

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Кузнецова Елизавета Андреевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выполнение заданий для самостоятельной работы</b>	<b>16</b>
<b>6</b>	<b>Выводы</b>	<b>22</b>

# Список иллюстраций

4.1	Создание директории . . . . .	8
4.2	Создание файла . . . . .	8
4.3	Копирование файла . . . . .	8
4.4	Редактирование файла . . . . .	9
4.5	Запуск исполняемого файла . . . . .	9
4.6	Редактирование файла . . . . .	9
4.7	Создание и запуск нового исполняемого файла . . . . .	10
4.8	Редактирование файла . . . . .	10
4.9	Создание и запуск нового исполняемого файла . . . . .	10
4.10	Создание файла . . . . .	10
4.11	Редактирование файла . . . . .	11
4.12	Создание и запуск нового исполняемого файла . . . . .	11
4.13	Создание файла и его открытие . . . . .	12
4.14	Ознакомление с файлом . . . . .	13
4.15	Редактирование файла . . . . .	14
4.16	Трансляция с получением файла листинга . . . . .	14
4.17	Открытие файла после выполнения трансляции . . . . .	15
5.1	Создание файла . . . . .	16
5.2	Редактирование файла . . . . .	17
5.3	Создание и запуск нового исполняемого файла . . . . .	17
5.4	Создание файла . . . . .	17
5.5	Редактирование файла . . . . .	18
5.6	Создание и запуск нового исполняемого файла . . . . .	18

## Список таблиц

# 1 Цель работы

Изучить команды условного и безусловного переходов и научиться писать программы с использованием этих переходов.

## 2 Задание

1. Ознакомиться с листингами, проделать с ними изменения.
2. Выполнить задания для самостоятельной работы.
3. Загрузить файлы на Github.

### 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. Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление. Для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

## 4 Выполнение лабораторной работы

С помощью утилиты `mkdir` создала директорию, в которой буду создавать файлы с программами для лабораторной работы. Перешла в созданный каталог с помощью утилиты `cd` (рис. [4.1]).

```
eakuznecova@dk8n68 ~ $ mkdir ~/work/arch-pc/lab07
eakuznecova@dk8n68 ~ $ cd ~/work/arch-pc/lab07
```

Рис. 4.1: Создание директории

С помощью утилиты `touch` создала файл `lab7-1.asm` (рис. [4.2]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab07 $ touch lab7-1.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab07 $ ls
lab7-1.asm
```

Рис. 4.2: Создание файла

Скопировала в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, так как он будет использоваться в других программах (рис. [4.3]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab07 $ cp ~/Загрузки/in_out.asm in_out.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab07 $ ls
in_out.asm  lab7-1.asm
```

Рис. 4.3: Копирование файла

Открыла созданный файл `lab7-1.asm`, вставила в него программу (программа с использованием инструкции `jmp`) (рис. [4.4]).



```
GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла

SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 1'

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 3'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Редактирование файла

Создала исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [4.5]).

```
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
```

Рис. 4.5: Запуск исполняемого файла

Добавила в текст программы после вывода сообщения № 2 инструкцию `jmp` с меткой `_label1` и и после вывода сообщения № 1 добавила инструкцию `jmp` с меткой `_end` (рис. [4.6]).

```
GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла

SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 3'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [4.7]).

```
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
```

Рис. 4.7: Создание и запуск нового исполняемого файла

Добавила в текст программы после вывода сообщения № 3 инструкцию `jmp` с меткой `_label2`, также использовала инструкцию `jmp _label3` (рис. [4.8]).

```
GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab07/lab7-1.asm
%include "in_out.asm" ; подключение внешнего файла

SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label3

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.8: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [4.9]).

```
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
```

Рис. 4.9: Создание и запуск нового исполняемого файла

С помощью утилиты `touch` создала файл `lab7-2.asm` (рис. [4.10]).

```
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ touch lab7-2.asm
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
```

Рис. 4.10: Создание файла

Открыла созданный файл lab7-2.asm, вставила в него программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C (рис. [4.11]).

```

GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab07/lab7-2.asm
#include "in_out.asm"
section .data
msg1 db "Введите B: ",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 B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call read
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoll ; Вывод подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B'
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoll ; Вывод подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как число)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin'
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint
mov eax,[max]
call printf ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Рис. 4.11: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результаты данной программы выводились правильно (рис. [4.12]).

```

eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 7
Наибольшее число: 50
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 25
Наибольшее число: 50
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 153
Наибольшее число: 153
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 58
Наибольшее число: 58

```

Рис. 4.12: Создание и запуск нового исполняемого файла

Создала файл листинга для программы из файла lab7-2.asm и открыла его с помощью с помощью текстового редактора mcedit (рис. [4.13]).

```
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst lab7-2.o
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
```

Рис. 4.13: Создание файла и его открытие

После открытия файла ознакомилась с ним и объяснила содержимое трёх строк файла. В строке номер 10 содержится номер строки [10], адрес [00000008], машинный код [40], содержимое строки кода [inc eax]. В строке номер 23 содержится номер строки [23], адрес [0000000F], машинный код [52], содержимое строки кода [push edx]. В строке номер 35 содержится номер строки [35], адрес [00000027], машинный код [35], содержимое строки кода [int 80h] (рис. [4.14]).

```

lab7-2.lst      [-----]  0 L:[ 1+ 0 1/225] *(0 /14626b) 0032 0x020
1               %include 'in_out.asm'
1               <1> ;----- slen -----
2               <1> ; Функция вычисления длины сообщения
3               <1> slen:
4 00000000 53      <1>      push    ebx
5 00000001 89C3    <1>      mov     ebx, eax
6               <1>
7               <1> nextchar:
8 00000003 803800  <1>      cmp     byte [eax], 0
9 00000006 7403    <1>      jz      finished
10 00000008 40     <1>      inc     eax
11 00000009 EBF8   <1>      jmp     nextchar
12               <1>
13               <1> finished:
14 0000000B 29D8   <1>      sub     eax, ebx
15 0000000D 5B     <1>      pop     ebx
16 0000000E C3     <1>      ret
17               <1>
18               <1>
19               <1> ;----- sprint -----
20               <1> ; Функция печати сообщения
21               <1> ; входные данные: mov eax,<message>
22               <1> sprint:
23 0000000F 52     <1>      push    edx
24 00000010 51     <1>      push    ecx
25 00000011 53     <1>      push    ebx
26 00000012 50     <1>      push    eax
27 00000013 E8E8FFFF <1>      call    slen
28               <1>
29 00000018 89C2   <1>      mov     edx, eax
30 0000001A 58     <1>      pop     eax
31               <1>
32 0000001B 89C1   <1>      mov     ecx, eax
33 0000001D BB01000000 <1>      mov     ebx, 1
34 00000022 B804000000 <1>      mov     eax, 4
35 00000027 CD80   <1>      int     80h
36               <1>
37 00000029 5B     <1>      pop     ebx
38 0000002A 59     <1>      pop     ecx
39 0000002B 5A     <1>      pop     edx
40 0000002C C3     <1>      ret
41               <1>
42               <1>

```

Рис. 4.14: Ознакомление с файлом

Открыла файл с программой lab7-2.asm и в инструкции с двумя операндами удалила один операнд (было: mov eax,msg1; стало: mov eax) (рис. [4.15]).

```
GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab07/lab7-2.asm
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax, 1
call sprintf
; ----- Ввод 'B'
mov ecx, B
mov edx, 10
call read
; ----- Преобразование 'B' из символа в число
mov ecx, B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B], eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx, [A] ; 'ecx = A'
mov [max], ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx, [C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx, [C] ; иначе 'ecx = C'
mov [max], ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax, max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max], eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx, [max]
cmp ecx, [B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx, [B] ; иначе 'ecx = B'
mov [max], ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprintf ; Вывод сообщения 'Наибольшее число: '
mov eax, [max]
call iprintlf ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 4.15: Редактирование файла

Выполнила трансляцию с получением файла листинга. Вывелась ошибка (рис. [4.16]).

```
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
```

Рис. 4.16: Трансляция с получением файла листинга

Открыла файл листинга для программы из файла lab7-2.asm. Если в коде появляется ошибка, то ее описание появится в файле листинга (рис. [4.17]).

```

170 000000E7 C3          <1>      ret
2          section      .data
3 00000000 0092D0B2D0B5D0B4D0-      msg1 db 'Введите B: ',0h
3 00000000 B8D182D0B520423A20-
3 00000012 00
4 00000013 D09D08B0D0B8D0B1D0-      msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D0B1-
4 0000002E D0BBD0BE3A2000
5 00000035 32300000      A dd '20'
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
14         ***** error: invalid combination of opcode and operands
15 000000E8 E822FFFFFF      call sprint
16         ; ----- Ввод 'B'
17 000000ED B0[0A000000]      mov ecx,B
18 000000F2 BA0A000000      mov edx,10
19 000000F7 E847FFFFFF      call sread
20         ; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000]      mov eax,B
22 00000101 E896FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число

```

Рис. 4.17: Открытие файла после выполнения трансляции

## 5 Выполнение заданий для самостоятельной работы

С помощью утилиты touch создала файл lab7-3.asm (рис. [5.1]).

```
eakuznecova8dk2n24 ~/work/arch-pc/lab07 $ touch lab7-3.asm
eakuznecova8dk2n24 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst lab7-3.asm
```

Рис. 5.1: Создание файла

Открыла созданный файл lab7-3.asm, написала программу нахождение наименьшей из 3 целочисленных переменных. Выбрала значения переменных 20 варианта, полученным при выполнении лабораторной работы № 6 (рис. [5.2]).



```

GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'
SECTION .data
msg2 db "Наименьшее число: ",0h
A dd '95'
B dd '2'
C dd '61'
SECTION .bss
min resb 10
SECTION .text
global _start
_start:
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A < C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'min'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C) < B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprintf ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Рис. 5.2: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы вывелся правильно (рис. [5.3]).

```

eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 2

```

Рис. 5.3: Создание и запуск нового исполняемого файла

С помощью утилиты touch создала файл lab7-4.asm (рис. [5.4]).

```

eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ touch lab7-4.asm
eakuznecova@dk2n24 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst lab7-3 lab7-3.asm lab7-3.o lab7-4.asm

```

Рис. 5.4: Создание файла

Открыла созданный файл lab7-4.asm, написала программу, которая для введенных с клавиатуры значений  $x$  и  $y$  вычисляет значение заданной функции  $\max(x, y)$  и выводит результат вычислений. Выбрала значения переменных 20 варианта, полученным при выполнении лабораторной работы № 6 (рис. [5.5]).

```

GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab07/lab7-4.asm
#include 'in_out.asm'
SECTION .data
msg1 db "Введите x: ",0h
msg2 db "Введите a: ",0h
msg3 db "f(x)= "
SECTION .bss
x resb 10
a resb 10
SECTION .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
call atoi
mov [x],eax
mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
mov eax,a
call atoi
mov [a],eax
mov ecx,[a]
cmp ecx,[x]
jg check_a
mov eax,[a]
neg eax
mov ecx,[x]
add ecx,eax
jmp _end
check_a:
mov ecx,5
_end:
mov eax,msg3
call sprint
mov eax,ecx
call iprintlnf
call quit

```

Рис. 5.5: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы вывелся правильно (рис. [5.6]).

```

eakuznecova@dk8n68 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
eakuznecova@dk8n68 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 1
Введите a: 2
f(x)= 5
eakuznecova@dk8n68 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 2
Введите a: 1
f(x)= 1

```

Рис. 5.6: Создание и запуск нового исполняемого файла

**Программу, которую я использовала для выполнения 1 пункта самостоятельной работы:**

```

#include 'in_out.asm'
SECTION .data
msg2 db "Наименьшее число: ",0h
A dd '95'
B dd '2'
C dd '61'

```

```

SECTION .bss
    min resb 10
SECTION .text
    global _start
_start:

; ----- Преобразование 'B' из символа в число
    mov eax,B
    call atoi      ; Вызов подпрограммы перевода символа в число
    mov [B],eax    ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'min'
    mov ecx,[A]    ; 'ecx = A'
    mov [min],ecx  ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
    cmp ecx,[C]    ; Сравниваем 'A' и 'C'
    jl  check_B    ; если 'A<C', то переход на метку 'check_B',
    mov ecx,[C]    ; иначе 'ecx = C'
    mov [min],ecx  ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
    mov eax,min
    call atoi      ; Вызов подпрограммы перевода символа в число
    mov [min],eax  ; запись преобразованного числа в `max`
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
    mov ecx,[min]
    cmp ecx,[B]    ; Сравниваем 'min(A,C)' и 'B'
    jl  fin        ; если 'min(A,C)>B', то переход на 'fin',
    mov ecx,[B]    ; иначе 'ecx = B'
    mov [min],ecx

```

```

; ----- Вывод результата
fin:
    mov eax, msg2
    call sprint ; Вывод сообщения 'Наименьшее число: '
    mov eax,[min]
    call iprintLF ; Вывод 'min(A,B,C)'
    call quit ; Выход

```

**Программа, которую я использовала для выполнения 2 пункта самостоятельной работы:**

```

#include 'in_out.asm'
SECTION .data
    msg1 db "Введите x: ",0h
    msg2 db "Введите a: ",0h
    msg3 db "f(x)= "
SECTION .bss
    x resb 10
    a resb 10
SECTION .text
    global _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,x
    mov edx,10
    call sread
    mov eax,x
    call atoi
    mov [x],eax

```

```

    mov eax,msg2
    call sprint
    mov ecx,a
    mov edx,10
    call sread
    mov eax,a
    call atoi
    mov [a],eax
    mov ecx,[a]
    cmp ecx,[x]
    jg check_a
    mov eax,[a]
    neg eax
    mov ecx,[x]
    add ecx,eax
    jmp _end
check_a:
    mov ecx,5
_end:
    mov eax,msg3
    call sprint
    mov eax,ecx
    call iprintLF
    call quit

```

## 6 Выводы

Были изучены команды условного и безусловного перехода, также был получен навык по написанию программы с использованием этих переходов.