

Отчёта по лабораторной работе 8

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

ОНВУДИВЕ ВИКТОР ЧИБУИКЕ!

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	24
	Список литературы	25

Список иллюстраций

4.1	Файл lab8-1.asm:	9
4.2	Программа lab8-1.asm:	10
4.3	Файл lab8-1.asm:	11
4.4	Программа lab8-1.asm:	12
4.5	Файл lab8-1.asm	12
4.6	Программа lab8-1.asm	13
4.7	Файл lab8-2.asm	14
4.8	Программа lab8-2.asm	15
4.9	Файл листинга lab8-2	16
4.10	ошибка трансляции lab8-2	18
4.11	файл листинга с ошибкой lab8-2	19
4.12	Файл lab8-3.asm	20
4.13	Программа lab8-3.asm	21
4.14	Файл lab8-4.asm	22
4.15	Программа lab8-4.asm	23

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 Задание

1. Изучите примеры программ.
2. Изучите файл листинга.
3. 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
4. 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.

3 Теоретическое введение

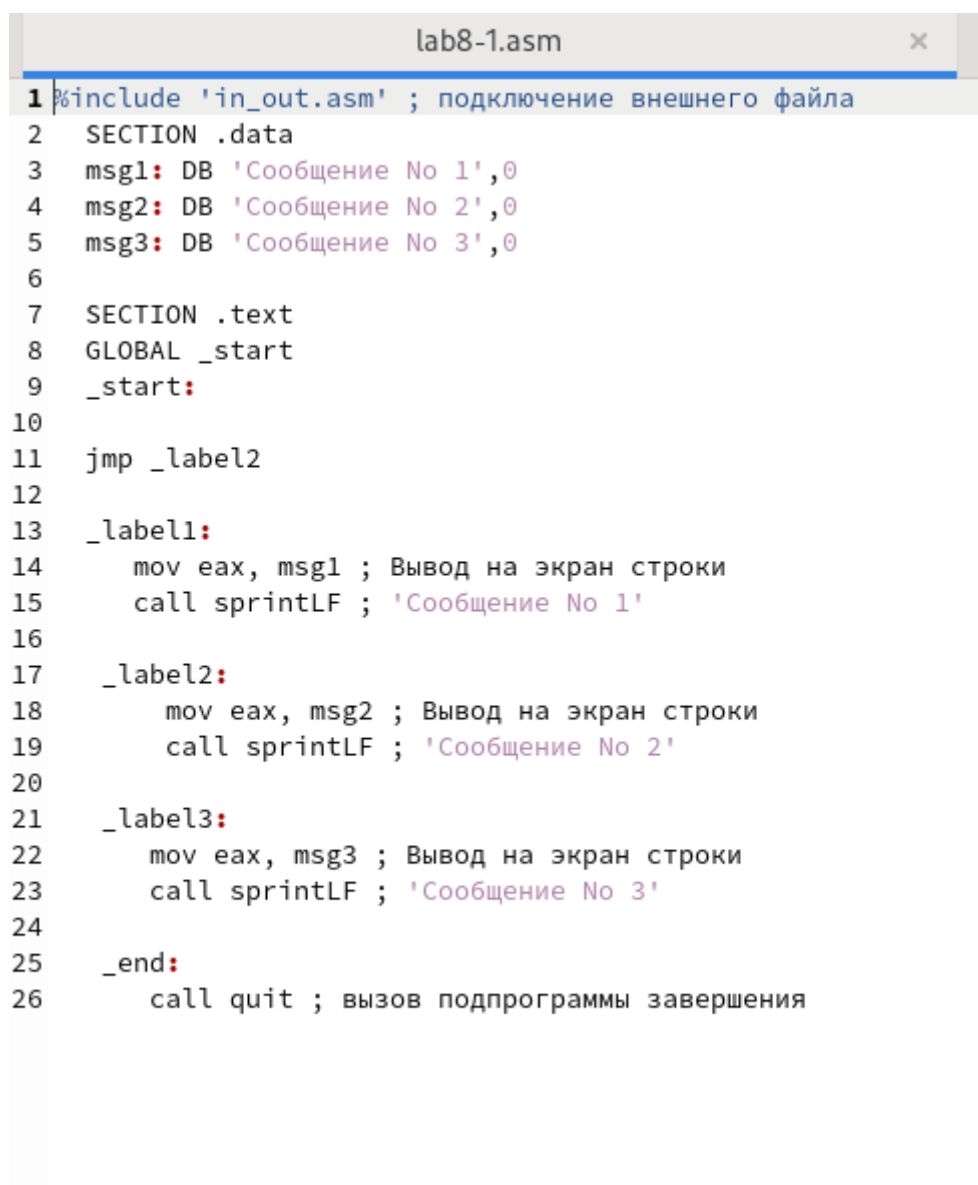
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

4 Выполнение лабораторной работы

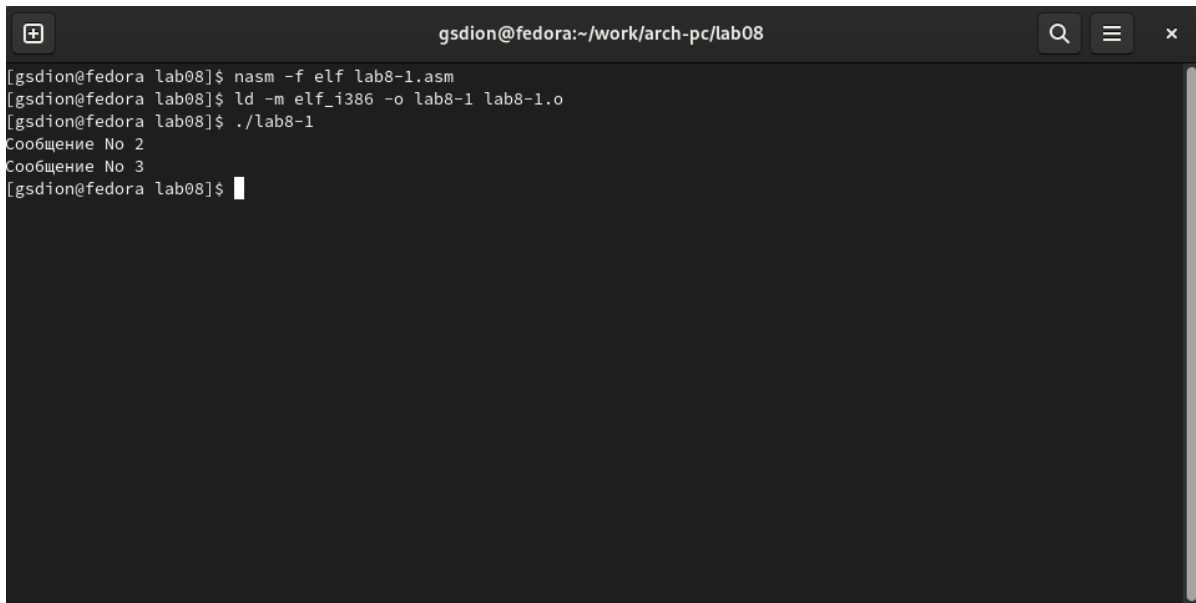
1. Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введите в файл lab8-1.asm текст программы из листинга 8.1. (рис. 4.1)



```
lab8-1.asm
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение No 1',0
4 msg2: DB 'Сообщение No 2',0
5 msg3: DB 'Сообщение No 3',0
6
7 SECTION .text
8 GLOBAL _start
9 _start:
10
11 jmp _label2
12
13 _label1:
14     mov eax, msg1 ; Вывод на экран строки
15     call sprintfLF ; 'Сообщение No 1'
16
17 _label2:
18     mov eax, msg2 ; Вывод на экран строки
19     call sprintfLF ; 'Сообщение No 2'
20
21 _label3:
22     mov eax, msg3 ; Вывод на экран строки
23     call sprintfLF ; 'Сообщение No 3'
24
25 _end:
26     call quit ; вызов подпрограммы завершения
```

Рис. 4.1: Файл lab8-1.asm:

Создайте исполняемый файл и запустите его. (рис. 4.2)

A terminal window with a dark background. The title bar shows 'gsdion@fedora:~/work/arch-pc/lab08'. The terminal contains the following text:

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

Рис. 4.2: 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. 4.3, 4.4)

```
lab8-1.asm ×
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение No 1',0
4 msg2: DB 'Сообщение No 2',0
5 msg3: DB 'Сообщение No 3',0
6
7 SECTION .text
8 GLOBAL _start
9 _start:
10
11 jmp _label2
12
13 _label1:
14     mov eax, msg1 ; Вывод на экран строки
15     call sprintf ; 'Сообщение No 1'
16     jmp _end
17
18 _label2:
19     mov eax, msg2 ; Вывод на экран строки
20     call sprintf ; 'Сообщение No 2'
21     jmp _label1
22
23 _label3:
24     mov eax, msg3 ; Вывод на экран строки
25     call sprintf ; 'Сообщение No 3'
26
27 _end:
28     call quit ; вызов подпрограммы завершения
```

Рис. 4.3: Файл lab8-1.asm:

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

Рис. 4.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. 4.5, 4.6):

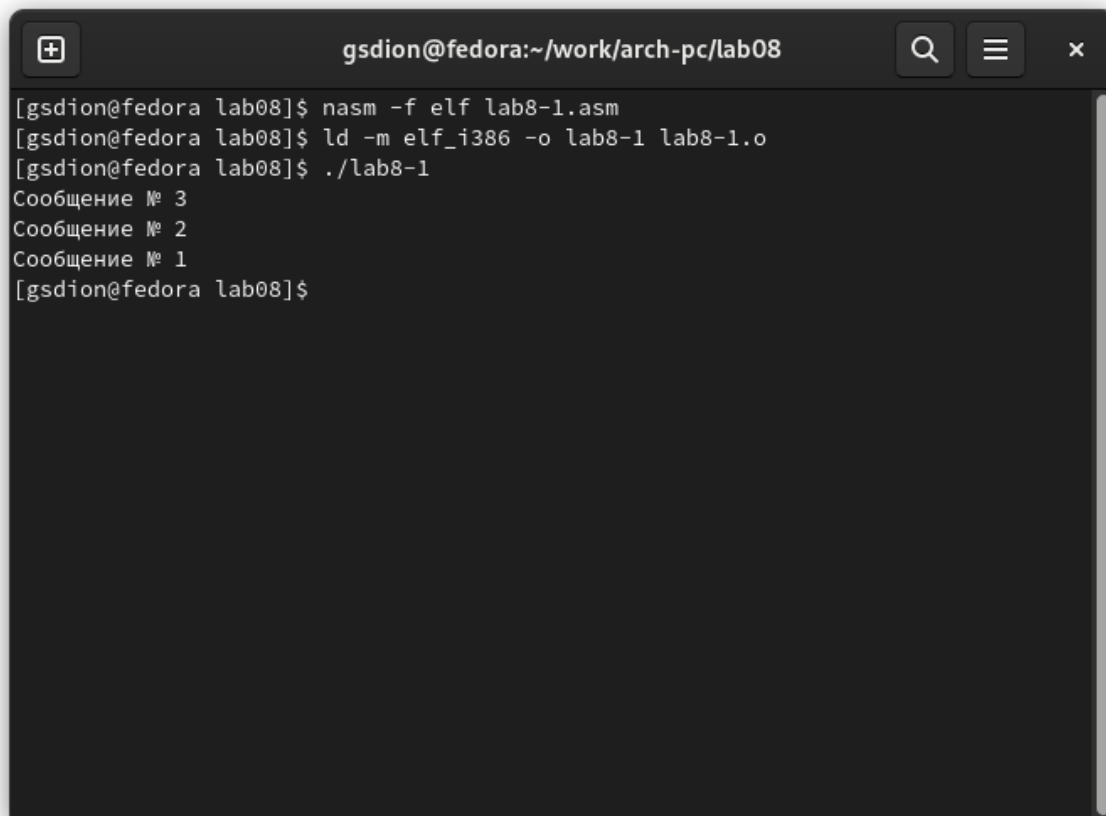
Сообщение № 3

Сообщение № 2

Сообщение № 1

```
Ouvrir  + lab8-1.asm
~/work/arch-pc/lab08
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 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 jmp _label2
26
27 _end:
28 call quit ; вызов подпрограммы завершения
```

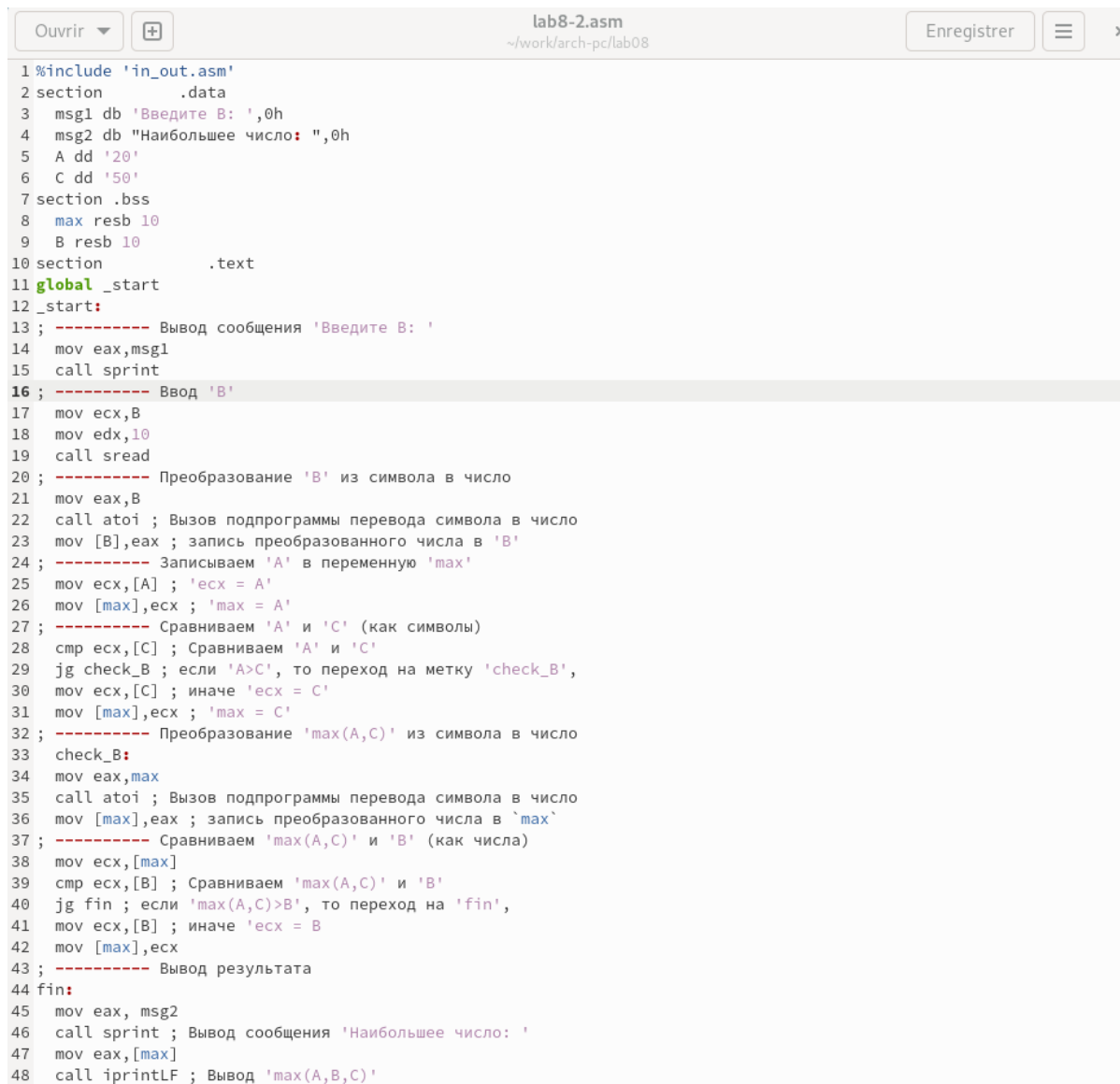
Рис. 4.5: Файл lab8-1.asm

A terminal window with a dark background. The title bar shows the user 'gsdion' on a 'fedora' machine, in the directory '~/work/arch-pc/lab08'. The terminal contains the following text:

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

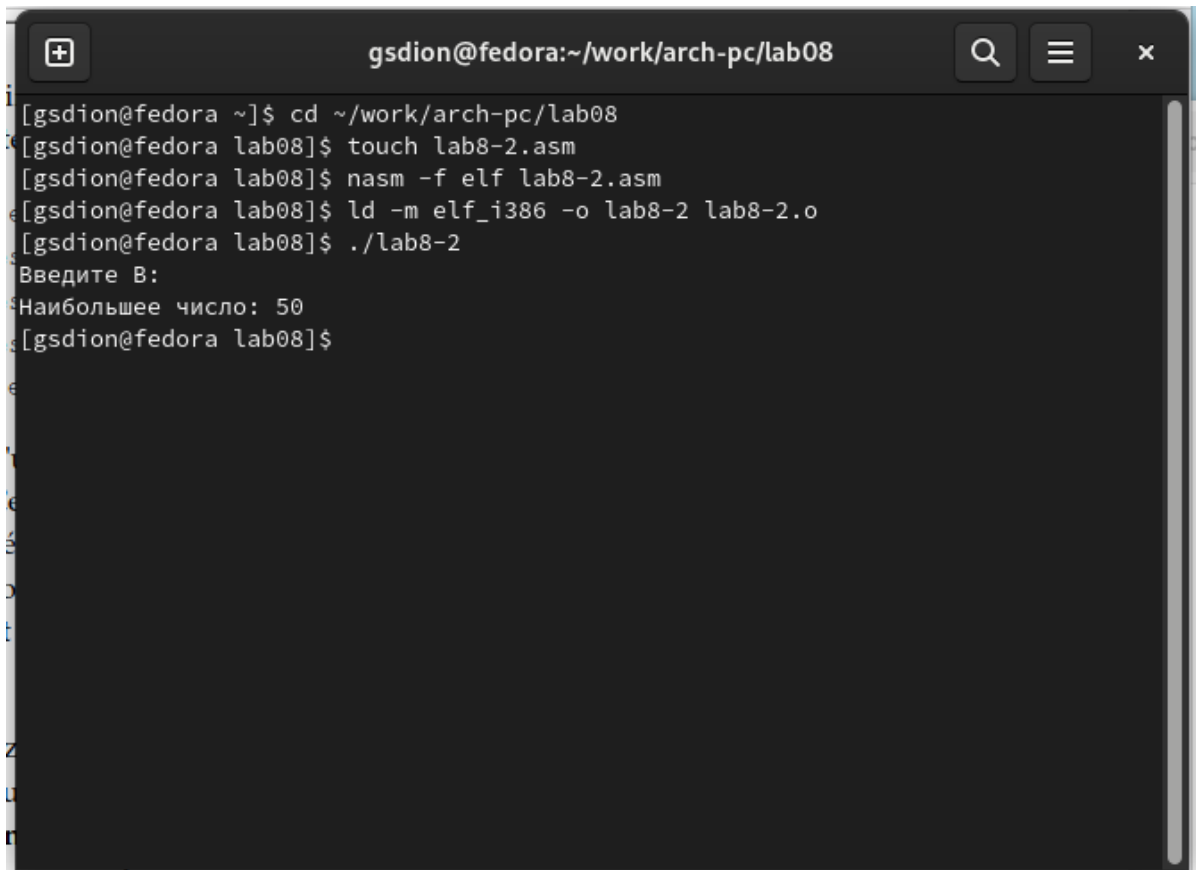
Рис. 4.6: Программа lab8-1.asm

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. 4.7, 4.8)



```
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)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
```

Рис. 4.7: Файл lab8-2.asm

A terminal window titled "gsdion@fedora:~/work/arch-pc/lab08" with search, menu, and close icons. The terminal shows the following commands and output:

```
[gsdion@fedora ~]$ cd ~/work/arch-pc/lab08
[gsdion@fedora lab08]$ touch lab8-2.asm
[gsdion@fedora lab08]$ nasm -f elf lab8-2.asm
[gsdion@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[gsdion@fedora lab08]$ ./lab8-2
Введите В:
Наибольшее число: 50
[gsdion@fedora lab08]$
```

Рис. 4.8: 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. 4.9)

```

Ouvrir  + lab8-2.lst ~\work\arch-pc\lab08 Enregistrer
1 1 %include 'in_out.asm'
2 2 <1> ;----- slen -----
3 3 <1> ; Функция вычисления длины сообщения
4 4 <1> slen:
5 5 00000000 53 <1> push ebx
6 6 00000001 89C3 <1> mov ebx, eax
7 7 <1>
8 8 <1> nextchar:
9 9 00000003 803800 <1> cmp byte [eax], 0
10 10 00000006 7403 <1> jz finished
11 11 00000008 40 <1> inc eax
12 12 00000009 EBF8 <1> jmp nextchar
13 13 <1>
14 14 <1> finished:
15 15 0000000B 29D8 <1> sub eax, ebx
16 16 0000000D 5B <1> pop ebx
17 17 0000000E C3 <1> ret
18 18 <1>
19 19 <1>
20 20 <1> ;----- sprint -----
21 21 <1> ; Функция печати сообщения
22 22 <1> ; входные данные: mov eax,<message>
23 23 <1> sprint:
24 24 0000000F 52 <1> push edx
25 25 00000010 51 <1> push ecx
26 26 00000011 53 <1> push ebx
27 27 00000012 50 <1> push eax
28 28 00000013 E8E8FFFFFF <1> call slen
29 29 <1>
30 30 00000018 89C2 <1> mov edx, eax
31 31 0000001A 58 <1> pop eax
32 32 <1>
33 33 0000001B 89C1 <1> mov ecx, eax
34 34 0000001D BB01000000 <1> mov ebx, 1
35 35 00000022 B804000000 <1> mov eax, 4
36 36 00000027 CD80 <1> int 80h
37 37 <1>
38 38 00000029 5B <1> pop ebx
39 39 0000002A 59 <1> pop ecx
40 40 0000002B 5A <1> pop edx
41 41 0000002C C3 <1> ret
42 42 <1>
43 43 <1>
44 44 <1> ;----- sprintLF -----
45 45 <1> ; Функция печати сообщения с переводом строки
46 46 <1> ; входные данные: mov eax,<message>
47 47 <1> sprintLF:
48 48 0000002D E8DDFFFFFF <1> call sprint

```

Texte brut L largeur des tabulations : 8 l in 38. Col 56 INS

Рис. 4.9: Файл листинга lab8-2

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

строка 144

- 144 - номер строки
- 000000BB - адрес
- 80EB30 - машинный код

- `sub bl, 48` - код программы

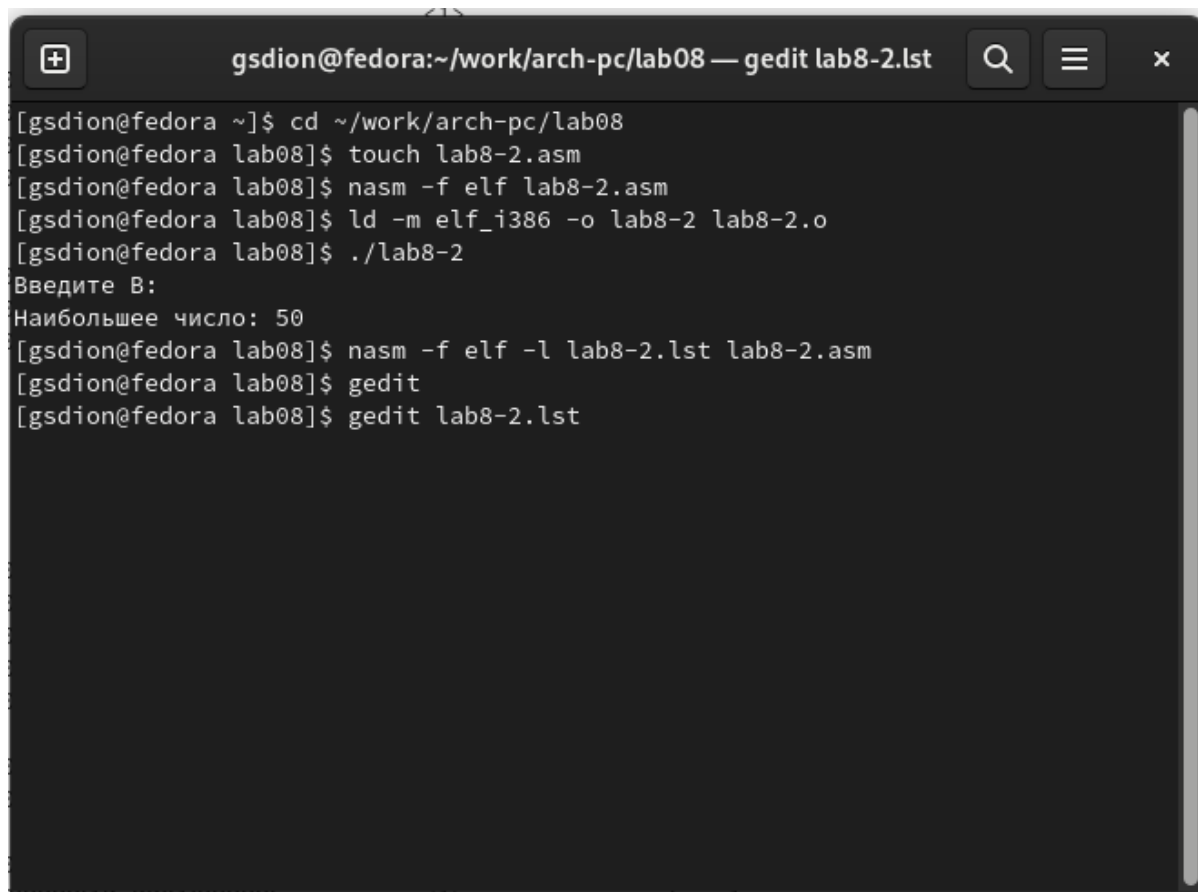
строка 145

- 145 - номер строки
- 000000BE - адрес
- 01D8 - машинный код
- `add eax, ebx` - код программы

строка 146

- 146 - номер строки
- 000000C0 - адрес
- BB0A000000 - машинный код
- `mov ebx, 10` - 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. 4.10,4.11)



The image shows a terminal window titled "gsdion@fedora:~/work/arch-pc/lab08 — gedit lab8-2.lst". The terminal contains the following commands and output:

```
[gsdion@fedora ~]$ cd ~/work/arch-pc/lab08
[gsdion@fedora lab08]$ touch lab8-2.asm
[gsdion@fedora lab08]$ nasm -f elf lab8-2.asm
[gsdion@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[gsdion@fedora lab08]$ ./lab8-2
Введите В:
Наибольшее число: 50
[gsdion@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
[gsdion@fedora lab08]$ gedit
[gsdion@fedora lab08]$ gedit lab8-2.lst
```

Рис. 4.10: ошибка трансляции lab8-2

```

1      1      %include 'in_out.asm'
2      2      <1> ;----- slen -----
3      3      <1> ; Функция вычисления длины сообщения
4      4      <1> slen:
5      5      5 00000000 53      <1>      push    ebx
6      6      6 00000001 89C3    <1>      mov     ebx, eax
7      7      7
8      8      8
9      9      9 00000003 803800    <1>      cmp     byte [eax], 0
10     10 10 00000006 7403      <1>      jz      finished
11     11 11 00000008 40        <1>      inc     eax
12     12 12 00000009 EBF8      <1>      jmp     nextchar
13     13 13
14     14 14
15     15 15 0000000B 29D8      <1>      sub     eax, ebx
16     16 16 0000000D 5B        <1>      pop     ebx
17     17 17 0000000E C3        <1>      ret
18     18 18
19     19 19
20     20 20      <1> ;----- sprint -----
21     21 21      <1> ; Функция печати сообщения
22     22 22      <1> ; входные данные: mov eax,<message>
23     23 23      <1> sprint:
24     24 24 0000000F 52        <1>      push    edx
25     25 25 00000010 51        <1>      push    ecx
26     26 26 00000011 53        <1>      push    ebx
27     27 27 00000012 50        <1>      push    eax
28     28 28 00000013 E8E8FFFF    <1>      call    slen
29     29 29
30     30 30 00000018 89C2      <1>      mov     edx, eax
31     31 31 0000001A 58        <1>      pop     eax
32     32 32
33     33 33 0000001B 89C1      <1>      mov     ecx, eax
34     34 34 0000001D BB01000000 <1>      mov     ebx, 1
35     35 35 00000022 B804000000 <1>      mov     eax, 4
36     36 36 00000027 CD80      <1>      int     80h
37     37 37
38     38 38 00000029 5B        <1>      pop     ebx
39     39 39 0000002A 59        <1>      pop     ecx
40     40 40 0000002B 5A        <1>      pop     edx
41     41 41 0000002C C3        <1>      ret
42     42 42
43     43 43
44     44 44      <1> ;----- sprintf -----
45     45 45      <1> ; Функция печати сообщения с переводом строки
46     46 46      <1> ; входные данные: mov eax,<message>
47     47 47      <1> sprintf:
48     48 48 0000002D E8DDFFFF    <1>      call    sprint

```

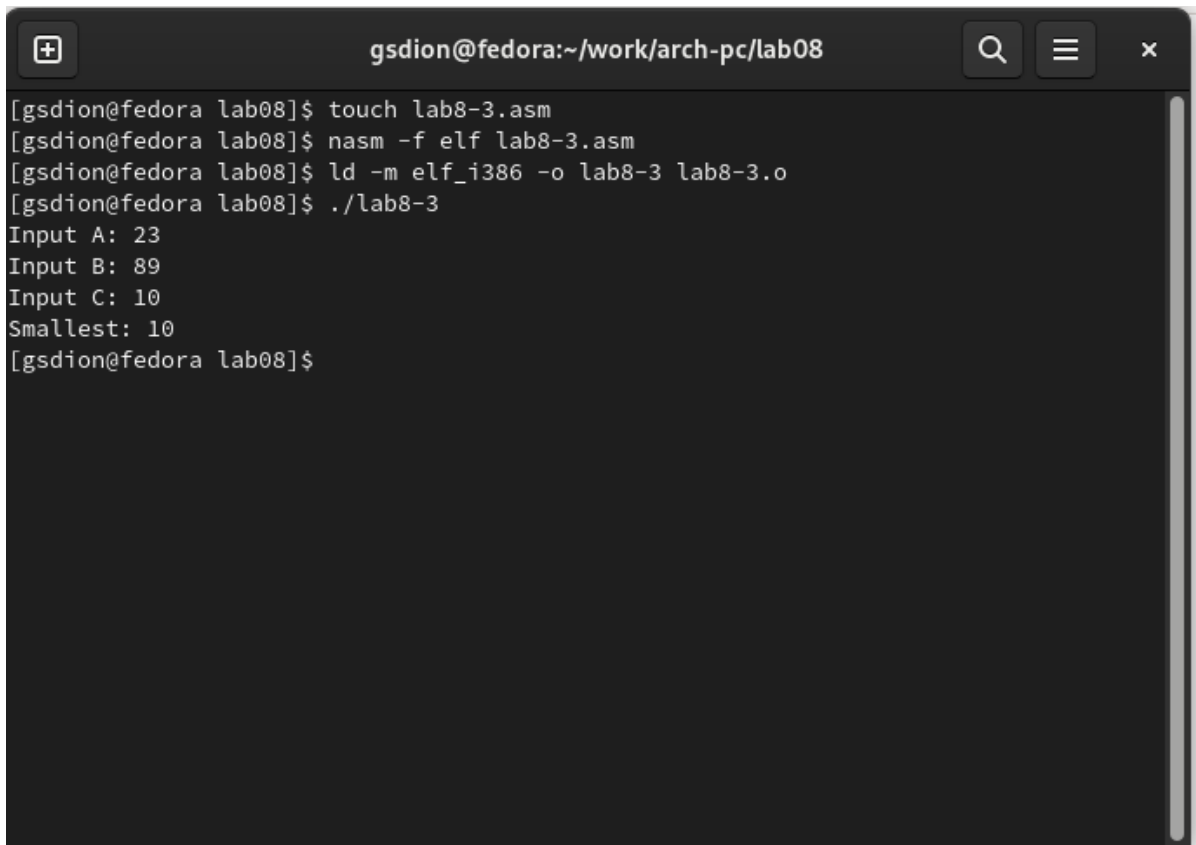
Рис. 4.11: файл листинга с ошибкой lab8-2

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. 4.12, 4.13)

для варианта 10 - 41, 62, 35

```
*report.md x lab8-3.asm x
18 _start:
19     mov eax,msgA
20     call sprint
21     mov ecx,A
22     mov edx,80
23     call sread
24     mov eax,A
25     call atoi
26     mov [A],eax
27
28     mov eax, msgB
29     call sprint
30     mov ecx,B
31     mov edx,80
32     call sread
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 ecx,answer
```

Рис. 4.12: Файл lab8-3.asm



```
gsdion@fedora:~/work/arch-pc/lab08
[gsdion@fedora lab08]$ touch lab8-3.asm
[gsdion@fedora lab08]$ nasm -f elf lab8-3.asm
[gsdion@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[gsdion@fedora lab08]$ ./lab8-3
Input A: 23
Input B: 89
Input C: 10
Smallest: 10
[gsdion@fedora lab08]$
```

Рис. 4.13: 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. 4.14,4.15)

для варианта 10

$$\begin{cases} a - 7, a \geq 7 \\ ax, a < 7 \end{cases}$$

report.md	lab8-3.asm	lab8-4.asm
-----------	------------	------------

```
6 SECTION .bss
7     A: RESB 80
8     X: RESB 80
9     result: RESB 80
10
11 SECTION .text
12     GLOBAL _start
13
14 _start:
15     mov eax,msgA
16     call sprint
17     mov ecx,A
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, [A]
35     cmp ebx, 7
36     ja first
37     jmp second
38
39 first:
40     mov eax,[A]
41     sub eax,7
42     call iprintLF
43     call quit
44 second:
45     mov eax, [X]
46     mov ebx,[A]
47     mul ebx
48     call iprintLF
49     call quit
50
```

Рис. 4.14: Файл lab8-4.asm

```
smallest: 10
[gsdion@fedora lab08]$ touch lab8-4.asm
[gsdion@fedora lab08]$ nasm -f elf lab8-4.asm
lab8-4.asm:46: error: symbol `a' not defined
[gsdion@fedora lab08]$ nasm -f elf lab8-4.asm
[gsdion@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[gsdion@fedora lab08]$ ./lab8-4
Input A: 9
Input X: 3
2
[gsdion@fedora lab08]$ ./lab8-4
Input A: 4
Input X: 6
24
[gsdion@fedora lab08]$
```

Рис. 4.15: Программа lab8-4.asm

5 Выводы

В заключение мы изучили команды условного и безусловного перехода и узнали о файле листинга.

Список литературы

1. Расширенный ассемблер: NASM
2. MASM, TASM, FASM, NASM под Windows и Linux