

# Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Козин Иван Евгеньевич

## Содержание

### 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

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

### 3 Теоретическое введение

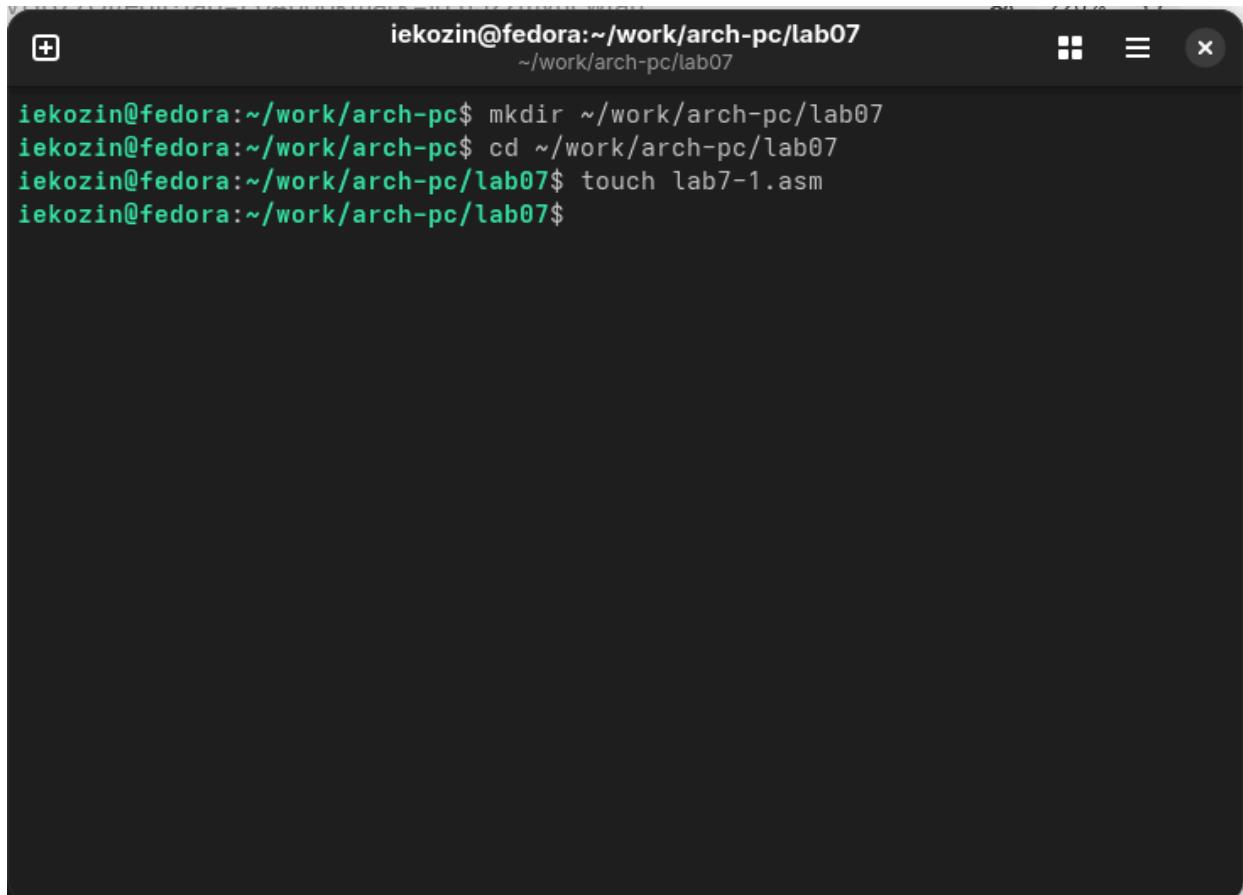
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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.

### 4 Выполнение лабораторной работы

#### 4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7 (рис. 1).



A screenshot of a terminal window titled "iekozin@fedora:~/work/arch-pc/lab07". The window shows the command history:

```
iekozin@fedora:~/work/arch-pc$ mkdir ~/work/arch-pc/lab07
iekozin@fedora:~/work/arch-pc$ cd ~/work/arch-pc/lab07
iekozin@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
iekozin@fedora:~/work/arch-pc/lab07$
```

Рис. 1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. 2).

```
mc [iekozin@fedora]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.p...      +  ⌂  ⌄  ⌁
Left      File      Command      Options      Right
«= ~/work/arch-pc/lab07 =====•[^\"]»  «= ~/work/arch-pc/lab06 =====•[^\"]»
↓n      Name      Size      Modify time      ↓n      Name      Size      Modify time
/..      UP--DIR      Nov 22 11:40      /..      UP--DIR      Nov 22 11:40
in_out.asm      3942      Nov 13 12:20      in_out.asm      3942      Nov 13 12:20
lab7-1.asm      0      Nov 22 11:41      *lab6-1      1192      Nov 16 12:34
                                         lab6-1.asm      168      Nov 16 12:34
                                         lab6-1.o      1200      Nov 16 12:34
                                         *lab6-2      1088      Nov 16 12:49
                                         lab6-2.asm      111      Nov 16 12:49
                                         lab6-2.o      1040      Nov 16 12:49
                                         *lab6-3      4976      Nov 16 13:08
                                         lab6-3.asm      345      Nov 16 13:07
                                         lab6-3.o      1328      Nov 16 13:07
                                         *lab6-4      5060      Nov 16 13:29
                                         lab6-4.asm      384      Nov 16 13:27
                                         lab6-4.o      1424      Nov 16 13:29
                                         *variant      5068      Nov 16 13:22
                                         lab7-1.asm      7537M / 13G (54%)      in_out.asm      7537M / 13G (54%)
                                         Hint: Want to see your *~ backup files? Set it in the Configuration dialog.
                                         [^]
iekozin@fedora:~/work/arch-pc/lab07$ [^]
 1Help   2Menu   3View   4Edit   5Copy   6RenMov  7Mkdir  8Delete  9PullDn 10Quit
```

The screenshot shows a terminal window titled 'mc [iekozin@fedora]'. The command entered is 'mc' followed by file paths. The file 'lab7-1.asm' is open, displaying assembly code. The code includes includes, sections (.data and .text), labels (\_start, \_label1, \_label2, \_label3), and jumps (jmp). It also contains three messages (msg1, msg2, msg3) and prints them using the sprintLF instruction. The assembly code uses the 'eax' register. At the bottom of the terminal, there is a menu bar with numbered options from 1 to 10.

```
lab7-1.asm      [----]  9 L:[ 1+14 15/ 21] *(267 / 325b) 0110 0x06E  [*][X]
%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

 1Help  2Save  3Mark  4Replac  5Copy  6Move  7Search  8Delete  9PullDn 10Quit
```

Рис. 2: Сохранение программы

При запуске программы я убедился в том, что неусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

The screenshot shows a terminal window with the following session:

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

Рис. 3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4).

The screenshot shows a terminal window titled 'mc [iekozin@fedora]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.p...'. The working directory is indicated as '~/work/arch-pc/lab07'. The window displays assembly code for a program named 'lab7-1.asm'. The code includes sections for data and text, global entry point '\_start', and three labels (\_label1, \_label2, \_label3) each printing a message from memory to the console. The assembly instructions use labels like msg1, msg2, msg3 and registers like eax. The bottom of the window shows a menu bar with numbered options: 1Help, 2Save, 3Mark, 4Replace, 5Copy, 6Move, 7Search, 8Delete, 9PullDn, and 10Quit.

```
lab7-1.asm      [-M--]  0 L:[ 1+14 15/ 27] *(237 / 350b) 0106 0x06A [*][X]
%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

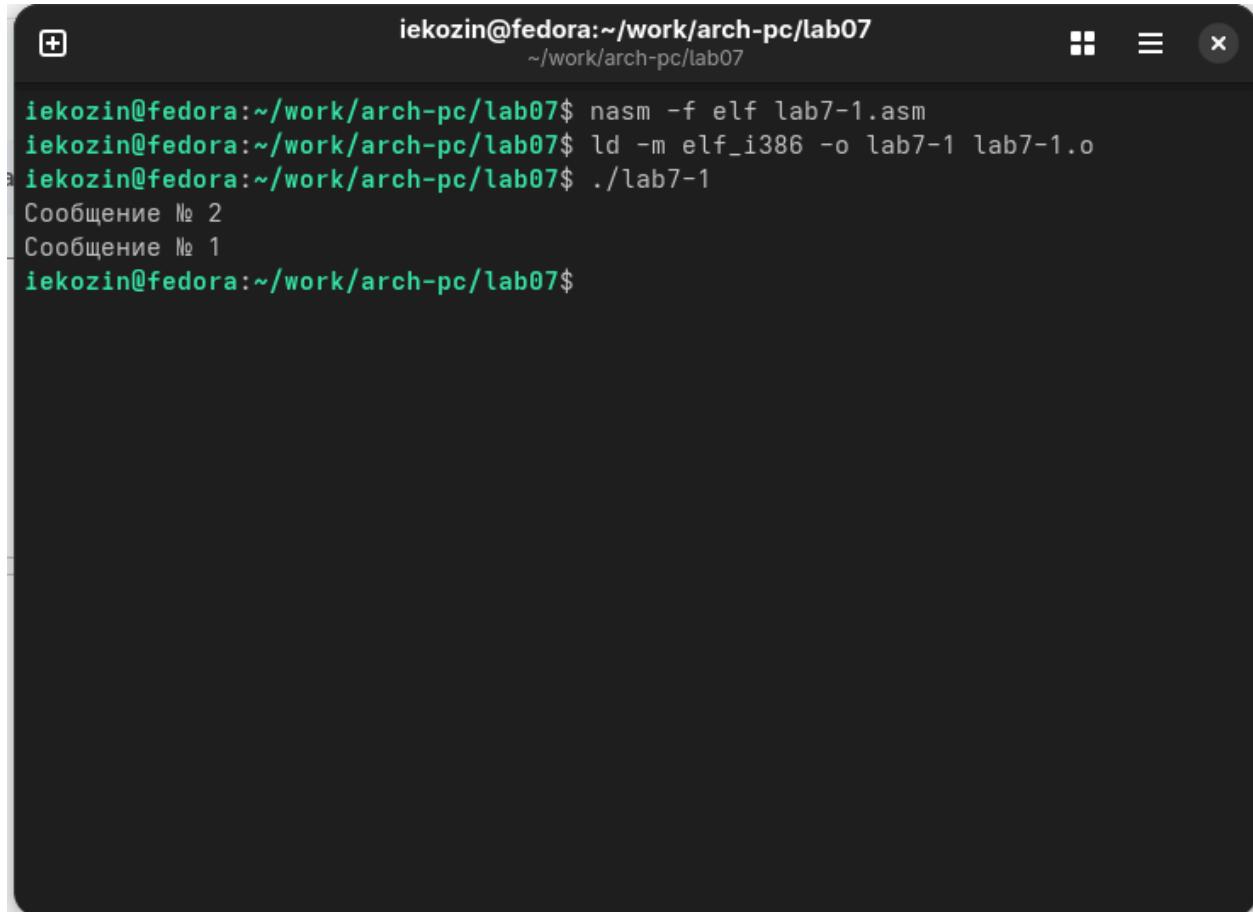
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 5).



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it displays the user's name, the host, the current directory (~/work/arch-pc/lab07), and the path to the working directory (~/work/arch-pc/lab07). The terminal window has standard window controls (minimize, maximize, close) in the top right corner.

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

*Рис. 5: Запуск измененой программы*

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6).

The screenshot shows a terminal window titled 'mc [iekozin@fedora]:' running on a Fedora system. The command entered is '/usr/bin/mc -P /tmp/mc.p...'. The current file is 'lab7-1.asm' located at '~/work/arch-pc/lab07'. The assembly code is as follows:

```
lab7-1.asm      [----]  9 L:[ 1+ 3  4/ 30] *(46   / 367b) 0039 0x027 [*][X]
%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 sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).

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

*Рис. 7: Проверка изменений*

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 8).

The screenshot shows a terminal window titled 'mc [iekozin@fedora]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.p...'. The working directory is ~/work/arch-pc/lab07. The file being edited is 'lab7-2.asm'. The assembly code is as follows:

```
lab7-2.asm      [-M--]  0 L:[ 1+ 0  1/ 49] *(0    /1743b) 0037 0x025  [*][X]
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число

```

At the bottom of the window, there is a menu bar with the following options: 1Help, 2Save, 3Mark, 4Replace, 5Copy, 6Move, 7Search, 8Delete, 9PullDn, 10Quit.

Рис. 8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 9).

```

iekozin@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
iekozin@fedora:~/work/arch-pc/lab07$ mc

iekozin@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
iekozin@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
iekozin@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
iekozin@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 56
Наибольшее число: 56
iekozin@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
iekozin@fedora:~/work/arch-pc/lab07$
```

Рис. 9: Проверка программы из листинга

## 4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. 10).

```

iekozin@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
iekozin@fedora:~/work/arch-pc/lab07$ mc

iekozin@fedora:~/work/arch-pc/lab07$
```

```

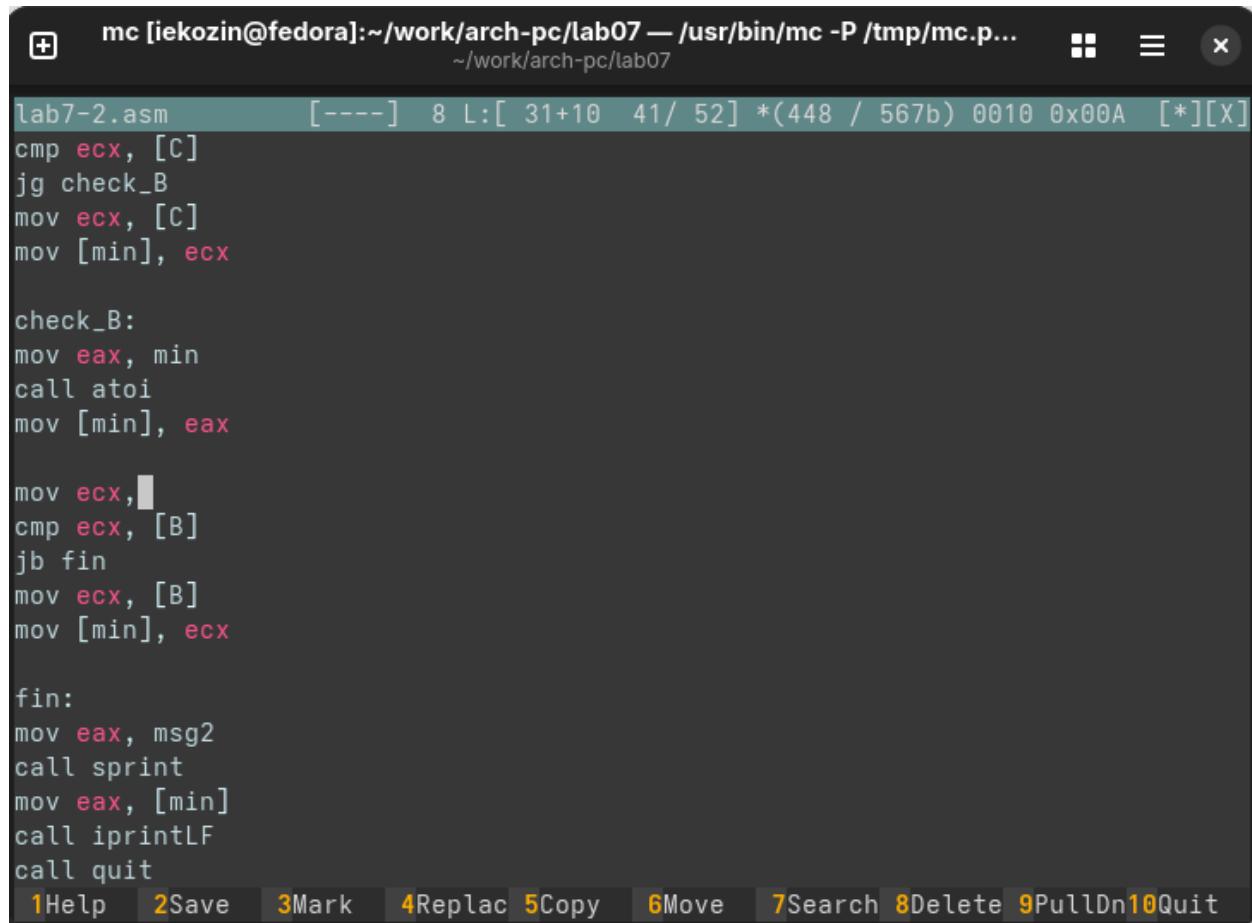
mc [iekozin@fedora:~/work/arch-pc/lab07] — /usr/bin/mc -P /tmp/mc.pwd.5n3FGJ
--workarch-pc/lab07

1          ;----- slen -----
2          ;Функция вычисления длины сообщения
3          ; slen:
4 00000000 53    ; push  ebx
5 00000001 89C3  ; mov   ebx, eax
6          ;<|
7          ;<> nextchar:
8 00000003 B03B00  ; cmp   byte [eax], 0
9 00000006 7403  ; jz    finished
10 00000008 48   ; inc   eax
11 00000009 EBF8  ; jmp   nextchar
12          ;<|
13          ;<> finished:
14 00000008 E908  ; sub   eax, ebx
15 0000000D 5B   ; pop   ebx
16 0000000E C3   ; ret
17          ;<|
18          ;<|
19          ;----- sprint -----
20          ;<| : Функция печати сообщения
21          ;<| : входные данные: mov eax,<message>
22          ;<| sprint:
23 0000000F 52   ; push  edx
24 00000010 51   ; push  ecx
25 00000011 53   ; push  ebx
26 00000012 58   ; push  eax
27 00000013 E8E8FFFF  ; call  slen
28          ;<|
29 00000018 89C2  ; mov   edx, eax
30 0000001A 58   ; pop   eax
31          ;<|
32 0000001B 89C1  ; mov   ecx, eax
33 0000001D B8B1000000  ; mov   ebx, 1
34 00000022 B804000000  ; mov   eax, 4
35 00000027 C0E0  ; int   80h
36          ;<|
37 00000029 5B   ; pop   ebx
38 0000002A 59   ; pop   ecx
39 0000002B 5A   ; pop   edx
40 0000002C C3   ; ret
```

Рис. 10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один operand из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).



The screenshot shows the mc (Mnemonics and Comments) tool window. The title bar says "mc [iekozin@fedora]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.p...". The main area displays assembly code for "lab7-2.asm". The code includes labels like "check\_B:", instructions like "cmp ecx, [C]", and comments like "\* (448 / 567b) 0010 0x00A [\*][X]". The bottom of the window has a menu bar with options 1 through 10: Help, Save, Mark, Replace, Copy, Move, Search, Delete, PullDn, and Quit.

```
lab7-2.asm      [----]  8 L:[ 31+10  41/ 52] *(448 / 567b) 0010 0x00A [*][X]
cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, []
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit
1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 11: Удаление операнда из программы

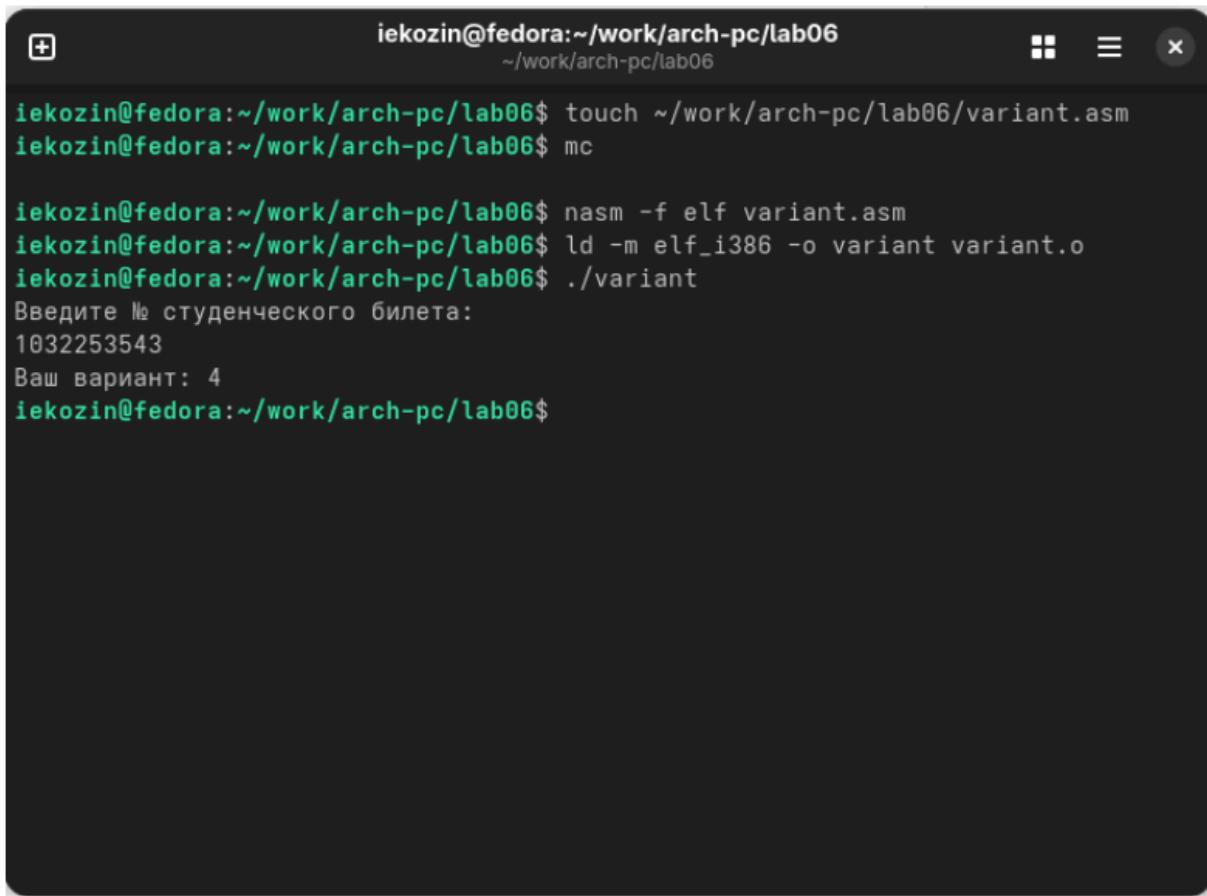
В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 12).

The screenshot shows the mc (Mnemonics and C) tool running in a terminal window. The command used is `mc [iekозин@fedora]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.p...`. The file being viewed is `/home/iekозин/work/arch-pc/lab07/lab7-2.lst`, which contains assembly code and some C-like pseudocode. The assembly code includes instructions like `mov ecx, [C]`, `mov [min], ecx`, and `call atoi`. There is also a section labeled `check_B:` containing `mov eax, min` and `call atoi`. The assembly code ends with a series of `*****` followed by the message `error: invalid combination of opcode an`. Below the assembly code, there is a section labeled `d operands` containing more assembly code. The bottom of the window shows a menu bar with options: 1Help, 2UnWrap, 3Quit, 4Hex, 5Goto, 6, 7Search, 8Raw, 9Format, and 10Quit.

Рис. 12: Просмотр ошибки в файле листинга

### 4.3 Задания для самостоятельной работы

Возвращаю operand к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).



Terminal window showing the assembly code compilation process:

```
iekozin@fedora:~/work/arch-pc/lab06
~/work/arch-pc/lab06

iekozin@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
iekozin@fedora:~/work/arch-pc/lab06$ mc

iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
iekozin@fedora:~/work/arch-pc/lab06$ ./variant
Ведите № студенческого билета:
1032253543
Ваш вариант: 4
iekozin@fedora:~/work/arch-pc/lab06$
```

Скорее всего мой вариант и в 7 лабе это [четвертый](#)



MC (Mars Cross Assembler) interface showing the assembly code:

```
SECTION .data
msgA db "Введите A: ", 0
msgB db "Введите B: ", 0
msgC db "Введите C: ", 0
msgMin db "наименьшее число: ", 0

SECTION .bss
A resb 10
B resb 10
C resb 10
min resd 1

SECTION .text
GLOBAL _start
_start:
; ===== Ввод A =====
mov eax, msgA
call sprint

mov ecx, A
mov edx, 10
call sread

mov eax, A
call atoi
mov [A], eax

; ===== Ввод B =====
mov eax, msgB
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

; ===== Ввод C =====
mov eax, msgC
call sprint

mov ecx, C
mov edx, 10
call sread

mov eax, C
call atoi
mov [C], eax
```

Рис. 13: Первая программа самостоятельной работы

Код первой программы:

```
%include "in_out.asm"
```

```
SECTION .data
```

```
msgA db "Введите A: ", 0
```

```
msgB db "Введите B: ", 0
```

```
msgC db "Введите C: ", 0
```

```
msgMin db "Наименьшее число: ", 0
```

```
SECTION .bss
```

```
A resb 10
```

```
B resb 10
```

```
C resb 10
```

```
min resd 1
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
; ===== Ввод A =====
```

```
mov eax, msgA
```

```
call sprint
```

```
mov ecx, A
```

```
mov edx, 10
```

```
call sread
```

```
mov eax, A
```

```
call atoi
mov [A], eax

; ===== Ввод В =====

mov eax, msgB
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

; ===== Ввод С =====

mov eax, msgC
call sprint

mov ecx, C
mov edx, 10
call sread

mov eax, C
call atoi
mov [C], eax

; ===== Поиск минимума =====
```

```
mov eax, [A]      ; min = A
mov [min], eax

mov eax, [B]
cmp eax, [min]
jge check_C
mov [min], eax

check_C:
mov eax, [C]
cmp eax, [min]
jge finish
mov [min], eax

finish:
mov eax, msgMin
call sprint

mov eax, [min]
call iprintLF

call quit
```

Проверяю корректность написания первой программы (рис. 14).

The screenshot shows a terminal window with the following session:

```
iekozin@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
iekozin@fedora:~/work/arch-pc/lab07$ mc

iekozin@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
iekozin@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
iekozin@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите A: 8
Введите B: 88
Введите C: 68
Наименьшее число: 8
iekozin@fedora:~/work/arch-pc/lab07$
```

Рис. 14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурых переменных a и x (рис. 15).

The screenshot shows the Mars Cross Assembler (mc) interface. The assembly code is as follows:

```
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0
SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov edi, eax

    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atoi
    mov esi, eax

    cmp edi, esi
    jle add_values
    mov eax, esi
    jmp print_result

add_values:
    mov eax, edi
    add eax, esi

print_result:
    mov edi, eax
    mov eax, res
    call sprint
    mov eax, edi
```

Рис. 15: Вторая программа самостоятельной работы

Код второй программы:

```
%include 'in_out.asm'
```

```
SECTION .data_
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0
```

```
SECTION .bss
```

```
x: RESB 80
a: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
; Ввод x
```

```
    mov eax, msg_x
```

```
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax      ; edi = x

; Ввод a
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax      ; esi = a

; Проверка a != 0
cmp esi, 0
je a_is_zero

; a != 0 => f(x) = 2*x + a
mov eax, edi      ; eax = x
shl eax, 1        ; eax = 2*x
add eax, esi      ; eax = 2*x + a
jmp print_result
```

```
a_is_zero:  
    ; a == 0 => f(x) = 2*x + 1  
    mov eax, edi          ; eax = x  
    shl eax, 1           ; eax = 2*x  
    add eax, 1           ; eax = 2*x + 1  
  
print_result:  
    mov edi, eax          ; сохраняем результат  
    mov eax, res  
    call sprint  
    mov eax, edi  
    call iprintfLF  
  
    call quit
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 16).

```
iekozin@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
iekozin@fedora:~/work/arch-pc/lab07$ mc

iekozin@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
iekozin@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
iekozin@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 3
Введите значение переменной a: 0
Результат: 0
iekozin@fedora:~/work/arch-pc/lab07$
```

```
iekozin@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 3
Введите значение переменной a: 2
Результат: 2
iekozin@fedora:~/work/arch-pc/lab07$
```

Рис. 16: Проверка работы второй программы

## 5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

## Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.