

# Отчёт по лабораторной работе 7

Архитектура компьютера

Диденко Дмитрий Владимирович НПИбд-03-23

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM . . . . .	6
2.2	Изучение структуры файлы листинга . . . . .	13
2.3	Задание для самостоятельной работы . . . . .	17
3	Выводы	20

## Список иллюстраций

2.1	Код программы lab7-1.asm . . . . .	7
2.2	Компиляция и запуск программы lab7-1.asm . . . . .	7
2.3	Код программы lab7-1.asm . . . . .	9
2.4	Компиляция и запуск программы lab7-1.asm . . . . .	9
2.5	Код программы lab7-1.asm . . . . .	10
2.6	Компиляция и запуск программы lab7-1.asm . . . . .	11
2.7	Код программы lab7-2.asm . . . . .	12
2.8	Компиляция и запуск программы lab7-2.asm . . . . .	13
2.9	Файл листинга lab7-2 . . . . .	14
2.10	Ошибка трансляции lab7-2 . . . . .	15
2.11	Файл листинга с ошибкой lab7-2 . . . . .	16
2.12	Код программы prog-1.asm . . . . .	17
2.13	Компиляция и запуск программы prog-1.asm . . . . .	18
2.14	Код программы prog-2.asm . . . . .	19
2.15	Компиляция и запуск программы prog-2.asm . . . . .	19

## Список таблиц

# 1 Цель работы

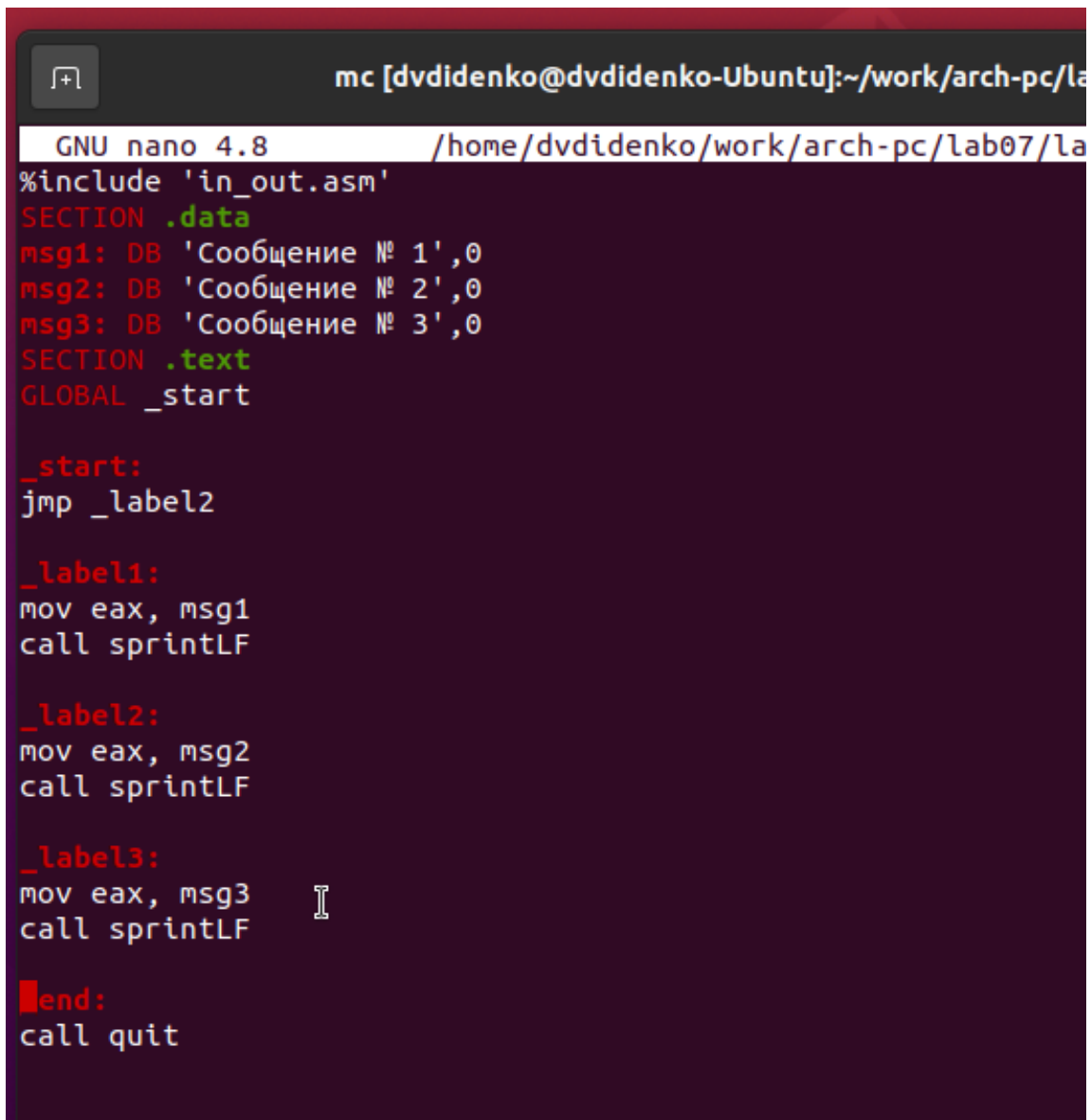
Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

### 2.1 Реализация переходов в NASM

Я создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.

Инструкция `jmp` в NASM используется для выполнения безусловных переходов. Рассмотрим пример программы, в которой используется инструкция `jmp`. Написал текст программы из листинга 7.1 в файле lab7-1.asm. (рис. [2.1])



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/la
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab07/la
%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

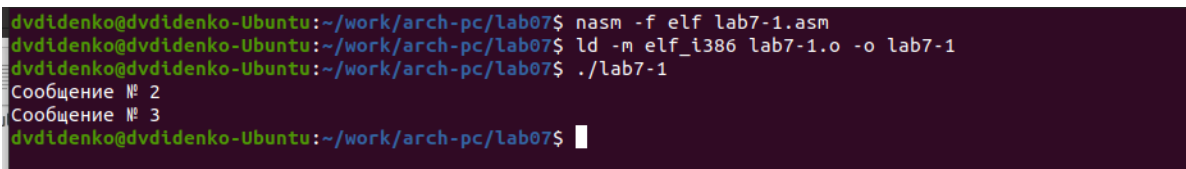
_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

end:
call quit
```

Рис. 2.1: Код программы lab7-1.asm

Создал исполняемый файл и запустил его. (рис. [2.2])



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

Рис. 2.2: Компиляция и запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Мы изменим программу так, чтобы она сначала выводила “Сообщение № 2”, затем “Сообщение № 1” и завершала работу. Для этого мы добавим в текст программы после вывода “Сообщения № 2” инструкцию `jmp` с меткой `_label1` (чтобы перейти к инструкциям вывода “Сообщения № 1”) и после вывода “Сообщения № 1” добавим инструкцию `jmp` с меткой `_end` (чтобы перейти к инструкции `call quit`).

Изменил текст программы в соответствии с листингом 7.2. (рис. [2.3] [2.4])



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/
GNU nano 4.8 /home/dvdidenko/work/arch-pc/
%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
```

Рис. 2.3: Код программы lab7-1.asm

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

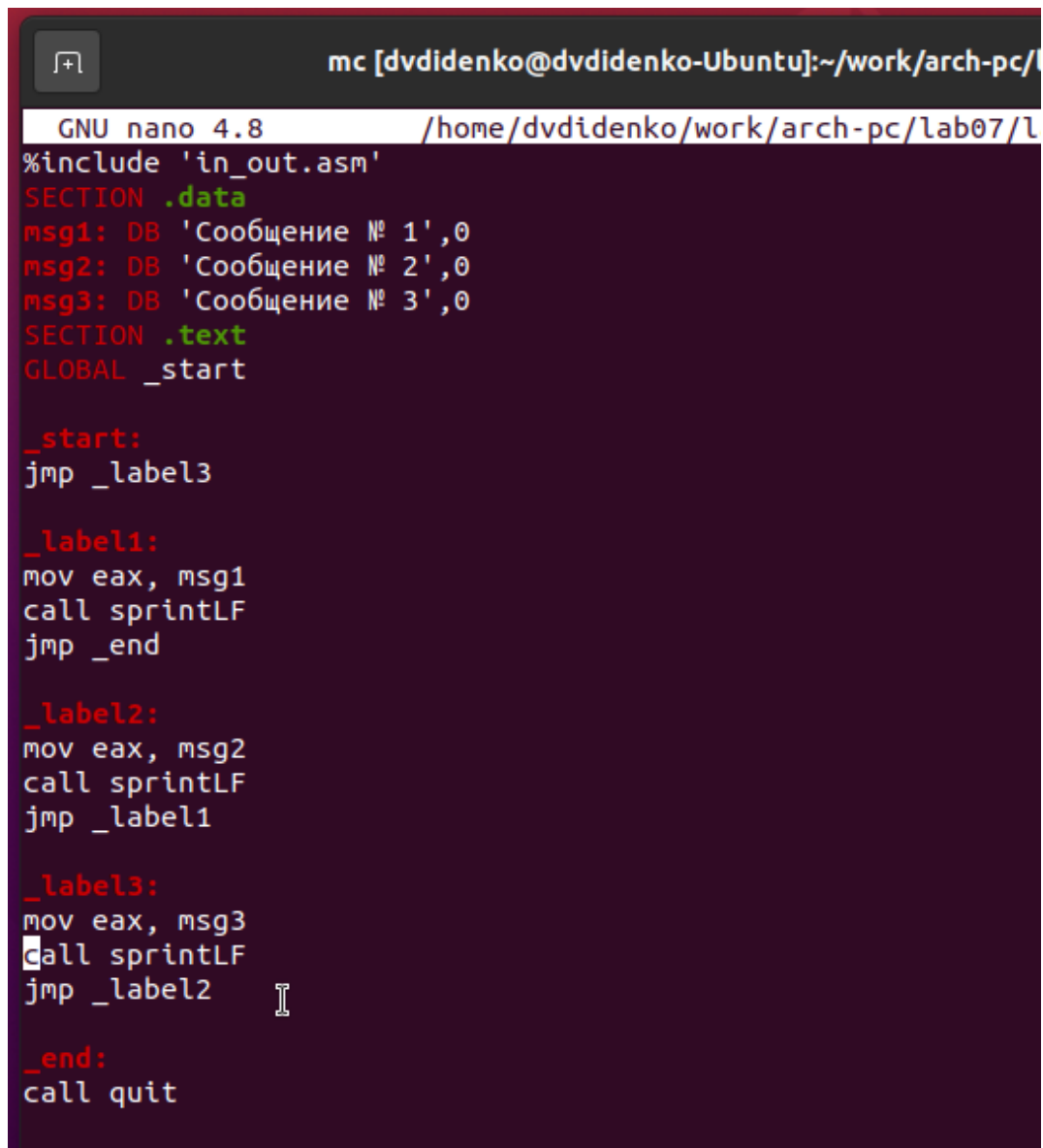
Рис. 2.4: Компиляция и запуск программы lab7-1.asm

Изменил текст программы (рис. [2.5] [2.6]), изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab07/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 _label3

_label1:
mov eax, msg1
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf
jmp _label2

_end:
call quit
```

Рис. 2.5: Код программы lab7-1.asm

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

Рис. 2.6: Компиляция и запуск программы lab7-1.asm

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, то есть переход должен происходить, если выполнено какое-либо условие.

Давайте рассмотрим программу, которая определяет и выводит на экран наибольшую из трех целочисленных переменных: А, В и С. Значения для А и С задаются в программе, а значение В вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений В. (рис. [2.7] [2.8])

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab07
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab07/lab7-2.asm
call sprint
; ----- Ввод 'B'
mov ecx,B
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
call sprint
mov eax,[max]
call iprintLF
call quit
```

Рис. 2.7: Код программы lab7-2.asm

```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 40
Наибольшее число: 50
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2 60
Введите В:
Наибольшее число: 50
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.8: Компиляция и запуск программы lab7-2.asm

## 2.2 Изучение структуры файлы листинга

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` (рис. [2.9])

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab07 12756/13853 92%
/home/dvdidenko/work/arch-pc/lab07/lab7-2.lst
171 000000E5 CD80 <1> int 80h
172 000000E7 C3 <1> ret
2 section .data
3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите В: ',0h
3 00000009 B8D182D0B520423A20-
3 00000012 00
4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000
5 00000035 32300000 A dd '20'
6 00000039 35300000 C dd '50'
7 section .bss
8 00000000 <res 0000000A> max resb 10
9 0000000A <res 0000000A> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите В: '
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'В'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'В' из символа в чи
сло
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax
24 ; ----- Записываем 'А' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A]
26 00000116 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'А' и 'С' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C jg check_B
30 00000124 8B0D[39000000] mov ecx,[C]
31 0000012A 890D[00000000] mov [max],ecx
1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format 10Quit
```

Рис. 2.9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга.

строка 192

- 17 - номер строки в подпрограмме
- 000000F2 - адрес

- B9[0A000000] - машинный код
- mov esx,B - код программы - копирует B в esx

строка 193

- 18 - номер строки в подпрограмме
- 000000F7 - адрес
- BA0A000000 - машинный код
- mov edx,10 - код программы - копирует 10 в edx

строка 194

- 19 - номер строки в подпрограмме
- 000000FC - адрес
- E842FFFFFF - машинный код
- call sread - код программы - вызов подпрограммы чтения

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. [2.10]) (рис. [2.11])

```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:36: error: invalid combination of opcode and operands
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.10: Ошибка трансляции lab7-2

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab07
/home/dvdidenko/work/arch-pc/lab07/lab7-2.lst 13941/13941 100%
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в чи
сло
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A]
26 00000116 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C jg check_B
30 00000124 8B0D[39000000] mov ecx,[C]
31 0000012A 890D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из симво
ла в число
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi
36 mov [max],
36 ***** error: invalid combination of opcode and opera
nds
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как ч
исла)
38 0000013A 8B0D[00000000] mov ecx,[max]
39 00000140 3B0D[0A000000] cmp ecx,[B]
40 00000146 7F0C jg fin
41 00000148 8B0D[0A000000] mov ecx,[B]
42 0000014E 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000154 B8[13000000] mov eax,msg2
46 00000159 E8B1FFFFFF call sprint
47 0000015E A1[00000000] mov eax,[max]
48 00000163 E81EFFFFFF call iprintLF
49 00000168 E86EFFFFFF call quit
1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format 10Quit
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

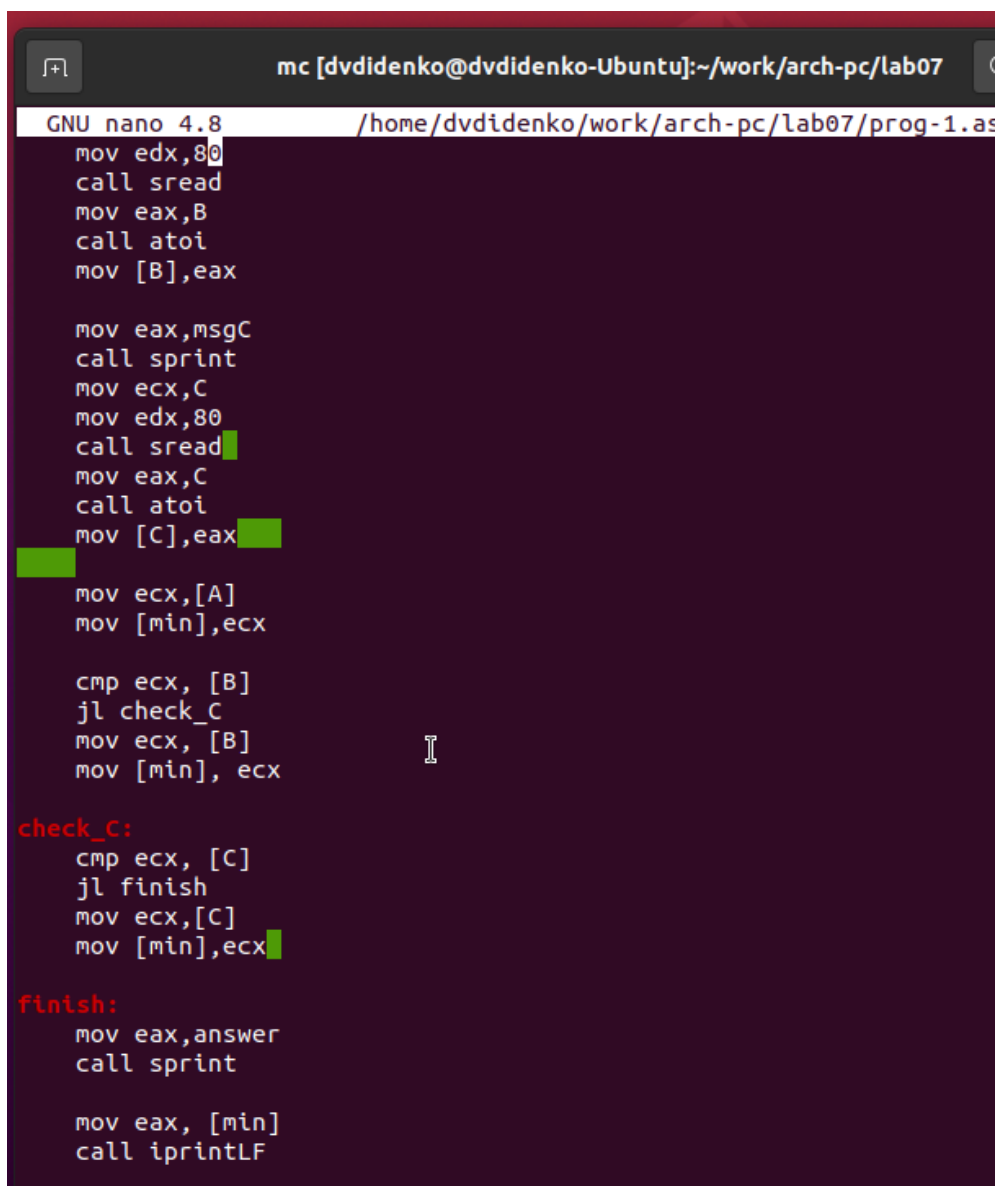
Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.



## 2.3 Задание для самостоятельной работы

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 (рис. [2.12]) (рис. [2.13])

Мой вариант 13 - числа: 84,32,77



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab07
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab07/prog-1.asm
mov edx,80
call sread
mov eax,B
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
mov ecx,[C]
mov [min],ecx

finish:
mov eax,answer
call sprint

mov eax, [min]
call iprintLF
```

Рис. 2.12: Код программы prog-1.asm

```

dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf prog-1.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 prog-1.o -o prog-1
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ./prog-1
Input A: 84
Input B: 32
Input C: 77
Smallest: 32
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$

```

Рис. 2.13: Компиляция и запуск программы prog-1.asm

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. (рис. [2.14]) (рис. [2.15])

Мой вариант 13

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

Если подставить  $x = 3, a = 9$ , тогда  $f(x) = 2$

Если подставить  $x = 6, a = 4$ , тогда  $f(x) = 24$

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-  
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab07  
    mov eax,msgA  
    call sprint  
    mov ecx,A  
    mov edx,80  
    call sread  
    mov eax,A  
    call atoi  
    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 edx,7  
    mov ebx,[A]  
    cmp ebx, edx  
    jge first  
    jmp second  
  
first:  
    mov eax,[A]  
    sub eax,7  
    call iprintLF  
    call quit  
second:  
    mov eax,[A]  
    mov ebx,[X]  
    mul ebx  
    call iprintLF  
    call quit
```

Рис. 2.14: Код программы prog-2.asm

```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf prog-2.asm  
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 prog-2.o -o prog-2  
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ./prog-2  
Input A: 9  
Input X: 3  
2  
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$ ./prog-2  
Input A: 4  
Input X: 6  
24  
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.15: Компиляция и запуск программы prog-2.asm

## 3 Выводы

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