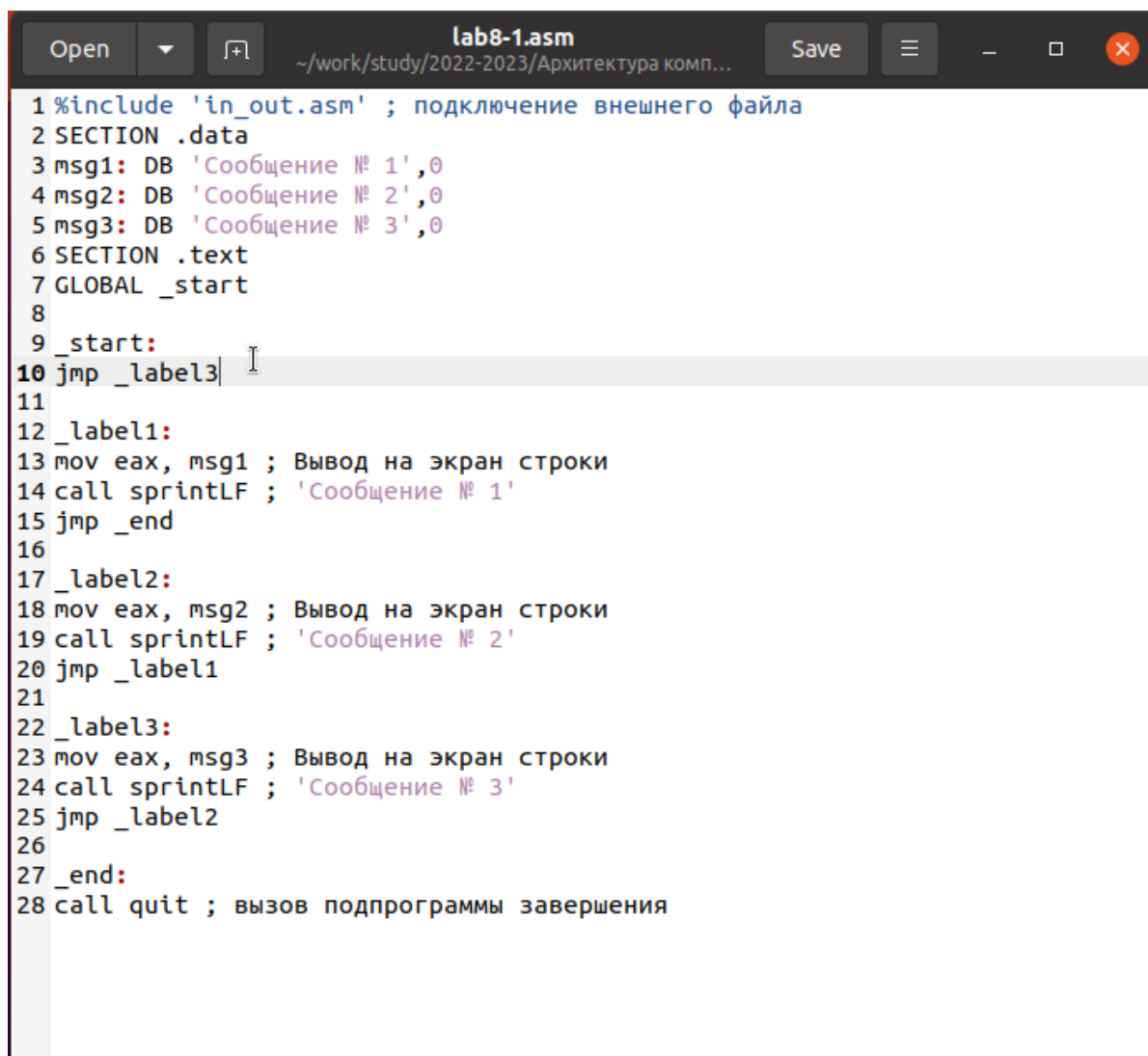


```
miguel@VirtualBox: ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-1.asm
miguel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ld -m elf_i386 -o lab8-1 lab8-1.o
miguel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-1
Сообщение № 2
Сообщение № 3
miguel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-1.asm
miguel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ld -m elf_i386 -o lab8-1 lab8-1.o
miguel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-1
Сообщение № 2
Сообщение № 1
miguel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$
```

Рис. 2.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим (рис. 2.5, 2.6):

Сообщение № 3
Сообщение № 2
Сообщение № 1



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

Рис. 2.5: Файл lab8-1.asm

```

migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-1
Сообщение № 2
Сообщение № 1
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-1.asm
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ld -m elf_i386 -o lab8-1 lab8-1.o
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$

```

Рис. 2.6: Программа lab8-1.asm

3. Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создайте исполняемый файл и проверьте его работу для разных значений В. (рис. 2.7, 2.8)

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

Рис. 2.7: Файл lab8-2.asm

```
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-2.asm
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ld -m elf_i386 -o lab8-2 lab8-2.o
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-2
Введите B: 80
Наибольшее число: 80
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-2
Введите B: 60
Наибольшее число: 60
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-2
Введите B: 40
Наибольшее число: 50
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$
```

Рис. 2.8: Программа lab8-2.asm

4. Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создайте файл листинга для программы из файла lab8-2.asm (рис. 2.9)

```
1 1 %include 'in_out.asm'
2 1 <1> ;----- slen -----
3 2 <1> ; Функция вычисления длины сообщения
4 3 <1> slen:
5 4 00000000 53 <1> push ebx
6 5 00000001 89C3 <1> mov ebx, eax
7 6 <1>
8 7 <1> nextchar:
9 8 00000003 803800 <1> cmp byte [eax], 0
10 9 00000006 7403 <1> jz finished
11 10 00000008 40 <1> inc eax
12 11 00000009 EBF8 <1> jmp nextchar
13 12 <1>
14 13 <1> finished:
15 14 0000000B 29D8 <1> sub eax, ebx
16 15 0000000D 5B <1> pop ebx
17 16 0000000E C3 <1> ret
18 17 <1>
19 18 <1>
20 19 <1> ;----- sprint -----
21 20 <1> ; Функция печати сообщения
22 21 <1> ; входные данные: mov eax,<message>
23 22 <1> sprint:
24 23 0000000F 52 <1> push edx
25 24 00000010 51 <1> push ecx
26 25 00000011 53 <1> push ebx
27 26 00000012 50 <1> push eax
28 27 00000013 E8E8FFFFFF <1> call slen
29 28 <1>
30 29 00000018 89C2 <1> mov edx, eax
31 30 0000001A 58 <1> pop eax
32 31 <1>
33 32 0000001B 89C1 <1> mov ecx, eax
34 33 0000001D BB01000000 <1> mov ebx, 1
35 34 00000022 B804000000 <1> mov eax, 4
36 35 00000027 CD80 <1> int 80h
37 36 <1>
```

Рис. 2.9: Файл листинга lab8-2

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

строка 10

- 10 - номер строки
- 00000006 - адрес
- 7403 - машинный код
- jz finished - код программы

строка 11

- 11 - номер строки

- 00000008 - адрес
- 40 - машинный код
- inc eax - код программы

строка 12

- 12 - номер строки
- 00000009 - адрес
- EBF8 - машинный код
- jmp nextchar - код программы

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга (рис. 2.10,2.11)

```
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-2.asm -l lab8-2.lst
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:23: error: invalid combination of opcode and operands
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$
```

Рис. 2.10: ошибка трансляции lab8-2

```
lab8-2.lst                                lab8-2.asm
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 mov [B], ; запись преобразованного числа в 'B'
199 23 ***** error: invalid combination of opcode and operands
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 0000010B 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
202 26 00000111 890D[00000000] mov [max],ecx ; 'max = A'
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 00000117 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
205 29 0000011D 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
206 30 0000011F 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
207 31 00000125 890D[00000000] mov [max],ecx ; 'max = C'
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 0000012B B8[00000000] mov eax,max
211 35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
212 36 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013A 8B0D[00000000] mov ecx,[max]
215 39 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
216 40 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
217 41 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
218 42 0000014E 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
```

Рис. 2.11: файл листинга с ошибкой lab8-2

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу (рис. 2.12,2.13)

для варианта 14 - 81, 22, 72


```

lab8-3.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/

33  mov eax,B
34  call atoi
35  mov [B],eax
36
37  mov eax,msgC
38  call sprint
39  mov ecx,C
40  mov edx,80
41  call sread
42  mov eax,C
43  call atoi
44  mov [C],eax
45 ; _____algorithm_____
46
47  mov ecx,[A] ;ecx = A
48  mov [min],ecx ;min = A
49
50  cmp ecx, [B] ; A&B
51  jl check_C ; if a<b: goto check_C
52  mov ecx, [B]
53  mov [min], ecx ;else min = B
54
55 check_C:
56  cmp ecx, [C]
57  jl finish
58  mov ecx,[C]
59  mov [min],ecx
60
61 finish:
62  mov eax,answer
63  call sprint
64
65  mov eax, [min]
66  call iprintLF
67
68  call quit
69
70

```

Рис. 2.12: Файл lab8-3.asm

```

migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-3.asm
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ld -m elf_i386 -o lab8-3 lab8-3.o
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-3
Input A: 81
Input B: 22
Input C: 72
Smallest: 22
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$

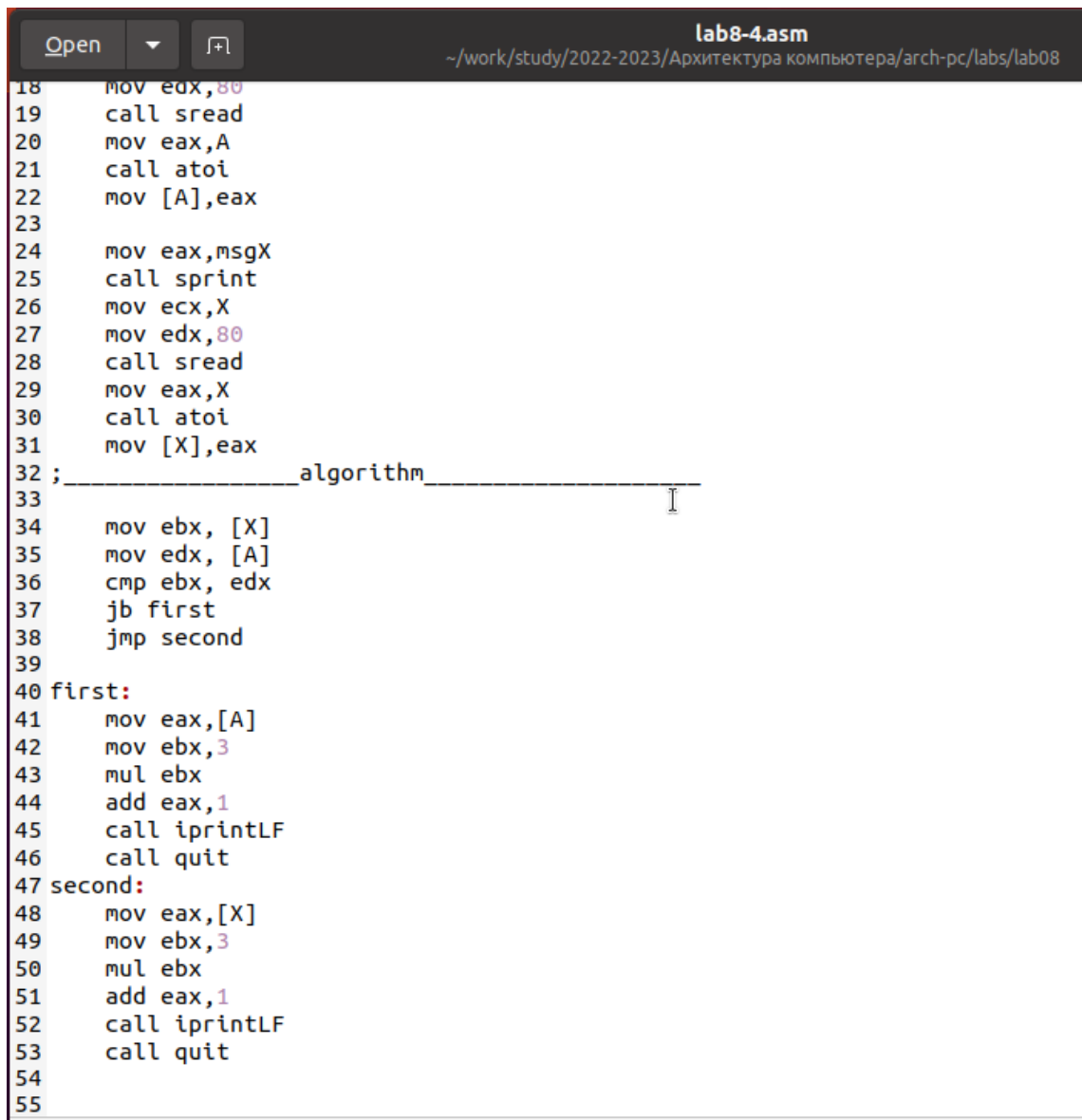
```

Рис. 2.13: Программа lab8-3.asm

6. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 8.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 8.6. (рис. 2.14,2.15)

для варианта 14

$$\begin{cases} 3a + 1, x < a \\ 3x + 1, x \geq a \end{cases}$$



```
lab8-4.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

18  mov edx,80
19  call sread
20  mov eax,A
21  call atoi
22  mov [A],eax
23
24  mov eax,msgX
25  call sprint
26  mov ecx,X
27  mov edx,80
28  call sread
29  mov eax,X
30  call atoi
31  mov [X],eax
32 ; _____algorithm_____
33
34  mov ebx, [X]
35  mov edx, [A]
36  cmp ebx, edx
37  jb first
38  jmp second
39
40 first:
41  mov eax,[A]
42  mov ebx,3
43  mul ebx
44  add eax,1
45  call iprintLF
46  call quit
47 second:
48  mov eax,[X]
49  mov ebx,3
50  mul ebx
51  add eax,1
52  call iprintLF
53  call quit
54
55
```

Рис. 2.14: Файл lab8-4.asm

```
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ nasm -f elf lab8-4.asm
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ld -m elf_i386 -o lab8-4 lab8-4.o
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-4
Input A: 3
Input X: 2
10
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$ ./lab8-4
Input A: 2
Input X: 4
13
migel@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab0
8$
```

Рис. 2.15: Программа lab8-4.asm

3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.