

Отчёт по лабораторной работе №7

дисциплина: Архитектура компьютера

Люкшина Влада Алексеевна

Содержание

1)Цель работы	6
2)Задание	7
3)Выполнение лабораторной работы	8
3.1) Создаем каталог для программ лабораторной работы № 7, переходим в него и создаем файл lab7-1.asm	8
3.2) Вводим в файл lab7-1.asm текст программы из листинга 7.1.	9
3.3) Создаем исполняемый файл и запускаем его.	10
3.4) Изменяем текст программы в соответствии с листингом 7.2.	11
3.5) Создаем исполняемый файл и проверяем его работу.	12
3.6) Изменяем текст программы, чтобы программа выводила сообщения в обратном порядке(сообщение №3, сообщение №2, сообщение №1).	13
3.7) Создаем исполняемый файл и запускаем его.	14
3.8) Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.	14
3.9) Вводим в файл текст из листинга 7.3.	16
3.10) Создаем исполняемый файл и проверяем его работу для разных значений В.	17
3.11) Создаем файл листинга для программы из файла lab7-2.asm.	17
3.12) Открываем файл листинга lab7-2.lst с помощью текстового редактора mcedit.	18
3.13) Ознакомимся с содержимым листинга и объясним три строки из него.	18
3.14) Открываем файл с программой lab7-2.asm и в инструкции с двумя операндами удаляем один операнд.	19
3.15) Выполняем трансляцию с получением файла листинга.	19
3.16) Смотрим, какие изменения произошли после трансляции файла.	20
3.17) Открываем листинг.	21
4) Самостоятельная работа	22
4.1) Создаем новый файл lab7-3.asm.	22
4.2) Пишем программу нахождения наименьшего значения из трех. Два значения(“а” и “с” прописываем в команде, “b” вводим с консоли).	24
4.3) Запускаем программу и вводим значения исходя из листинга 7.5(вариант 7).	25
4.4) Создаем новый файл lab7-4.asm для выполнения второго задания.	25
4.4) Пишем программу для вычисления заданной функции(вариант 3).	27

4.5) Запускаем программу и вводим значения из варианта 3.	28
5)Выводы	29

Список иллюстраций

1	Создаем каталог и файл	8
2	Вводим текст	9
3	Создаем и запускаем	10
4	Редактируем файл	11
5	Создаем и проверяем	12
6	Редактируем текст	13
7	Создаем и запускаем	14
8	Создаем файл	14
9	Вводим текст	16
10	Создаем файл и проверяем его работу	17
11	Создаем файл листинга	17
12	Открываем файл листинга	18
13	Удаляем операнд	19
1	Создаем файл	22
2	Пишем программу	24
3	Запускаем программу	25
4	Создаем файл	25
5	Пишем программу	27
6	Запускаем программу	28

Список таблиц

1)Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Изучить назначения и структуры файла листинга.

2)Задание

Написать программы для вычисления значений функция и нахождения наименьших и наибольших переменных.

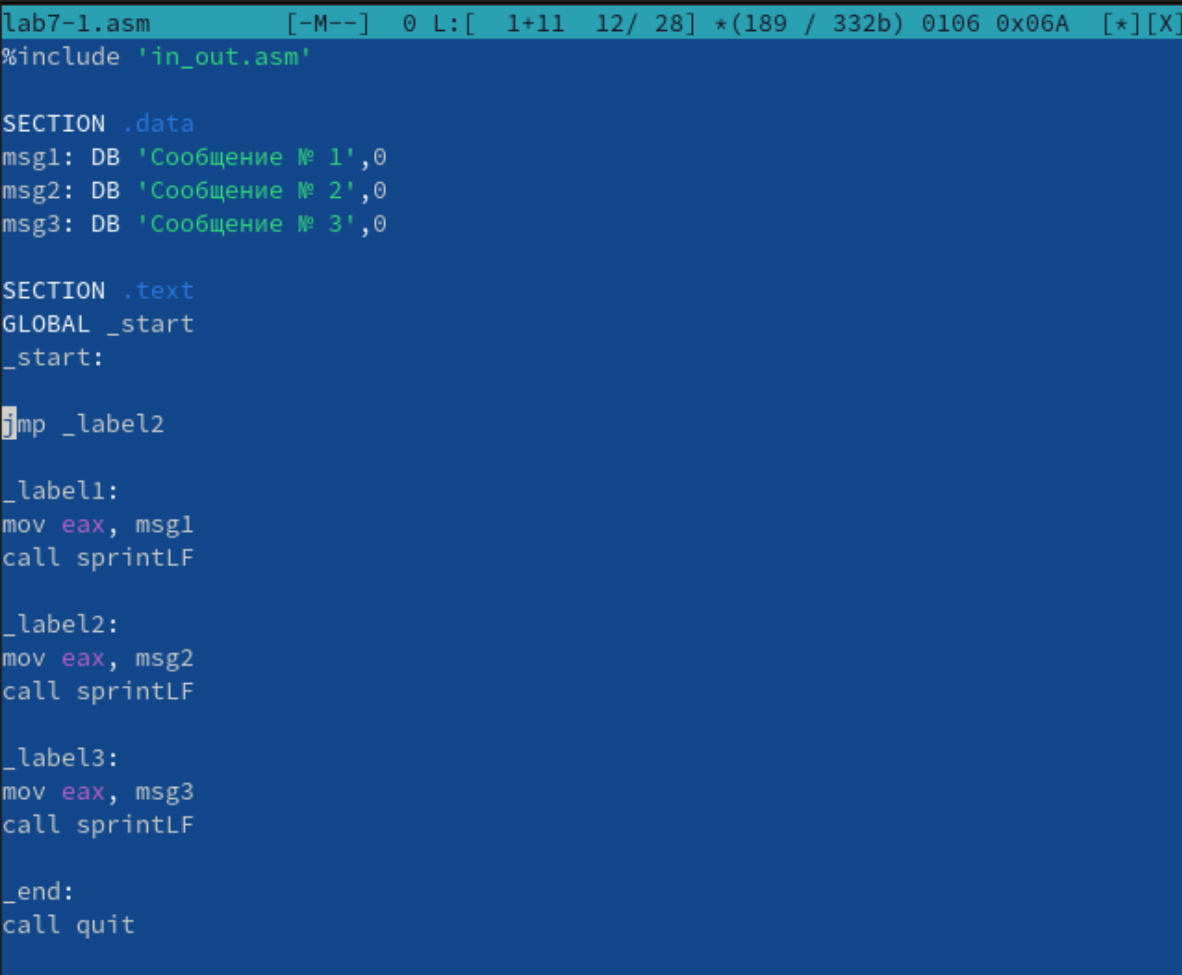
3)Выполнение лабораторной работы

3.1) Создаем каталог для программ лабораторной работы № 7, переходим в него и создаем файл lab7-1.asm

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

Рис. 1: Создаем каталог и файл

3.2) Вводим в файл lab7-1.asm текст программы из листинга 7.1.



```
lab7-1.asm [-M--] 0 L:[ 1+11 12/ 28] *(189 / 332b) 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

_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

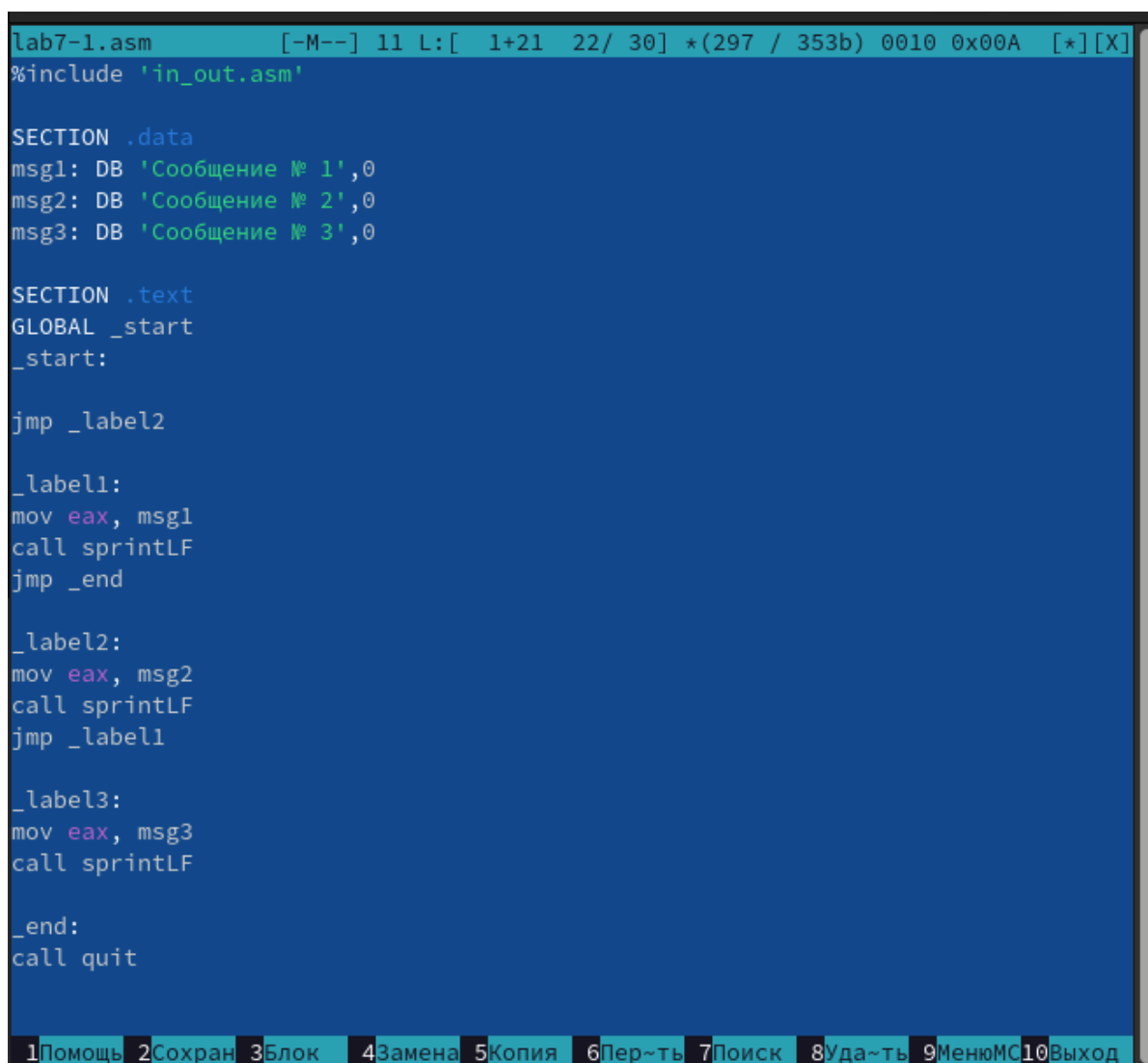
Рис. 2: Вводим текст

3.3) Создаем исполняемый файл и запускаем его.

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

Рис. 3: Создаем и запускаем

3.4) Изменяем текст программы в соответствии с листингом 7.2.



```
lab7-1.asm      [-M--] 11 L: [ 1+21 22/ 30] *(297 / 353b) 0010 0x00A  [*] [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:
mov eax, msg3
call sprintLF

_end:
call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС 10Выход

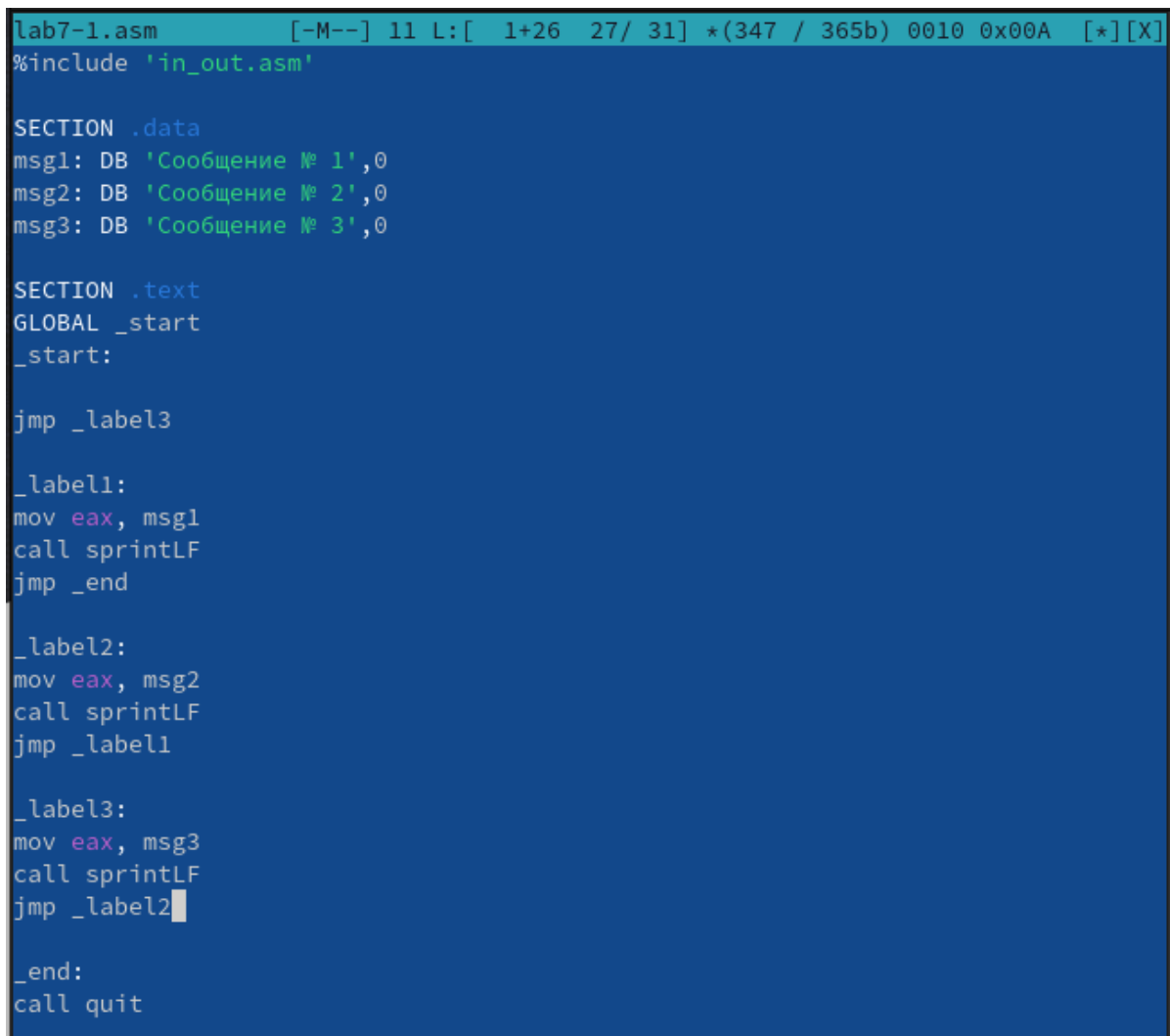
Рис. 4: Редактируем файл

3.5) Создаем исполняемый файл и проверяем его работу.

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

Рис. 5: Создаем и проверяем

3.6) Изменяем текст программы, чтобы программа выводила сообщения в обратном порядке(сообщение №3, сообщение №2, сообщение №1).



```
lab7-1.asm  [-M--] 11 L: [ 1+26 27/ 31] *(347 / 365b) 0010 0x00A  [*] [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

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 6: Редактируем текст

3.7) Создаем исполняемый файл и запускаем его.

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

Рис. 7: Создаем и запускаем

3.8) Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.

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

Рис. 8: Создаем файл

3.9) Вводим в файл текст из листинга 7.3.

```
lab7-2.asm [----] 9 L:[ 1+10 11/ 50] *(193 / 551b) 0116 0x074 [*][X]
#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:

mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

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

mov ecx,[A]
mov [max],ecx

cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx

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

mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx

fin:
mov eax, msg2
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

Рис. 9: Вводим текст

3.10) Создаем исполняемый файл и проверяем его работу для разных значений В.

```
lyukshinava@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
lyukshinava@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
lyukshinava@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
lyukshinava@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 80
Наибольшее число: 80
lyukshinava@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 70
Наибольшее число: 70
lyukshinava@fedora:~/work/arch-pc/lab07$
```

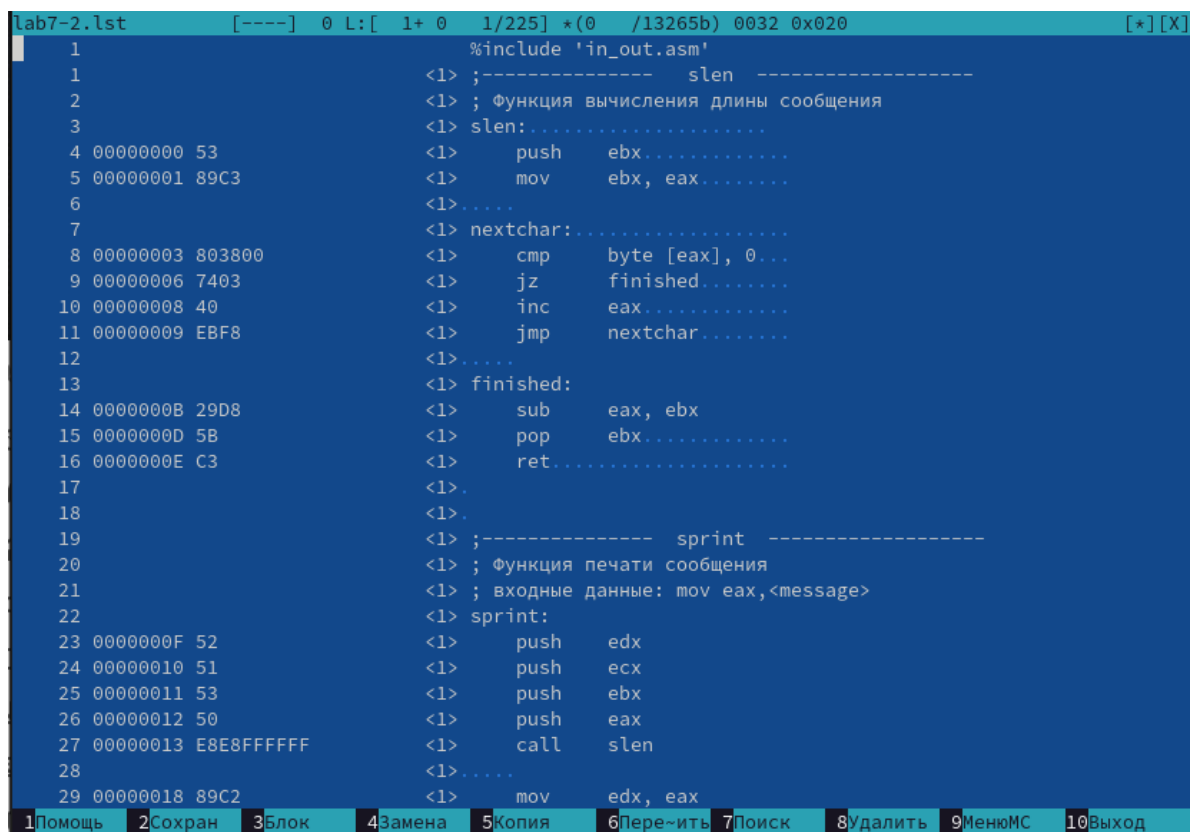
Рис. 10: Создаем файл и проверяем его работу

3.11) Создаем файл листинга для программы из файла lab7-2.asm.

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

Рис. 11: Создаем файл листинга

3.12) Открываем файл листинга lab7-2.lst с помощью текстового редактора mcedit.



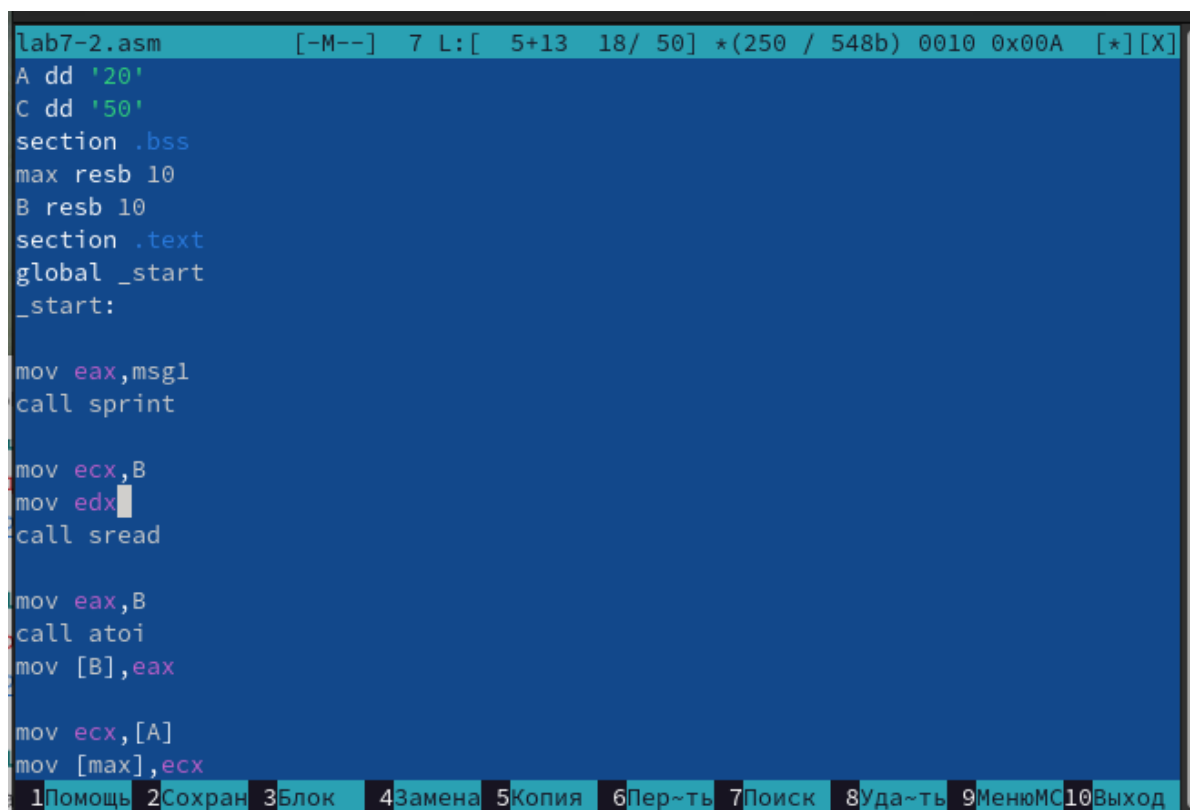
```
lab7-2.lst [----] 0 L: 1+ 0 1/225] *(0 /13265b) 0032 0x020 [*][X]
1      %include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:.....
5      00000000 53      <1>      push     ebx.....
6      00000001 89C3    <1>      mov      ebx, eax.....
7      <1>.....
8      <1> nextchar:.....
9      00000003 803800  <1>      cmp      byte [eax], 0...
10     00000006 7403    <1>      jz       finished.....
11     00000008 40      <1>      inc      eax.....
12     00000009 EBF8    <1>      jmp      nextchar.....
13     <1>.....
14     <1> finished:
15     0000000B 29D8    <1>      sub      eax, ebx
16     0000000D 5B      <1>      pop      ebx.....
17     0000000E C3      <1>      ret.....
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 E8E8FFFFFF <1>      call     slen
28     <1>.....
29     00000018 89C2    <1>      mov      edx, eax
```

Рис. 12: Открываем файл листинга

3.13) Ознакомимся с содержимым листинга и объясним три строки из него.

Строка 65: 00000045 - адрес в сегменте кода, BВ00000000 - машинный код, mov ebx,0 - присвоение переменной ebx значения 0. Строка 66: 0000004A - адрес в сегменте кода, B803000000 - машинный код, mov eax,3 - присвоение переменной ebx значения 3. Строка 67: 0000004F - адрес в сегменте кода, CD80 - машинный код, int 80h - вызов ядра.

3.14) Открываем файл с программой lab7-2.asm и в инструкции с двумя операндами удаляем один операнд.



```
lab7-2.asm [-M--] 7 L:[ 5+13 18/ 50] *(250 / 548b) 0010 0x00A [*][X]
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,B
mov edx,
call sread

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

mov ecx,[A]
mov [max],ecx
```

Рис. 13: Удаляем операнд

3.15) Выполняем трансляцию с получением файла листинга.

```
lyukshinava@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands
```

Выдается ошибка.

3.16) Смотрим, какие изменения произошли после трансляции файла.

```
lyukshinava@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
lyukshinava@fedora:~/work/arch-pc/lab07$
```

Создались выходные файлы lab7-2 и lab7-2.lst.

3.17) Открываем листинг.

```
12      _start:
13
14 000000E8 B8[00000000]      mov eax,msg1
15 000000ED E81DFFFFFF      call sprint
16
17 000000F2 B9[0A000000]      mov ecx,B
18      mov edx
18      *****      error: invalid combination of opcode and operands
19 000000F7 E847FFFFFF      call sread
20
21 000000FC B8[0A000000]      mov eax,B
22 00000101 E896FFFFFF      call atoi
23 00000106 A3[0A000000]      mov [B],eax
24
25 0000010B 8B0D[35000000]      mov ecx,[A]
26 00000111 890D[00000000]      mov [max],ecx
27
28 00000117 3B0D[39000000]      cmp ecx,[C]
29 0000011D 7F0C      jg check_B
30 0000011F 8B0D[39000000]      mov ecx,[C]
31 00000125 890D[00000000]      mov [max],ecx
32
33      check_B:
34 0000012B B8[00000000]      mov eax,max
35 00000130 E867FFFFFF      call atoi
36 00000135 A3[00000000]      mov [max],eax
37
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
44      fin:
45 00000154 B8[13000000]      mov eax, msg2
46 00000159 E8B1FEFFFF      call sprint
47 0000015E A1[00000000]      mov eax,[max]
48 00000163 E81EFFFFFF      call iprintLF
49 00000168 E86EFFFFFF      call quit
```

Замечаем, что в листинге так же добавилось сообщение об ошибке.

4) Самостоятельная работа

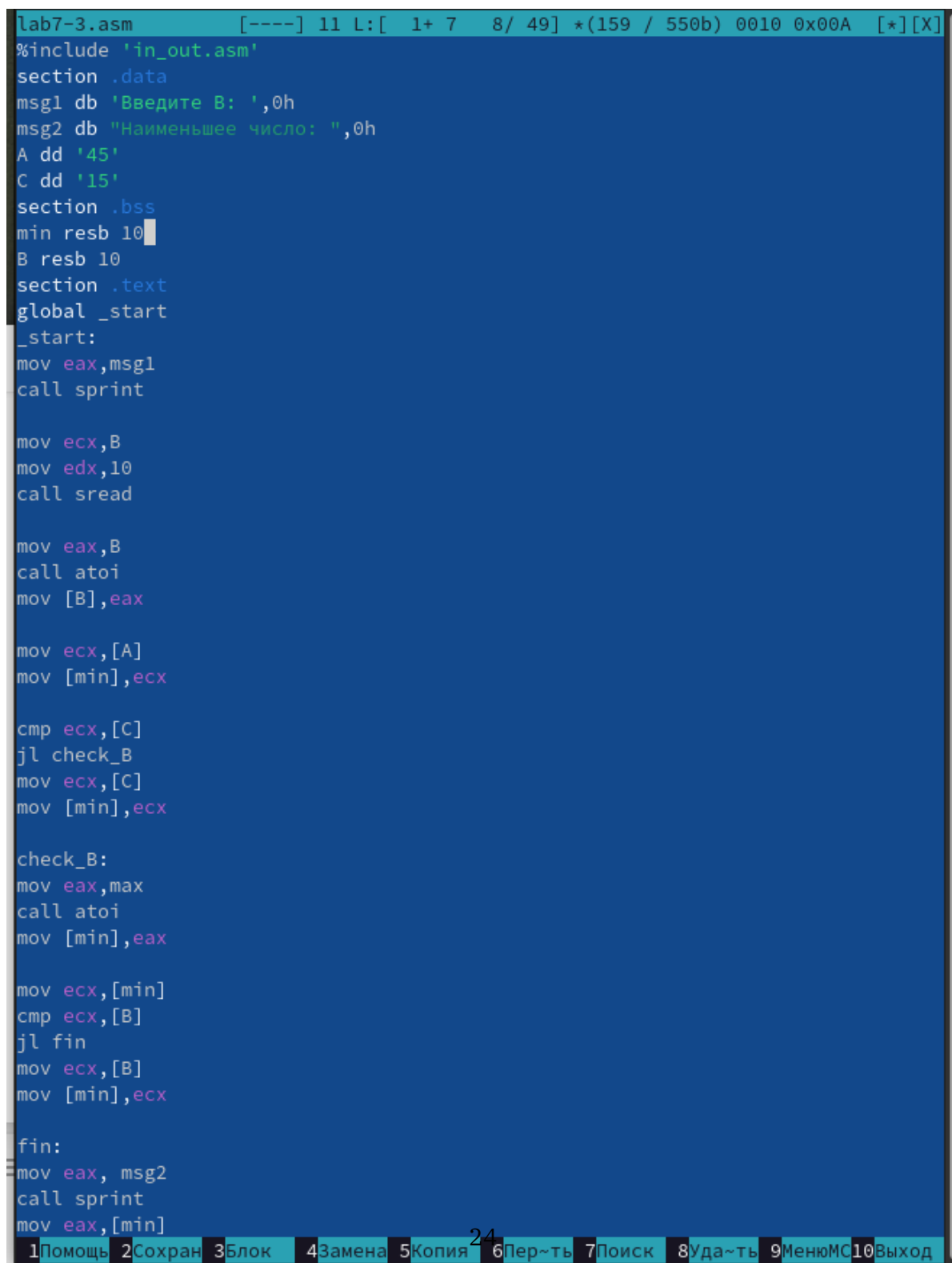
4.1) Создаем новый файл lab7-3.asm.



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

Рис. 1: Создаем файл

4.2) Пишем программу нахождения наименьшего значения из трех. Два значения(“а” и “с” прописываем в команде, “b” вводим с консоли).



```
[lab7-3.asm] [----] 11 L:[ 1+ 7 8/ 49] *(159 / 550b) 0010 0x00A [*][X]
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '45'
C dd '15'
section .bss
min resb 10
B resb 10
section .text
global _start
_start:
mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

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

mov ecx,[A]
mov [min],ecx

cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx

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

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

fin:
mov eax, msg2
call sprint
mov eax,[min]
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 2: Пишем программу

4.3) Запускаем программу и вводим значения исходя из листинга 7.5(вариант 7).

```
lyukshinava@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
lyukshinava@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
lyukshinava@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите В: 67
Наименьшее число: 15
lyukshinava@fedora:~/work/arch-pc/lab07$
```

Рис. 3: Запускаем программу

4.4) Создаем новый файл lab7-4.asm для выполнения второго задания.

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

Рис. 4: Создаем файл

4.4) Пишем программу для вычисления заданной функции(вариант 3).

```
GNU nano 7.2 /home/lyukshinava/work/arch-pc/lab07/lab7-4.asm Изменён
#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
    res: 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 check_a
    mov eax,[a]
    add eax,1
    mov [res],eax
    jmp fin
check_a:
    mov eax,[x]
    xor ebx,ebx
    mov ebx,3
    mul ebx
    mov [res],eax
    jmp fin
fin:
    mov eax,msg3
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/ К строке

4.5) Запускаем программу и вводим значения из варианта 3.



```
lyukshinava@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите X: 3
Введите A: 4
Ответ: 9
lyukshinava@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите X: 1
Введите A: 4
Ответ: 5
lyukshinava@fedora:~/work/arch-pc/lab07$
```

Рис. 6: Запускаем программу

5)Выводы

Мы изучили структуру файла листинга, команды условного и безусловного перехода.