

Шаблон отчёта по лабораторной работе № 7

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

Ван Сихэм Франклин О Нил Джон (Миша)

Содержание

1	Цель работы	5
2	Порядок выполнения лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	19
3	Задание для самостоятельной работы	26
4	Заключение	31

Список иллюстраций

2.1	“Создание файл лаб7-1.asm	6
2.2	Листинг 7.1. Программа с использованием инструкции jmp в NASM	8
2.3	Результат работы данной программы	9
2.4	Программа чтобы выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу	10
2.5	Результат программы чтобы выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу	11
2.6	Листинг 7.2. Программа с использованием инструкции jmp . .	13
2.7	Результат Листинг 7.2. Программа с использованием инструкции jmp	14
2.8	Создан файл lab7-2.asm	14
2.9	Листинг 7.3. Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C.	18
2.10	Результат программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. .	19
2.11	файл lab7-2.asm листинга создан	20
2.12	файл листинга lab7-2.lst	23
2.13	3 строки которые выбраны	24
2.14	один операнд который будет удалён	24
2.15	один операнд который удалён	25
2.16	Сбой в листинге	25
3.1	Программа для самой-работы выбран из табл. 7.5	27
3.2	Результат программы для самой-работы выбран из табл. 7.5 .	28

Список таблиц

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Порядок выполнения лабораторной работы

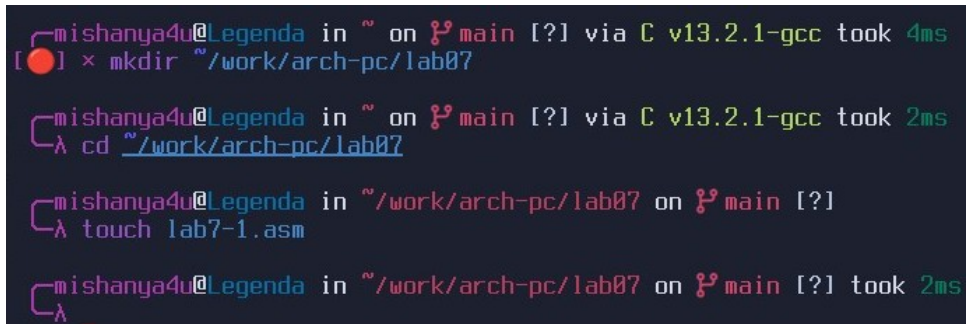
2.1 Реализация переходов в NASM

1. Создайте каталог для программ лабораторной работы No 7, перейдите в него и создайте файл lab7-1.asm:

```
mkdir ~/work/arch-pc/lab07
```

```
cd ~/work/arch-pc/lab07
```

```
touch lab7-1.asm
```



```
mishanya4u@Legenda in ~ on 3 main [?] via C v13.2.1-gcc took 4ms  
[ ] x mkdir ~/work/arch-pc/lab07  
  
mishanya4u@Legenda in ~ on 3 main [?] via C v13.2.1-gcc took 2ms  
[ ] cd ~/work/arch-pc/lab07  
  
mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?]  
[ ] touch lab7-1.asm  
  
mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 2ms  
[ ] _
```

Рис. 2.1: “Создание файл лаб7-1.asm

2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введите в файл lab7-1.asm текст программы из листинга 7.1.

Листинг 7.1. Программа с использованием инструкции `jmp`

```
%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 sprintf ; 'Сообщение No 1'
```

```
_label2:
```

```
mov eax, msg2 ; Вывод на экран строки
```

```
call sprintf ; 'Сообщение No 2'
```

```
_label3:
```

```
mov eax, msg3 ; Вывод на экран строки
```

```
call sprintf ; 'Сообщение No 3'
```

```
_end:
```

```
call quit ; вызов подпрограммы завершения
```

```

lab7-1.asm      [-M--] 54 L:[ 1+26 27/ 27] *(685 / 685b) <EOF> [*][X]
%include<----->'in_out.asm'<-->;подключение внешнего файла

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

SECTION .txt
GLOBAL _start
_start

jmp _label2

_label1:
    mov eax, msg1<----->;Вывод на экран строки
    call sprintf<----->;'Сообщение № 1'
. . .
_label2:
    mov eax, msg2<----->;Вывод на экран строки
    call sprintf<----->;'Сообщение № 2'
. . .
_label3:
    mov eax, msg3<----->;Вывод на экран строки
    call sprintf<----->;'Сообщение № 3'
. . .
_end:
    call quit<--><----->;вызов подпрограммы завершения_

```

Рис. 2.2: Листинг 7.1. Программа с использованием инструкции `jmp` в NASM

Создайте исполняемый файл и запустите его. Результат работы данной программы будет следующим:

```

user@dk4n31:~$ ./lab7-1
сообщение №2
сообщение №3
user@dk4n31:~$

```



```
mishanya4u@Legenda in ~/work/arch-pc/lab07 on 19 main [?] took 16ms
λ mcedit lab7-1.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 19 main [?] took 36s
λ nasm -f elf lab7-1.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 19 main [?] took 5ms
λ ld -m elf_i386 -o lab7-1 lab7-1.o

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 19 main [?] took 6ms
λ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 2.3: Результат работы данной программы

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу. Для этого в текст программы после вывода сообщения No 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения No 1) и после вывода сообщения No 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом 7.2.

```

lab7-1-2.asm      [----] 46 L:[ 1+26 27/ 31] *(585 / 711b) 0010 0[*]1[X]
%include<----->'in_out.asm'<-->;подключение внешнего файла

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<----->;Вывод на экран строки
    call sprintf<----->;'Сообщение № 1'

    jmp _end

_label2:
    mov eax, msg2<----->;Вывод на экран строки
    call sprintf<----->;'Сообщение № 2'

    jmp _label1

_label3:
    mov eax, msg3<----->;Вывод на экран строки
    call sprintf<----->;'Сообщение № 3'

_end:
    call quit<----><----->;Вызов подпрограммы завершения

```

Рис. 2.4: Программа чтобы выводила сначала 'Сообщение No 2', потом 'Сообщение No 1' и завершала работу

```

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 89 main [?] took 9ms
[●] × mcedit lab7-1-2.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 89 main [?] took 32s
λ nasm -f elf lab7-1-2.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 89 main [?] took 6ms
λ ld -m elf_i386 -o lab7-1-2 lab7-1-2.o

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 89 main [?] took 14ms
λ ./lab7-1-2
Сообщение № 2
Сообщение № 1

```

Рис. 2.5: Результат программы чтобы выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу

Листинг 7.2. Программа с использованием инструкции jmp

```
%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 sprintf ; '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
```

```

lab7-1-3.asm      [----] 13 L:[ 1+26 27/ 30] *(642 / 722b) 0010 0[*](X)
%include<----->'in_out.asm'<--><----->;подключение внешнего файла

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<----->;Вывод на экран строки
    call sprintf<----->;'Сообщение № 1'
    jmp _end

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

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

_end:
    call quit<----><----->;Вызов подпрограммы завершения

```

Рис. 2.6: Листинг 7.2. Программа с использованием инструкции jmp

Создайте исполняемый файл и проверьте его работу. Измените текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим:

```

user@dk4n31:~$ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
user@dk4n31:~$

```

```
mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 1ms
λ mcedit lab7-1-3.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 2m4s
λ nasm -f elf lab7-1-3.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 8ms
λ ld -m elf_i386 -o lab7-1-3 lab7-1-3.o

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 10ms
λ ./lab7-1-3
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 2.7: Результат Листинг 7.2. Программа с использованием инструкции `jmp`

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.

Создайте файл `lab7-2.asm` в каталоге `~/work/arch-pc/lab07`. Внимательно изучите текст программы из листинга 7.3 и введите в `lab7-2.asm`.

```
mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 1ms
λ touch lab7-2.asm
```

Рис. 2.8: Создан файл `lab7-2.asm`

Листинг 7.3. Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C.

```

#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 sread

; ----- 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 '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   fprintf    ; Вывод 'max(A,B,C)'
```

```
    call   quit       ; Выход
```

```

lab7-2.asm      [----] 30 L:[ 15+32  47/ 53] *(1736/1934b) 1072 0x430
_start:

;----- Вывод сообщение 'Введите B: '
mov eax, msg1
call sprint

;----- Ввод 'B'
mov ecx, B
mov edx, 10
call sread

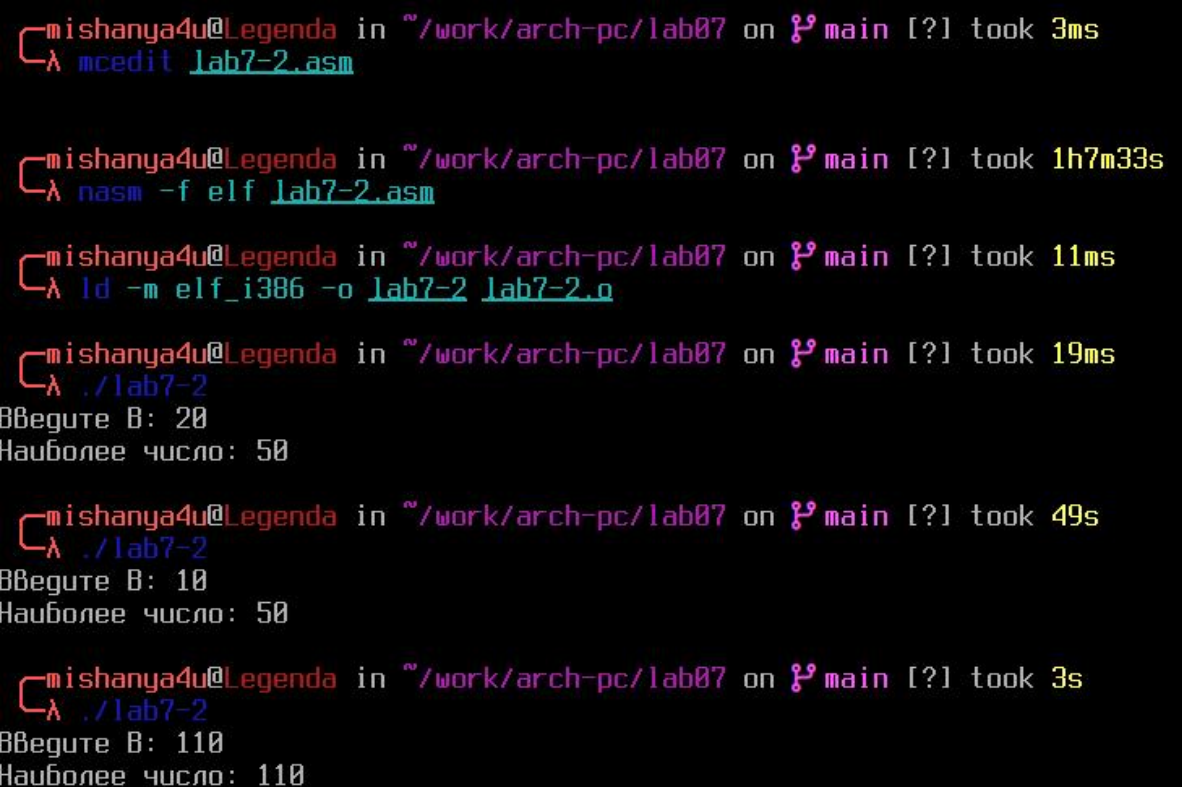
;----- Преобразование 'B' из символа в число
mov eax, B
call atoi<-><----->;Вызов подпрограммы перевода символа в число
mov [B], eax<----->;запись преобразованного числа в 'B'
;----- Записываем 'A' в переменную 'max'
mov ecx, [A]<----->;ecx = A
mov [max], ecx<----->;max = A
;----- Сравниваем 'A' и 'C' (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 sprint<-><----->;Вывод сообщения 'Наиболее число: '
mov eax, [max]
call iprintLF<----->;Вывод 'max(A,B,C)'
call quit<-><----->;Выход

```

Рис. 2.9: Листинг 7.3. Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C.

Создайте исполняемый файл и проверьте его работу для разных значений B. Обратите внимание, в данном примере переменные A и C сравниваются как

символы, 0a переменная B и максимум из A и C как числа (для этого используется функция `atoi` преобразования символа в число). Это сделано для демонстрации того, как сравниваются данные. Данную программу можно упростить и сравнивать все 3 переменные как символы (т.е. не использовать функцию `atoi`). Однако если переменные преобразовать из символов числа, над ними можно корректно проводить арифметические операции.



```
mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐚 main [?] took 3ms
λ mcedit lab7-2.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐚 main [?] took 1h7m33s
λ nasm -f elf lab7-2.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐚 main [?] took 11ms
λ ld -m elf_i386 -o lab7-2 lab7-2.o

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐚 main [?] took 19ms
λ ./lab7-2
Введите B: 20
Наиболее число: 50

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐚 main [?] took 49s
λ ./lab7-2
Введите B: 10
Наиболее число: 50

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐚 main [?] took 3s
λ ./lab7-2
Введите B: 110
Наиболее число: 110
```

Рис. 2.10: Результат программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C.

2.2 Изучение структуры файлы листинга

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 lab7-2.asm

```
nasm -f elf -l lab7-2.lst lab7-2.asm
```

```
mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 2s
λ nasm -f elf -l lab7-2.lst lab7-2.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 12ms
λ ls
-rw-r--r-- 3,9k mishanya4u 11 ноя 23:17 ASM in_out.asm
-rwxr-xr-x 9,2k mishanya4u 26 дек 23:13 lab7-1
-rwxr-xr-x 9,2k mishanya4u 26 дек 00:31 lab7-1-2
-rw-r--r-- 710 mishanya4u 26 дек 00:30 ASM lab7-1-2.asm
-rw-r--r-- 1,5k mishanya4u 26 дек 00:30 lab7-1-2.o
-rwxr-xr-x 9,2k mishanya4u 26 дек 01:15 lab7-1-3
-rw-r--r-- 722 mishanya4u 26 дек 01:15 ASM lab7-1-3.asm
-rw-r--r-- 1,5k mishanya4u 26 дек 01:15 lab7-1-3.o
-rw-r--r-- 688 mishanya4u 25 дек 23:13 ASM lab7-1.asm
-rw-r--r-- 1,4k mishanya4u 25 дек 23:13 lab7-1.o
-rwxr-xr-x 9,2k mishanya4u 26 дек 02:35 lab7-2
-rw-r--r-- 1,9k mishanya4u 26 дек 02:33 ASM lab7-2.asm
-rw-r--r-- 15k mishanya4u 26 дек 02:53 lab7-2.lst
-rw-r--r-- 1,7k mishanya4u 26 дек 02:53 lab7-2.o
```

Рис. 2.11: файл lab7-2.asm листинга создан

Откройте файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit:

```
mcedit lab7-2.lst
```

```
mishanya4u@Legenda in ~/work/arch-pc/lab07 on 3 main [?] took 18ms
λ mcedit lab7-2.lst
```

```

lab7-2.lst 68 L: 12/22/2011 11:11 *3221714896* 0010 0x0000
1  include <stdio.h>
2  ; Функция вычисления длины сообщения
3  ;
4  00000000 53          ; len:
5  00000001 09C3          ;
6  ;
7  ; nextchar:
8  00000003 003000          ; cmp byte [eax], 0
9  00000006 7403          ; jz finished
10 00000008 40          ; inc eax
11 00000009 E0F0          ; jmp nextchar
12 ;
13 ; finished:
14 0000000B 2908          ; 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 ;
23 0000000F 52          ; push edx
24 00000010 51          ; push ecx
25 00000011 53          ; push ebx
26 00000012 50          ; push eax
27 00000013 E000FFFFFF    ; call len
28 ;
29 00000018 09C2          ; mov edx, eax
30 0000001A 5B          ; pop ebx
31 ;
32 0000001B 09C1          ; mov ecx, eax
33 0000001D 00010000          ; mov ebx, 1
34 00000022 00040000          ; mov eax, 4
35 00000027 C000          ; int 00h
36 ;
37 00000029 50          ; pop ebx
38 0000002A 59          ; pop ecx
39 0000002B 5A          ; pop edx
40 0000002C C3          ; ret
41 ;
42 ;
43 ; ;----- sprintf -----
44 ; ; Функция печати сообщения с переводом строки
45 ; ; Входные данные: mov eax, <message>
46 ;
47 0000002D E000FFFFFF    ; call sprintf
48 ;
49 00000032 5B          ; pop ebx
50 ;
51 00000033 00000000          ; mov eax, 0
52 00000035 50          ; push eax
53 00000036 09E0          ; mov eax, esp
54 00000038 E000FFFFFF    ; call sprintf
55 0000003B 50          ; pop eax
56 0000003D C3          ; ret
57 ;
58 ; ;----- sread -----
59 ; ; Функция считывания сообщения
60 ; ; Входные данные: mov eax, <buffer>, mov ebx, <N>
61 ;
62 00000043 53          ; push ebx
63 00000044 5B          ; pop ebx
64 ;
65 00000045 00000000          ; mov ebx, 0
66 00000048 00030000          ; mov eax, 3
67 0000004F C000          ; int 00h
68 ;
69 00000051 5B          ; pop ebx
70 00000052 59          ; pop ecx
71 00000053 C3          ; ret
72 ;
73 ; ;----- iprint -----
74 ; ; Функция вывода на экран чисел в формате ASCII
75 ; ; Входные данные: mov eax, <int>
76 ;
77 00000054 50          ; push eax
78 00000055 51          ; push ecx
79 00000056 52          ; push edx
80 00000057 56          ; push esi
81 00000058 00000000          ; mov ecx, 0
82 ;
83 ; divideloop:
84 0000005D 41          ; inc ecx
85 0000005E 00000000          ; mov ebx, 0
86 00000063 0E000000          ; mov esi, 10
87 00000068 F7FE          ; idiv esi
88 0000006A 03C230          ; add edx, 40
89 0000006D 52          ; push edx
90 0000006E 03F000          ; cmp eax, 0
91 00000071 75EA          ; jnz divideloop
92 ;
93 ; printLoop:
94 00000073 49          ; dec ecx
95 00000074 09E0          ; mov eax, esp
96 00000076 E094FFFFFF    ; call sprintf
97 0000007B 5B          ; pop ebx
98 0000007C 03F000          ; cmp ecx, 0
99 00000077 75F2          ; jnz printLoop

```


[illegible]

```

5 00000024 302000          R dd '20'
6 00000025 323000          C dd '50'
7 00000026 353000          C dd '50'
8
9
10 00000000 <res Rb>      max resb 10
11 00000000 <res Rb>      B resb 10
12
13
14
15
16
17
18 000000E8 0010000000      mov eax, msg1
19 000000ED E010FFFF      call sprint
20
21 000000F2 0910000000      mov ecx, 0
22 000000F7 0A00000000      mov edx, 10
23 000000FC E042FFFF      call sread
24
25 00000101 0010000000      mov eax, 0
26 00000106 E091FFFF      call atoi<---->;Вызов подпрограммы перевода символа в число
27 0000010B 0310000000      mov [B], eax<---->;запись преобразованного числа в 'B'
28
29 00000110 000032000000      mov ecx, [A]<---->;ecx = A
30 00000116 0900000000      mov [max], ecx<---->;max = A
31
32 0000011C 300036000000      cmp ecx, [C]<---->;Сравниваем 'A' и 'C' (как символы)
33 00000122 7F0C          jg check_B<---->;если 'A>C', то переход на метку 'check_B'
34 00000124 000036000000      mov ecx, [C]<---->;иначе 'ecx = C'
35 00000129 0900000000      mov [max], ecx<---->;max = C
36
37
38 00000130 0010000000      check_B:
39 00000135 E062FFFF      mov eax, max
40 0000013B 0310000000      call atoi<---->;Вызов подпрограммы перевода символа в число
41
42 00000141 0000000000      mov [max], eax<---->;запись преобразованного числа в 'max'
43
44 00000145 3000000000      cmp ecx, [max]
45 0000014B 7F0C          jg fin<---->;Сравниваем 'max(A,C)' и 'B'
46 00000153 0900000000      mov ecx, [B]<---->;если 'max(A,C)>B', то переход на 'fin'
47
48 00000159 0010000000      mov [max], ecx<---->;иначе 'ecx = B'
49
50 0000015E E010FFFF      fin:
51 00000163 0110000000      mov eax, msg2
52 00000168 E019FFFF      call sprint<---->;Выход сообщения 'Наиболее число: '
53 0000016D 0059FFFF      call iprintf<---->;Выход 'max(A,B,C)'
54
55 00000172 0059FFFF      call quit<---->;Выход

```

Рис. 2.12: файл листинга lab7-2.lst

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

- 127 строка:

1. **0000009C** - адрес
2. **53** - машинный код
3. **push ebx** - используется для помещения значения регистра EBX на вершину стека. Когда происходит операция push, значение регистра EBX сохраняется на вершине стека, а указатель стека увеличивается на размер одного элемента (обычно 4 байта для 32-битной архитектуры).

- 128 строка:

1. **0000009D** - адрес
2. **51** - машинный код

3. **push ecx** - используется для помещения значения регистра EBX на вершину стека. Когда происходит операция push, значение регистра EBX сохраняется на вершине стека, а указатель стека увеличивается на размер одного элемента (обычно 4 байта для 32-битной архитектуры).

- 129 строка:

1. **0000009E** - адрес

2. **52** - машинный код

3. **push edx** - используется для помещения значения регистра EBX на вершину стека. Когда происходит операция push, значение регистра EBX сохраняется на вершине стека, а указатель стека увеличивается на размер одного элемента (обычно 4 байта для 32-битной архитектуры).

127	0000009C	53	<1>	push	ebx.....
128	0000009D	51	<1>	push	ecx.....
129	0000009E	52	<1>	push	edx.....

Рис. 2.13: 3 строки которые выбраны

Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд.

```
fin:
    mov eax, msg2
    call sprint<----->;Вывод сообщ
    mov eax, [max]
    call iprintLF<----->;Вывод 'max(
    call quit<-><----->;Выход
```

Рис. 2.14: один операнд который будет удалён


```

fin:
    mov eax
    call sprint<----->;Вывод сообщ
    mov eax, [max]
    call iprintLF<----->;Вывод 'max'
    call quit<-><----->;Выход

```

Рис. 2.15: один операнд который удалён

Выполните трансляцию с получением файла листинга:

```
nasm -f elf -l lab7-2.lst lab7-2.asm
```

Какие выходные файлы создаются в этом случае? Что добавляется в листинге?

В этот раз lab7-2.o не был создан только lab7-2.lst. В листинге пишет что появился сбой: недопустимая комбинация кода операции и операндов

```

48                                fin:
49                                mov eax
49                                ***** error: invalid combination of opcode and operands
50 00000159 E8B1FEFFFF            call sprint<----->;Вывод сообщения 'Наиболее число: '
51 0000015E A11000000001          mov eax, [max]
52 00000163 E81EFFFFFF            call iprintLF<----->;Вывод 'max(A,B,C)'
53 00000168 E86EFFFFFF            call quit<-><----->;Выход

```

Рис. 2.16: Сбой в листинге

3 Задание для самостоятельной работы

1. 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 No 7. Создайте исполняемый файл и проверьте его работу.

```

lab7-3.asm      [----]  9 L:  1+ 4  5/
%include 'in_out.asm'

section .data
msg2 db "Наименьшее число: ",0h
A dd '94'
B dd '5'
C dd '58'

section .bss
min resb 10

section .text
global _start
_start:

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

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

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

    ; -----
    mov ecx,[A]
    mov [min],ecx

    ; ----- Сравниваем 'A' и 'C'
    cmp ecx,[C]

    jl check_B

    mov ecx,[C]
    mov [min],ecx

    check_B:
    ; -----
    mov ecx,[min]
    cmp ecx,[B]
    jl fin
    mov ecx,[B]
    mov [min],ecx

    ; -----
    fin:

```

Рис. 3.1: Программа для самой-работы выбран из табл. 7.5

```

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐘 main [x?] took 7ms
[●] × mcedit lab7-3.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐘 main [x?] took 37s
Λ nasm -f elf lab7-3.asm

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐘 main [x?] took 8ms
Λ ld -m elf_i386 -o lab7-3 lab7-3.o

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐘 main [x?] took 22ms
Λ ./lab7-3
Наименьшее число: 5

mishanya4u@Legenda in ~/work/arch-pc/lab07 on 🐘 main [x?] took 1ms
Λ _

```

Рис. 3.2: Результат программы для самой-работы выбран из табл. 7.5

2. 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 No 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

```

#include 'in_out.asm'

section .data
msg1 DB 'Введите x: ',0h
msg2 DB 'Введите a: ',0h
msg3: DB 'Ответ: ',0h

section .bss
x: RESB 80
a: RESB 80

```

```
r: RESB 80
```

```
section .text
```

```
global _start
```

```
_start:
```

```
    mov eax,msg1
```

```
    call sprint
```

```
    mov ecx,x
```

```
    mov edx,80
```

```
    call sread
```

```
    mov eax,x
```

```
    call atoi
```

```
    mov [x],eax
```

```
    mov eax,msg2
```

```
    call sprint
```

```
    mov ecx,a
```

```
    mov edx,80
```

```
    call sread
```

```
    mov eax,a
```

```
    call atoi
```

```
    mov [a],eax
```

```
    mov eax, [x]
```

```
    cmp eax, 3
```

```
je x_ravno_3
```

```
mov eax, [a]
```

```
add eax, 1
```

```
jmp res
```

```
x_ravno_3:
```

```
mov eax, [x]
```

```
imul eax, 3
```

```
res:
```

```
mov [r], eax
```

```
fin:
```

```
mov eax, msg3
```

```
call sprint
```

```
mov eax, [r]
```

```
call iprintLF
```

```
call quit
```

4 Заключение

Теперь могу сказать, что умею создавать программы на ассемблере, которые могут принимать различные решения в зависимости от условий.