

Шаблон отчёта по лабораторной работе

7

Талебу Тенке Франк Устон НКАбд-05-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы :	6
2.1	Изучение структуры файлы листинга :	12
2.2	Выводы по результатам выполнения заданий :	14
3	Задание для самостоятельной работы :	15
3.1	Написание программы нахождения наименьшей из 3 целочислен- ных переменных :	15
4	Выводы по результатам выполнения заданий :	20
5	Выводы, согласованные с целью работы :	21
	Список литературы	22

Список иллюстраций

2.1	Ресунок	6
2.2	Ресунок	7
2.3	Ресунок	8
2.4	Ресунок	8
2.5	Ресунок	9
2.6	Ресунок	11
2.7	Ресунок	11
2.8	Ресунок	12
2.9	Ресунок	13
2.10	Ресунок	13
2.11	Ресунок	14
3.1	Ресунок	16
3.2	Ресунок	17
3.3	Ресунок	18
3.4	Ресунок	19
3.5	Ресунок	19

Список таблиц

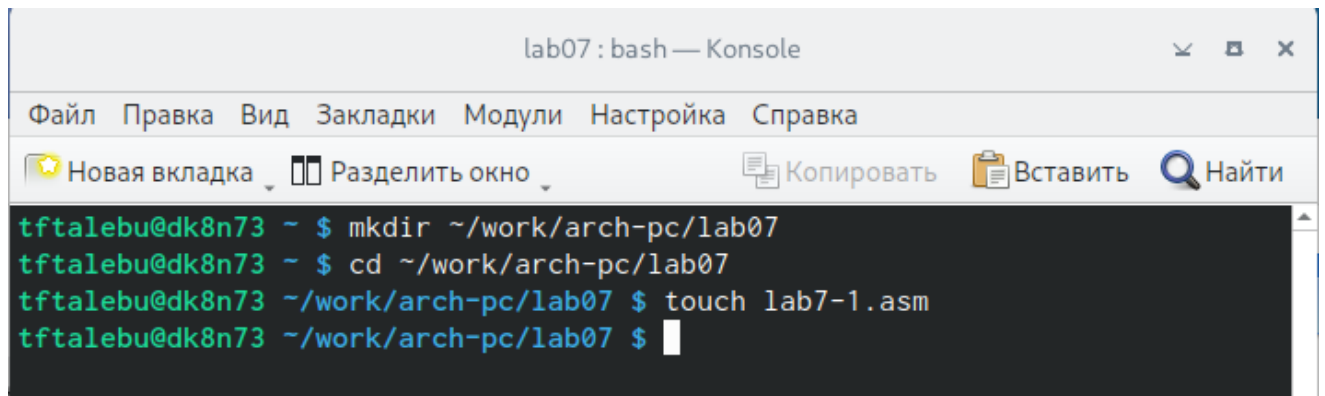
1 Цель работы

- мы узнаем о команде условных и безусловных переходов, делая это, мы освоим использование переходов, а также познакомимся со структурой файла листинга.

2 Выполнение лабораторной работы :

##Реализация переходов в NASM :

- Здесь мы начали с создания, а затем переместились “~/work/arch-pc/lab07”, после чего мы создали файл “lab7-1.asm”.(рис. [2.1])

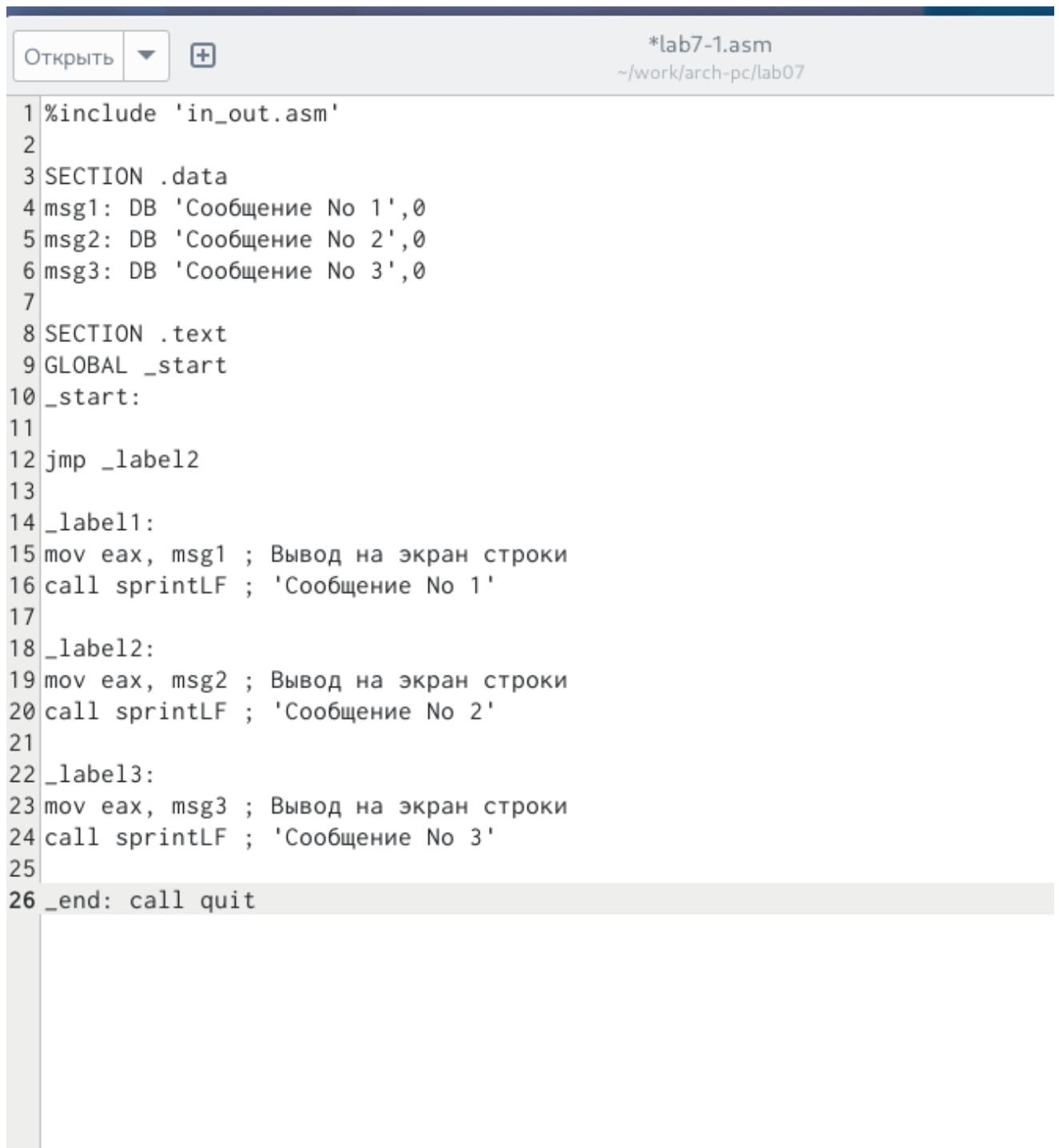


The image shows a terminal window titled "lab07 : bash — Konsole". The window has a menu bar with "Файл", "Правка", "Вид", "Закладки", "Модули", "Настройка", and "Справка". Below the menu bar is a toolbar with icons for "Новая вкладка", "Разделить окно", "Копировать", "Вставить", and "Найти". The terminal output shows the following commands and their results:

```
tftalebu@dk8n73 ~ $ mkdir ~/work/arch-pc/lab07
tftalebu@dk8n73 ~ $ cd ~/work/arch-pc/lab07
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ touch lab7-1.asm
tftalebu@dk8n73 ~/work/arch-pc/lab07 $
```

Рис. 2.1: Ресунок

- После этого мы заполнили файл .asm кодом программы, отображающей значение регистра eax.(рис. [2.2])



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение No 1',0
5 msg2: DB 'Сообщение No 2',0
6 msg3: DB 'Сообщение No 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call printf ; 'Сообщение No 1'
17
18 _label2:
19 mov eax, msg2 ; Вывод на экран строки
20 call printf ; 'Сообщение No 2'
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call printf ; 'Сообщение No 3'
25
26 _end: call quit
```

Рис. 2.2: Рисунок

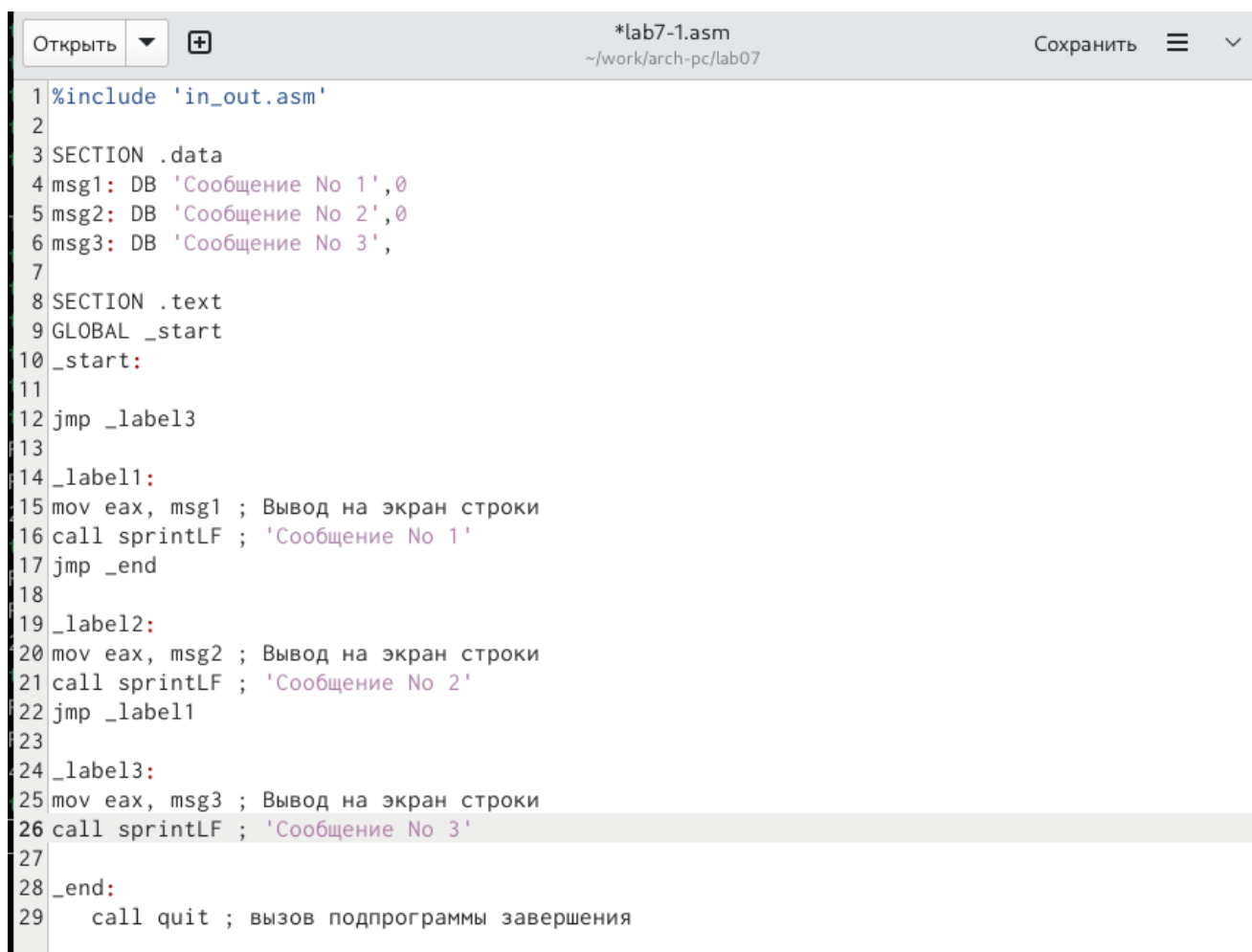
- Затем мы скомпилировали файл, создали исполняемый файл и запустили программу, все это после перемещения файла in_out.asm в тот же каталог,

где находится lab7-1.asm. (рис. [2.3])

```
of directory
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
tftalebu@dk8n73 ~/work/arch-pc/lab07 $
```

Рис. 2.3: Рисунок

- После этого мы изменили код в листинге.(рис. [2.4])



```
Открыть  + *lab7-1.asm Сохранить
~/work/arch-pc/lab07

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение No 1',0
5 msg2: DB 'Сообщение No 2',0
6 msg3: DB 'Сообщение No 3',
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call printf ; 'Сообщение No 1'
17 jmp _end
18
19 _label2:
20 mov eax, msg2 ; Вывод на экран строки
21 call printf ; 'Сообщение No 2'
22 jmp _label1
23
24 _label3:
25 mov eax, msg3 ; Вывод на экран строки
26 call printf ; 'Сообщение No 3'
27
28 _end:
29 call quit ; вызов подпрограммы завершения
```

Рис. 2.4: Рисунок

- Затем мы снова скомпилировали файл и создали исполняемый файл.(рис. [2.5])

```
of directory
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
tftalebu@dk8n73 ~/work/arch-pc/lab07 $
```

Рис. 2.5: Ресунок

- Затем мы снова изменили код в листинге ,чтобы вывод программы был следующим: user@dk4n31:~\$./lab7-1 Сообщение No 3 Сообщение No 2 Сообщение No 1 user@dk4n31:~\$ (рис. [??])(рис. [??])

The image shows a gedit editor window titled "lab7-1.asm" with the following assembly code:

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение No 1',0
5 msg2: DB 'Сообщение No 2',0
6 msg3: DB 'Сообщение No 3',
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call sprintf ; 'Сообщение No 1'
17 jmp _end
18
19 _label2:
20 mov eax, msg2 ; Вывод на экран строки
21 call sprintf ; 'Сообщение No 2'
22 jmp _label1
23
24 _label3:
25 mov eax, msg3 ; Вывод на экран строки
26 call sprintf ; 'Сообщение No 3'
27 jmp _label2
28
29 _end:
30 call quit ; вызов подпрограммы завершения
```

Below the editor is a terminal window showing the execution of the program:

```
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ gedit lab7-1.asm
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
tftalebu@dk8n73 ~/work/arch-pc/lab07 $
```

- После этого мы создали файл lab7-2.asm, в который мы добавим код нашей следующей программы (рис. [2.6])

```
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ touch lab7-2.asm
tftalebu@dk8n73 ~/work/arch-pc/lab07 $
```

Рис. 2.6: Ресунок

- После этого мы заполнили файл необходимым кодом для Программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C (рис. [2.7])

```
1 %include 'in_out.asm'
2
3 section .data
4     msg1 db 'Введите B: ',0h
5     msg2 db "Наибольшее число: ",0h
6     A dd '20'
7     C dd '50'
8 section .bss
9     max resb 10
10    B resb 10
11    section .text
12
13 global _start
14 _start:
15 ; ----- Вывод сообщения 'Введите B: '
16 mov eax,msg1
17 call sprint
18 ; ----- Ввод 'B'
19 mov ecx,B
20 mov edx,10
21 call sread
22 ; ----- Преобразование 'B' из символа в число
23 mov eax,B
24 call atoi ; Вызов подпрограммы перевода символа в число
25 mov [B],eax ; запись преобразованного числа в 'B'
26 ; ----- Записываем 'A' в переменную 'max'
27 mov ecx,[A] ; 'ecx = A'
28 mov [max],ecx ; 'max = A'
29 ; ----- Сравниваем 'A' и 'C' (как символы)
30 cmp ecx,[C] ; Сравниваем 'A' и 'C'
31 jg check_B ; если 'A>C', то переход на метку 'check_B',
32 mov ecx,[C] ; иначе 'ecx = C'
```

Рис. 2.7: Ресунок

- мы скомпилировали файл, создали исполняемый файл и запустили его.(рис. [2.8])

```
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 10
Наибольшее число: 50
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 51
Наибольшее число: 51
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 60
Наибольшее число: 60
tftalebu@dk8n73 ~/work/arch-pc/lab07 $
```

Рис. 2.8: Ресунок

2.1 Изучение структуры файлы листинга :

- Здесь и с помощью команды `nasm -f elf -l lab7-2.list lab7-2.asm` мы создали файл листинга файла `lab7-2.asm`, затем мы открыли файл с помощью `mcedit`. (рис. [2.9])

```

lab7-2.lst      [----]  0 L: 1+ 0 1/227] *(0 /14608b) 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>

```

Рис. 2.9: Ресунок

- мы выбрали эти три строки и пытаемся объяснить каждую из них.(рис. [2.10])

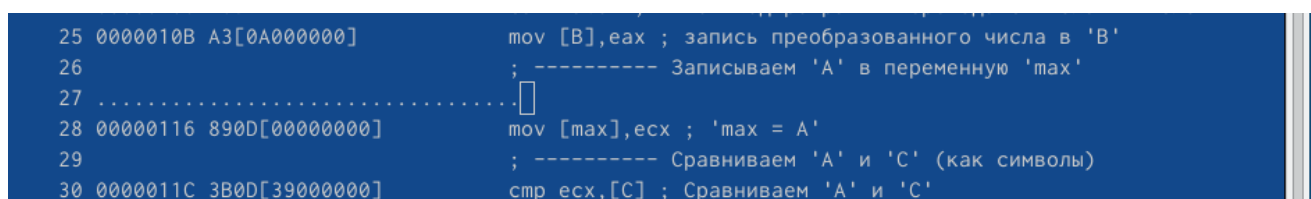
```

19 000000F2 B9[0A000000] mov ecx,B
20 000000F7 BA0A000000 mov edx,10
21 000000FC E842FFFFFF call sread

```

Рис. 2.10: Ресунок

- Здесь в 18-й строке мы переместили значение адреса переменной B в регистр `ecx`, после этого мы поместили значение 10 в регистре `edx`, который определяет размер переменной B с помощью подпрограммы `sread` и, наконец, мы вызвали подпрограмму `sread`
- мы открыли программный файл `lab 7-2.asm` и удалили один операнд в любой инструкции с двумя операндами. Мы выбрали строку под номером 27 (рис. [2.11])



```

25 0000010B A3[0A000000]      mov [B],ecx ; запись преобразованного числа в 'B'
26                          ; ----- Записываем 'A' в переменную 'max'
27 .....  

28 00000116 890D[00000000]      mov [max],ecx ; 'max = A'
29                          ; ----- Сравниваем 'A' и 'C' (как символы)
30 0000011C 3B0D[39000000]      cmp ecx,[C] ; Сравниваем 'A' и 'C'

```

Рис. 2.11: Ресунок

- В результате изменений был изменен файл листинга, в котором мы получили ошибку, объясняющую отсутствующий операнд, и файлы не были созданы.

2.2 Выводы по результатам выполнения заданий :

- Во время лабораторной работы мы узнали, как выполнять условные и безусловные переходы, как читать файл листинга.

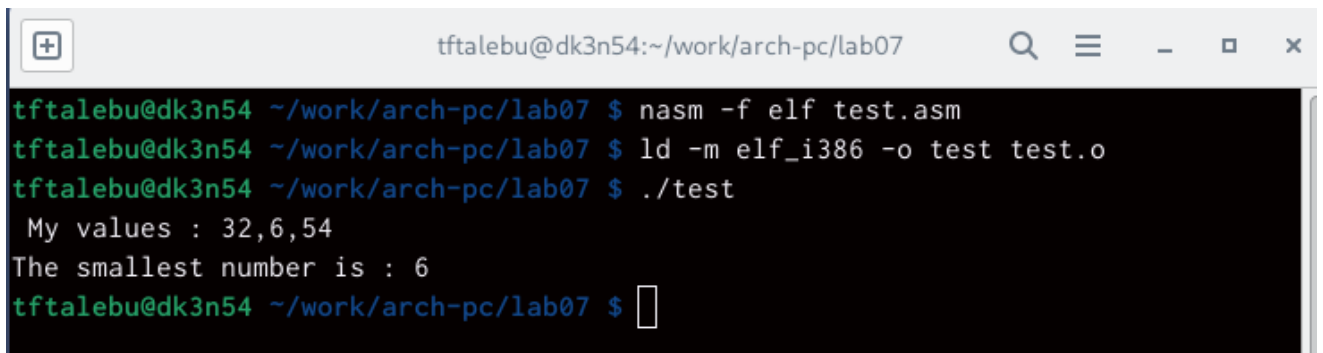
3 Задание для самостоятельной работы :

3.1 Написание программы нахождения наименьшей из 3 целочисленных переменных :

- Мой код : (рис. [3.1])

Рис. 3.1: Рисунок

- Вывод кода :(рис. [3.2])



```
tftalebu@dk3n54:~/work/arch-pc/lab07
tftalebu@dk3n54 ~/work/arch-pc/lab07 $ nasm -f elf test.asm
tftalebu@dk3n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o test test.o
tftalebu@dk3n54 ~/work/arch-pc/lab07 $ ./test
My values : 32,6,54
The smallest number is : 6
tftalebu@dk3n54 ~/work/arch-pc/lab07 $
```

Рис. 3.2: Ресунок

Мой код : (рис. [3.3])

```
~/work/arcn-pc/ia64/
1 %include 'in_out.asm'
2
3 SECTION .data
4 msgA: DB 'Please enter a value for a : ', 0
5 msgX: DB 'Please enter a value for x : ', 0
6 msg3: DB 'The result is : ', 0
7 SECTION .bss
8 A: RESB 80
9 X: RESB 80
10
11 SECTION .text
12 GLOBAL _start
13
14 _start:
15 mov eax, msgA
16 call sprint
17 mov ecx, A
18 mov edx, 80
19 call sread
20 mov eax, A
21 call atoi
22 mov [A], eax
23
24 mov eax, msgX
25 call sprint
26 mov ecx, X
27 mov edx, 80
28 call sread
29 mov eax, X
30 call atoi
31 mov [X], eax
32
33 mov ebx, [A]
34 cmp ebx, 7
```

Matlab ▾ Ширина табуляції: 8 ▾ Стр 49, Стлб 10 ▾ ВСТ

Рис. 3.3: Ресунок

Вывод кода :(рис. [3.4])

```

tftalebu@dk8n73 ~/work/arch-pc/lab07 $ nasm -f elf test-2.asm
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o test-2 test-2.o
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./test-2
Please enter a value for a : 3
Please enter a value for x : 9
27
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./test-2
Please enter a value for a : 6
Please enter a value for x : 4
24
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ ./test-2
Please enter a value for a : 5
Please enter a value for x : 9
45
tftalebu@dk8n73 ~/work/arch-pc/lab07 $ 

```

Рис. 3.4: Рисунок

:(рис. [3.5])

```

tftalebu@dk3n54 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07
/report $ git add .
tftalebu@dk3n54 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07
/report $ git commit -m "Add all files lab7"
[master cd8a4f9] Add all files lab7
4 files changed, 8 insertions(+), 17 deletions(-)
tftalebu@dk3n54 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07
/report $ git push
Перечисление объектов: 19, готово.
Подсчет объектов: 100% (19/19), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (10/10), готово.
Запись объектов: 100% (10/10), 188.83 КиБ | 2.45 МиБ/с, готово.
Всего 10 (изменений 6), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
to github.com:houstonTaleubou/study_2023-2024_arh-pc-.git
5893d85..cd8a4f9 master -> master
tftalebu@dk3n54 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07

```

Рис. 3.5: Рисунок

4 Выводы по результатам выполнения заданий :

- В этой части мы смогли применить наш полученный навык понятным способом, заставив программу вычислять конечное значение в зависимости от значений введенных переменных с использованием условных переходов.

5 Выводы, согласованные с целью работы :

- В 7 лаборатории мы в основном узнали, как использовать условные и безусловные переходы в NASM, как читать структуру файла листинга.

Список литературы