

Отчёт по лабораторной работе 7

дисциплина: Архитектура компьютера

Кылыш Гоктюрк

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Выводы	20

Список иллюстраций

2.1 Программа lab7-1.asm	7
2.2 Запуск программы lab7-1.asm	8
2.3 Программа lab7-1.asm	9
2.4 Запуск программы lab7-1.asm	9
2.5 Программа lab7-1.asm	10
2.6 Запуск программы lab7-1.asm	11
2.7 Программа lab7-2.asm	12
2.8 Запуск программы lab7-2.asm	13
2.9 Файл листинга lab7-2	13
2.10 Ошибка трансляции lab7-2	14
2.11 Файл листинга с ошибкой lab7-2	15
2.12 Программа lab7-3.asm	16
2.13 Запуск программы lab7-3.asm	16
2.14 Программа lab7-4.asm	18
2.15 Запуск программы lab7-4.asm	19

Список таблиц

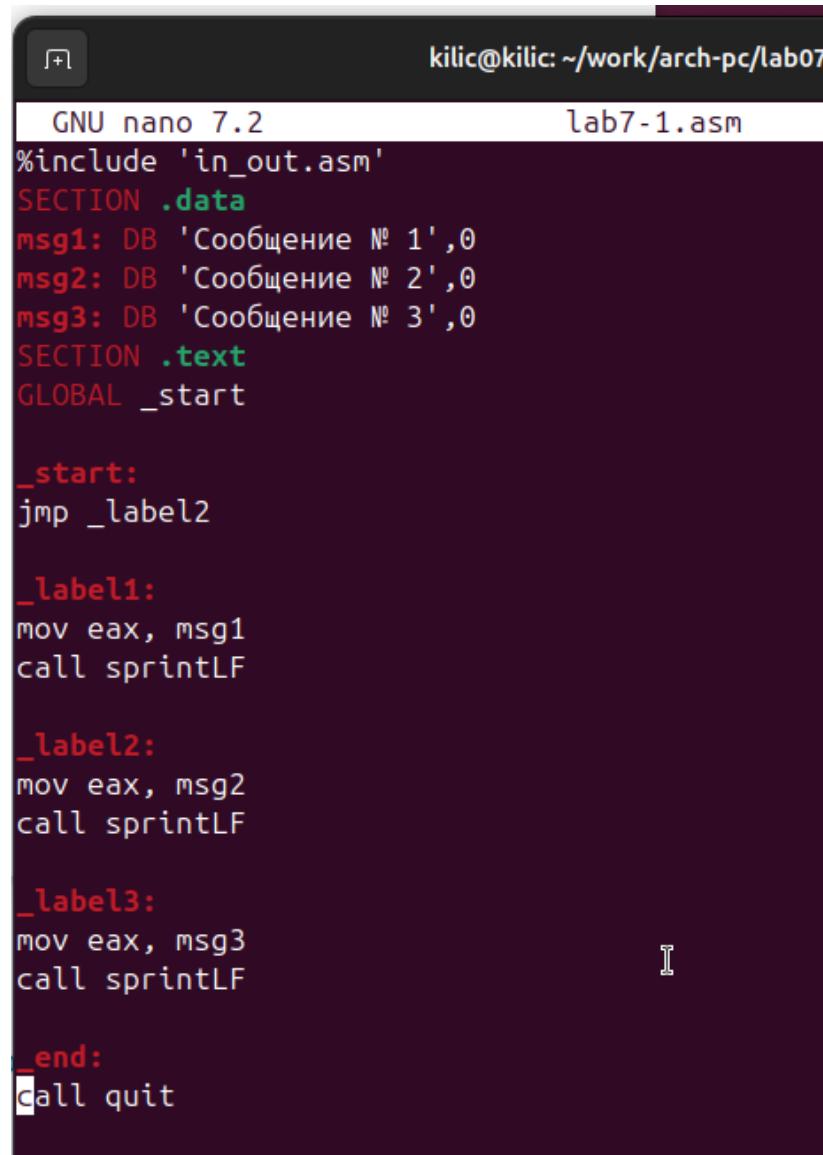
1 Цель работы

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. Создал каталог для программам лабораторной работы № 7 и файл lab7-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp.

Написал в файл lab7-1.asm текст программы из листинга 7.1.



The screenshot shows a terminal window with the title bar "kilic@kilic: ~/work/arch-pc/lab07". The window contains assembly code for a program named "lab7-1.asm". The code includes three data labels (msg1, msg2, msg3) containing Russian strings, a text section starting at _start, and three labels (_label1, _label2, _label3) each followed by a mov eax, msg... instruction and a call sprintLF. The _end label ends with a quit instruction. The assembly code uses mnemonics like DB, SECTION, GLOBAL, jmp, mov, call, and quit.

```
GNU nano 7.2                      lab7-1.asm
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

_label1:
mov eax, msg1
call sprintLF

_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рисунок 2.1: Программа lab7-1.asm

Создал исполняемый файл и запустил его.

```
kilic@kilic:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
kilic@kilic:~/work/arch-pc/lab07$ ld -m elf lab7-1.o -o lab7-1
ld: unrecognised emulation mode: elf
Supported emulations: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu i38
6pep i386pe
kilic@kilic:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
kilic@kilic:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
kilic@kilic:~/work/arch-pc/lab07$ █
```

Рисунок 2.2: Запуск программы lab7-1.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).

Изменил текст программы в соответствии с листингом 7.2.

The screenshot shows a terminal window with the title "kilic@kilic: ~/work/arch-pc/lab07". Inside the terminal, the file "lab7-1.asm" is being edited in the nano text editor. The code is an assembly program that prints three messages ("Сообщение № 1", "Сообщение № 2", and "Сообщение № 3") using the `sprintLF` function. It includes sections for data and text, labels for each message, and a loop structure. The assembly instructions use the AT&T syntax.

```
GNU nano 7.2          lab7-1.asm
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

_label1:
    mov eax, msg1
    call sprintLF
    jmp _end

_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1

_label3:
    mov eax, msg3
    call sprintLF

_end:
call quit
```

Рисунок 2.3: Программа lab7-1.asm

The screenshot shows a terminal window with the title "kilic@kilic: ~/work/arch-pc/lab07". The user runs the command "nano lab7-1.asm" to edit the source code. Then, they run "nasm -f elf lab7-1.asm" to assemble it into an ELF file. Next, they run "ld -m elf_i386 lab7-1.o -o lab7-1" to link the object file into an executable. Finally, they run "./lab7-1" to execute the program, which prints the three messages sequentially. The terminal uses color-coded syntax highlighting for assembly code.

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

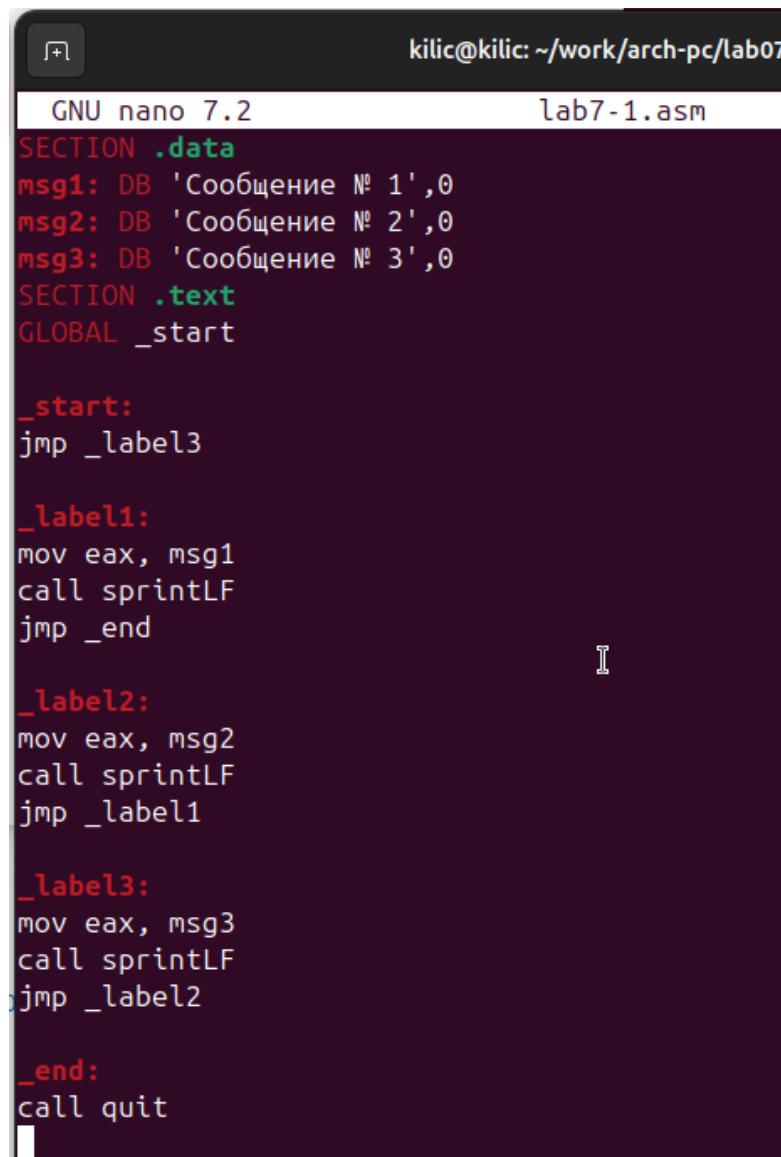
Рисунок 2.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции jmp, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1



The screenshot shows a terminal window titled "kilic@kilic: ~/work/arch-pc/lab07". The file being edited is "lab7-1.asm". The assembly code is as follows:

```
GNU nano 7.2          lab7-1.asm
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
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

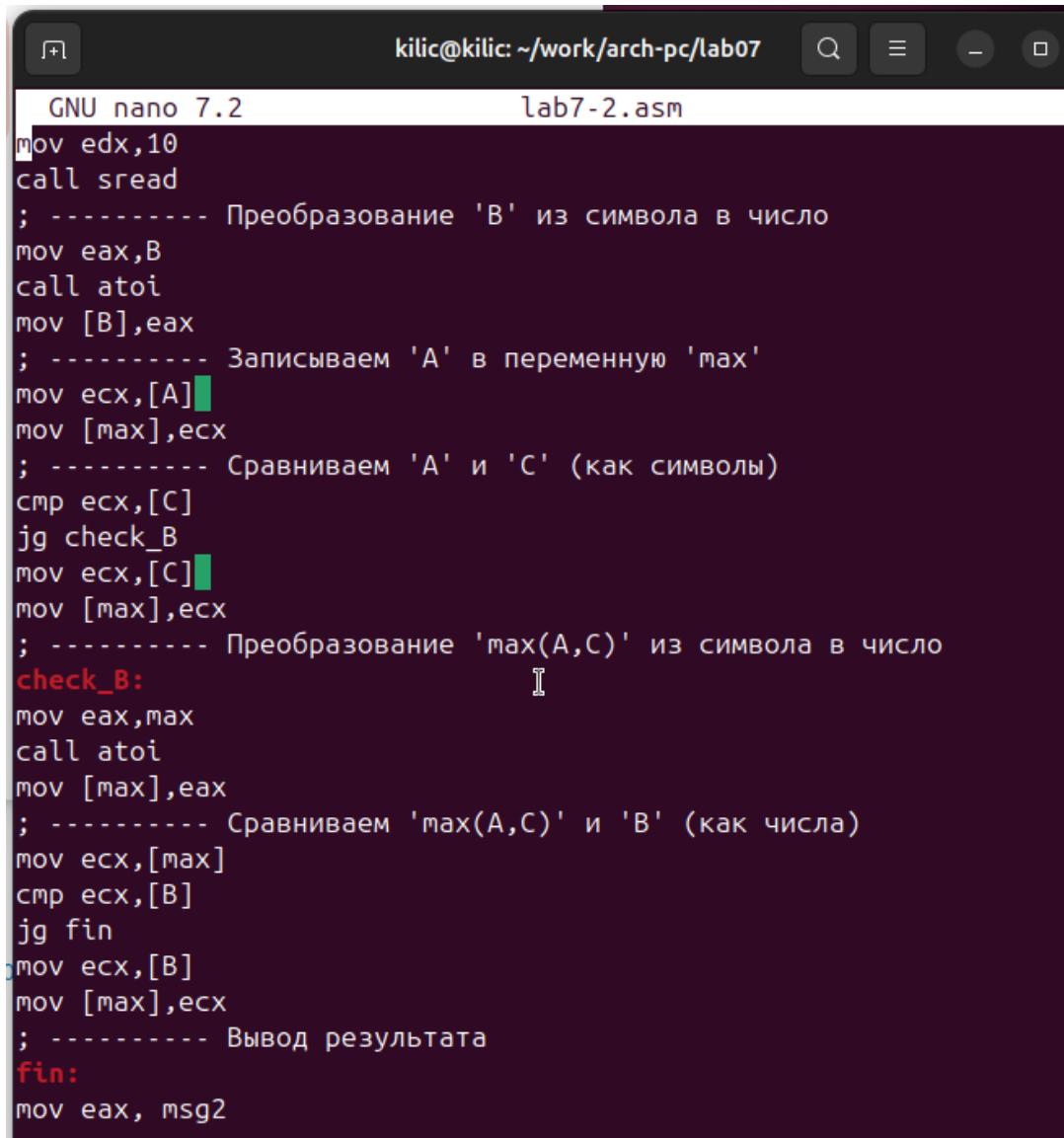
Рисунок 2.5: Программа lab7-1.asm

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

Рисунок 2.6: Запуск программы lab7-1.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: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений B.



The screenshot shows a terminal window titled "GNU nano 7.2" with the file name "lab7-2.asm". The assembly code is as follows:

```
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
    mov eax,max
    call atoi
    mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
    mov ecx,[max]
    cmp ecx,[B]
    jg fin
    mov ecx,[B]
    mov [max],ecx
; ----- Вывод результата
fin:
    mov eax, msg2
```

Рисунок 2.7: Программа lab7-2.asm

```

kilic@kilic:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kilic@kilic:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
kilic@kilic:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
kilic@kilic:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 70
Наибольшее число: 70
kilic@kilic:~/work/arch-pc/lab07$ 

```

Рисунок 2.8: Запуск программы lab7-2.asm

- Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке.

Создал файл листинга для программы из файла lab7-2.asm

```

GNU nano 7.2                                lab7-2.lst
22 00000106 E891FFFF
23 0000010B A3[0A000000]
24
25 00000110 8B0D[35000000]
26 00000116 890D[00000000]
27
28 0000011C 3B0D[39000000]
29 00000122 7F0C
30 00000124 8B0D[39000000]
31 0000012A 890D[00000000]
32
33
34 00000130 B8[00000000]
35 00000135 E862FFFFFF
36 0000013A A3[00000000]
37
38 0000013F 8B0D[00000000]
39 00000145 3B0D[0A000000]
40 0000014B 7F0C
41 0000014D 8B0D[0A000000]
42 00000153 890D[00000000]
43
44
45 00000159 B8[13000000]
46 0000015E E8ACFEFFFF
47 00000163 A1[00000000]
48 00000168 E819FFFFFF
49 0000016D E869FFFFFF
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
    mov eax,max
    call atoi
    mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
    mov ecx,[max]
    cmp ecx,[B]
    jg fin
    mov ecx,[B]
    mov [max],ecx
; ----- Вывод результата
fin:
    mov eax, msg2
    call sprint
    mov eax,[max]
    call iprintLF
    call quit

```

Рисунок 2.9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объяс-

нию содержимое трёх строк файла листинга по выбору.

строка 34

- 34 - номер строки
- 00000130 - адрес
- B8[00000000] - машинный код
- mov eax,max - код программы

строка 35

- 35 - номер строки
- 00000135 - адрес
- E862FFFFFF - машинный код
- call atoi- код программы

строка 36

- 36 - номер строки
- 0000013A - адрес
- A3[00000000] - машинный код
- mov [max],eax - код программы

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один operand. Выполнил трансляцию с получением файла листинга.

```
kilic@kilic:~/work/arch-pc/lab07$  
kilic@kilic:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:14: error: invalid combination of opcode and operands  
kilic@kilic:~/work/arch-pc/lab07$ █
```

Рисунок 2.10: Ошибка трансляции lab7-2

```

GNU nano 7.2                                lab7-2.lst
 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                                         mov eax,
15                                         ****error: invalid combination of opcode and operands
16                                         call sprint
17                                         ; ----- Ввод 'B'
18                                         mov ecx,B
19                                         mov edx,10
20                                         call sread
21                                         ; ----- Преобразование 'B' из символа в число
22                                         mov eax,B
23                                         call atoi
24                                         mov [B],eax
25                                         ; ----- Записываем 'A' в переменную 'max'
26                                         mov ecx,[A]
27                                         mov [max],ecx
28                                         ; ----- Сравниваем 'A' и 'C' (как символы)
29                                         cmp ecx,[C]
30                                         jg check_B
31                                         .end

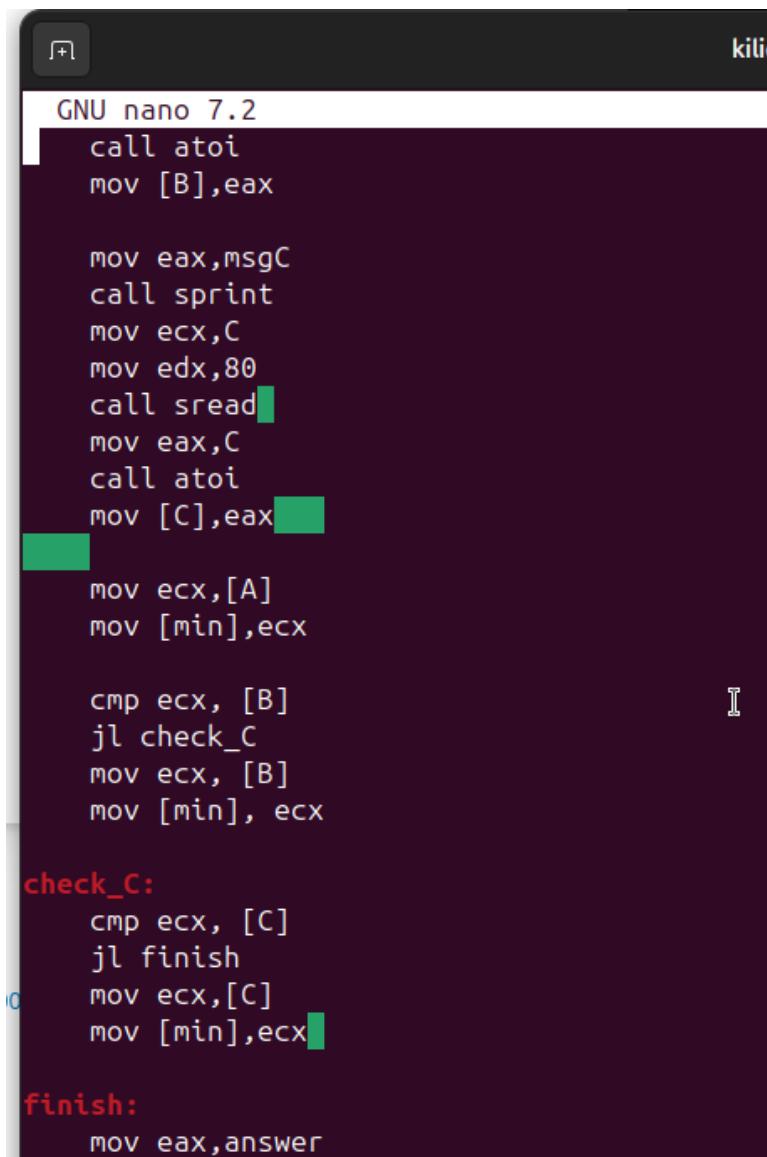
```

Рисунок 2.11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу

для варианта 1 - 17,23,45



GNU nano 7.2

```
    call atoi
    mov [B],eax

    mov eax,msgC
    call sprint
    mov ecx,C
    mov edx,80
    call sread
    mov eax,C
    call atoi
    mov [C],eax

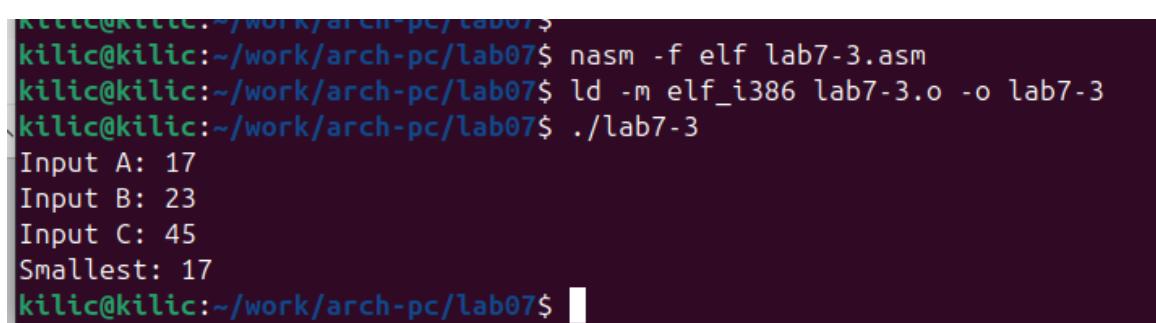
    mov ecx,[A]
    mov [min],ecx

    cmp ecx, [B]
    jl check_C
    mov ecx, [B]
    mov [min], ecx

check_C:
    cmp ecx, [C]
    jl finish
0:   mov ecx,[C]
    mov [min],ecx

finish:
    mov eax,answer
```

Рисунок 2.12: Программа lab7-3.asm



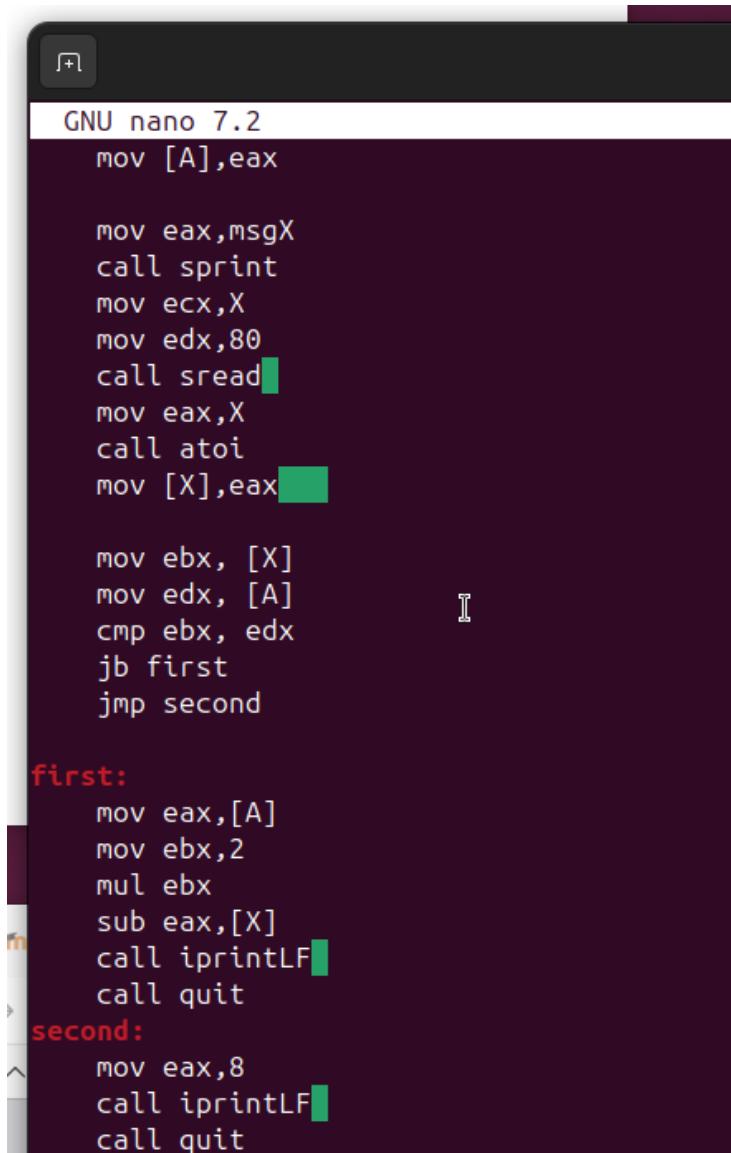
```
kilic@kilic:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
kilic@kilic:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
kilic@kilic:~/work/arch-pc/lab07$ ./lab7-3
Input A: 17
Input B: 23
Input C: 45
Smallest: 17
kilic@kilic:~/work/arch-pc/lab07$
```

Рисунок 2.13: Запуск программы lab7-3.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 7.6 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 7.6.

для варианта 1

$$\begin{cases} 2a - x, & x < a \\ 8, & a \geq 0 \end{cases}$$



The screenshot shows a terminal window with the title 'GNU nano 7.2'. The window contains the following assembly code:

```
GNU nano 7.2
mov [A],eax

    mov eax,msgX
    call sprint
    mov ecx,X
    mov edx,80
    call sread
    mov eax,X
    call atoi
    mov [X],eax

    mov ebx, [X]
    mov edx, [A]           ||
    cmp ebx, edx
    jb first
    jmp second

first:
    mov eax,[A]
    mov ebx,2
    mul ebx
    sub eax,[X]
    call iprintLF
    call quit
second:
    mov eax,8
    call iprintLF
    call quit
```

Рисунок 2.14: Программа lab7-4.asm

```
kilic@kilic:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
kilic@kilic:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
kilic@kilic:~/work/arch-pc/lab07$ ./lab7-4
Input A: 1
Input X: 2
8
kilic@kilic:~/work/arch-pc/lab07$ ./lab7-4
Input A: 2
Input X: 1
3
kilic@kilic:~/work/arch-pc/lab07$
```

Рисунок 2.15: Запуск программы lab7-4.asm

3 Выводы

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