

Лабораторная работа №7

Дисциплина: архитектура компьютера

Мусатова Екатерина Викторовна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Самостоятельная работа	14
4	Выводы	18

Список иллюстраций

2.1	Создание файла	6
2.2	Ввод текста программы	7
2.3	Проверка	7
2.4	Изменение программы	8
2.5	Проверка	8
2.6	Изменение программы	9
2.7	Проверка	9
2.8	Создание файла	10
2.9	Ввод текста	10
2.10	Проверка	11
2.11	Создание файла листинга	11
2.12	Открытие файла листинга	12
2.13	Удаление операнда	12
2.14	Трансляция	12
2.15	Проверка листинга	13
3.1	Проверка	14
3.2	Написание программы	15
3.3	Проверка	15

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1

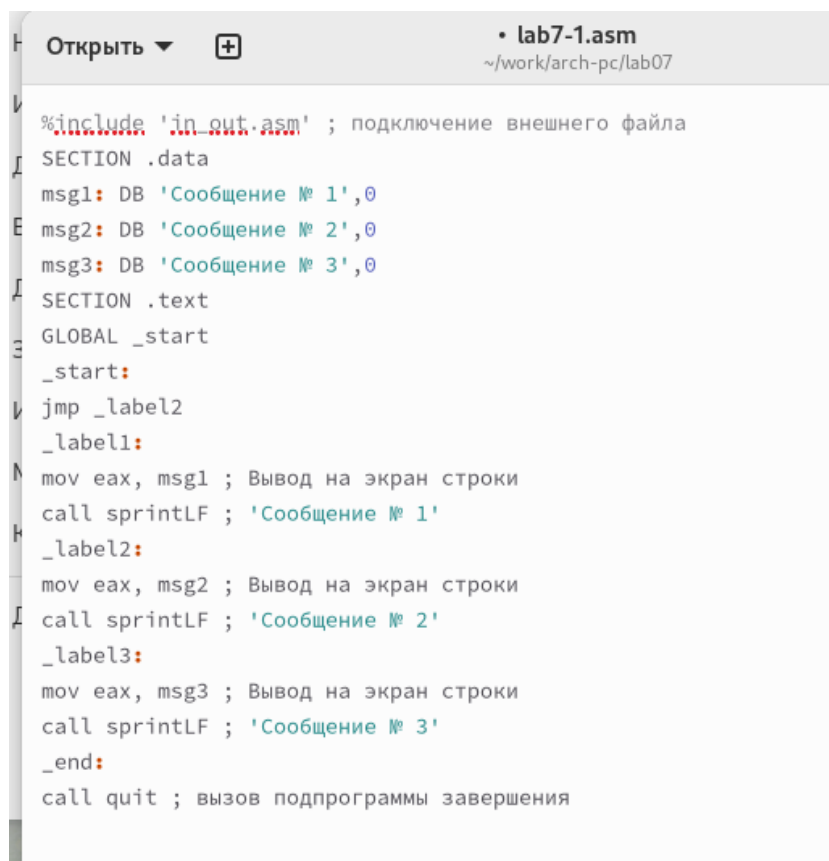
Создаю каталог для лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. 2.1).

```
[evmusatova@fedora ~]$ mkdir ~/work/arch-pc/lab07  
[evmusatova@fedora ~]$ cd ~/work/arch-pc/lab07  
[evmusatova@fedora lab07]$ touch lab7-1.asm  
[evmusatova@fedora lab07]$
```

Рис. 2.1: Создание файла

2

Ввожу в файл lab7-1.asm текст программы из листинга 7.1 (рис. 2.2).

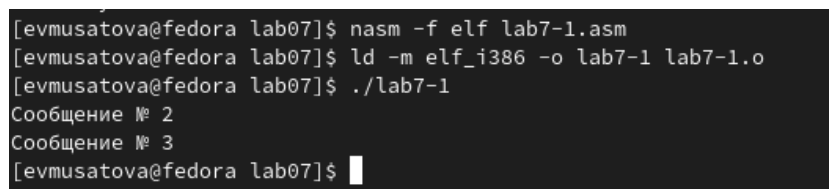


```
Открыть ▾ + • lab7-1.asm
~/work/arch-pc/lab07

%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'
_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: Ввод текста программы

Создаю исполняемый файл и проверяю правильность работы программы (рис. 2.3).



```
[evmusatova@fedora lab07]$ nasm -f elf lab7-1.asm
[evmusatova@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[evmusatova@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[evmusatova@fedora lab07]$
```

Рис. 2.3: Проверка

Изменим программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу в соответствии с листингом 7.2 (рис. 2.4).

```

~/work/касп-релаб07
%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: Изменение программы

Создаю исполняемый файл и проверяю работу программы (рис. 2.5).

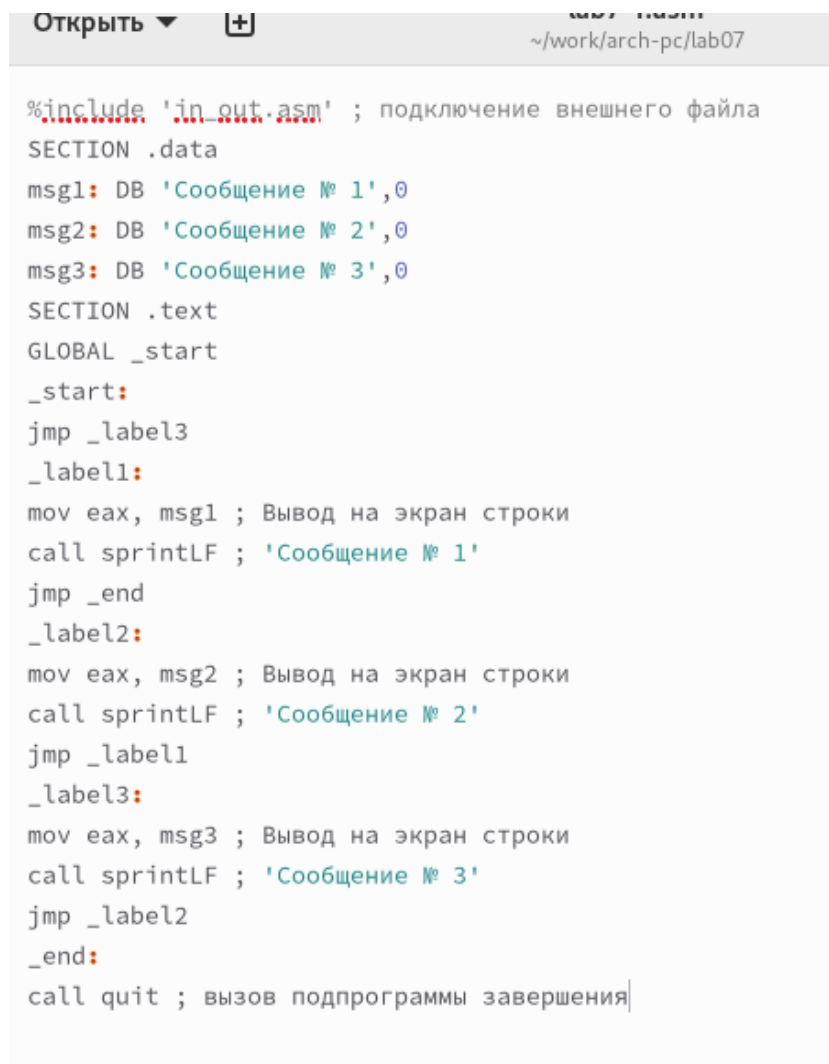
```

Сообщение № 3
[evmusatova@fedora lab07]$ nasm -f elf lab7-1.asm
[evmusatova@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[evmusatova@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[evmusatova@fedora lab07]$ 

```

Рис. 2.5: Проверка

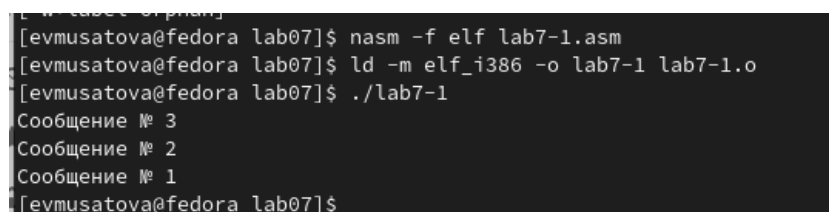
Изменяю текст программы добавляя и изменяя инструкции jmp (рис. 2.6).



```
Открыть ▾ [+]  
~/work/arch-pc/lab07  
  
%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 sprintfLF ; 'Сообщение № 1'  
jmp _end  
_label2:  
mov eax, msg2 ; Вывод на экран строки  
call sprintfLF ; 'Сообщение № 2'  
jmp _label1  
_label3:  
mov eax, msg3 ; Вывод на экран строки  
call sprintfLF ; 'Сообщение № 3'  
jmp _label2  
_end:  
call quit ; вызов подпрограммы завершения
```

Рис. 2.6: Изменение программы

Создаю исполняемый файл и проверяю работу программы (рис. 2.7).



```
[evmusatova@fedora lab07]$ nasm -f elf lab7-1.asm  
[evmusatova@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o  
[evmusatova@fedora lab07]$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
[evmusatova@fedora lab07]$
```

Рис. 2.7: Проверка

Создаю файл lab7-2.asm (рис. 2.8).

```
[evmusatova@fedora lab07]$ touch lab7-2.asm
[evmusatova@fedora lab07]$
```

Рис. 2.8: Создание файла

В созданный файл ввожу текст из листинга 7.3 (рис. 2.9).

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

%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
```

Рис. 2.9: Ввод текста

Создаю исполняемый файл и проверяю его работу для разных значений B (рис. 2.10).

```

[evmusatova@fedora lab07]$ nasm -f elf lab7-2.asm
[evmusatova@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[evmusatova@fedora lab07]$ ./lab7-2
Введите B: 5
Наибольшее число: 50
[evmusatova@fedora lab07]$ ./lab7-2
Введите B: 22
Наибольшее число: 50
[evmusatova@fedora lab07]$ ./lab7-2
Введите B: 55
Наибольшее число: 55
[evmusatova@fedora lab07]$

```

Рис. 2.10: Проверка

4

Создаю файл листинга для программы из файла lab7-2.asm (рис. 2.11).

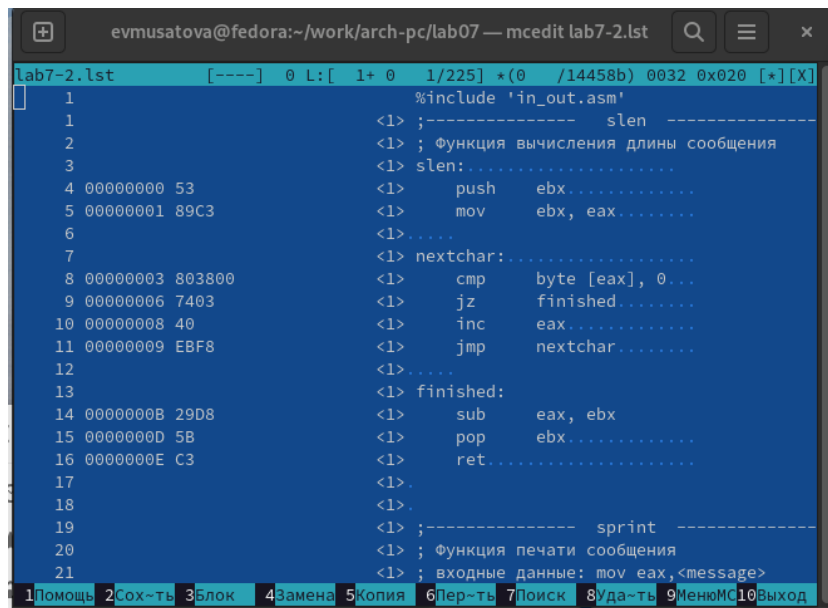
```

[evmusatova@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[evmusatova@fedora lab07]$

```

Рис. 2.11: Создание файла листинга

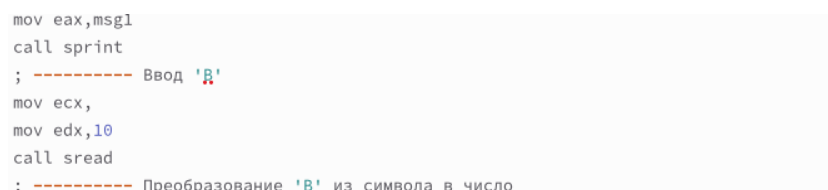
Открываю файл листинга с помощью текстового редактора mcedit (рис. 2.12). В строке 9 содержится номер сторки [8], адрес [00000003], машинный код [803800] и содержимое строки кода [cmp byte [eax], 0] в строке 11 содержится номер сторки [10], адрес [00000008], машинный код [40] и содержимое строки кода [inc eax] в строке 24 содержится номер сторки [23], адрес [0000000F], машинный код [52] и содержимое строки кода [push edx].



```
lab7-2.lst  [----]  0 L: [ 1+ 0 1/225] *(0 /14458b) 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>
```

Рис. 2.12: Открытие файла листинга

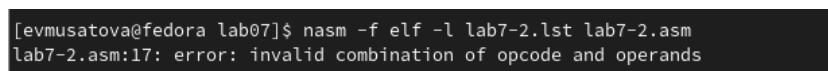
Открываю файл с программой lab7-2.asm и в инструкции с двумя операндами удаляю операнд В (рис. 2.13).



```
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
```

Рис. 2.13: Удаление операнда

Выполняю трансляцию с получением файла листинга (рис. 2.14).



```
[evmusatova@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
```

Рис. 2.14: Трансляция

Проверяю файл листинга и вижу, что если в коде появляется ошибка, то ее описание появится в файле листинга (рис. 2.15).

```

15 000000ED E810FFFFFF      call sprintf
16                      ; ----- Ввод 'B'
17                      mov ecx,
17                      ***** error: invalid combination of opcode and operands
18 000000F2 BA0A000000      mov edx,10
19 000000F7 E847FFFFFF      call sread
20                      ; ----- Преобразование 'B' из символа в число

```

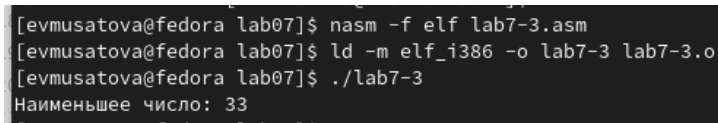
Рис. 2.15: Проверка листинга

3 Самостоятельная работа

Создаю файл lab7-3 и пишу программу из 3 целочисленных переменных A, B и C в соответствии с 8 вариантом (рис. ??).

! [Написание программы](image/16.png){#fig:016 width=70%}

Создаю исполняемый файл и проверяю работу программы (рис. 3.1).



```
[evmusatova@fedora lab07]$ nasm -f elf lab7-3.asm
[evmusatova@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[evmusatova@fedora lab07]$ ./lab7-3
Наименьшее число: 33
```

Рис. 3.1: Проверка

Создаю файл lab7-4 и пишу программу, которая для введенных с клавиатуры значений ☒ и ☒ вычисляет значение заданной функции из варианта 8 (рис. 3.2).

```

1  %include 'in_out.asm'
2
3  section .data
4      msgX db "x = ",0h
5      msgA db "a = ",0h
6
7  section .bss
8      x resb 10
9      a resb 10
10     f resb 10
11
12 section .text
13 global _start
14
15 _start:
16     ; ----- Ввод 'X'
17     mov eax, msgX
18     call sprint
19     mov ecx, x
20     mov edx, 10
21     call sread
22
23     ; ----- Ввод 'A'
24     mov eax, msgA
25     call sprint
26     mov ecx, a
27     mov edx, 10
28     call sread
29
30     ; ----- Преобразование 'X' из символа в число
31     mov eax, x
32     call atoi
33     mov [x], eax
34
35     ; ----- Преобразование 'a' из символа в число

```

Рис. 3.2: Написание программы

Создаю исполняемый файл и проверяю работу программы (рис. 3.3).

```

[evmusatova@fedora lab07]$ nasm -f elf lab7-4.asm
[evmusatova@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[evmusatova@fedora lab07]$ ./lab7-4
x = 1
a = 4
2
[evmusatova@fedora lab07]$ ./lab7-4
x = 1
a = 2
6
[evmusatova@fedora lab07]$

```

Рис. 3.3: Проверка

Листинг к заданию 1

```

%include 'in_out.asm' section .data msg1 db "Наименьшее число:",0h a dd 33 b
dd 40 c dd 52 section .bss min resb 10 section .text global _start _start: mov ecx, [a]
mov [min], ecx ; 'min = A' ; ----- Сравниваем 'A' и 'C' (как числа) 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: ; ----- 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, msg1 call sprint mov eax,[min] call iprintLF ; Вывод 'min(A,B,C)' call quit ;
Выход

```

****Листинг к заданию 2****

```

%include 'in_out.asm'
section .data msgX db "x =",0h msgA db "a =",0h
section .bss x resb 10 a resb 10 f resb 10
section .text global _start
_start: ; ----- Ввод 'X' mov eax, msgX call sprint mov ecx, x mov edx,10 call sread

; ----- Ввод 'A'
mov eax, msgA
call sprint
mov ecx, a
mov edx,10
call sread

; ----- Преобразование 'x' из символа в число
mov eax, x
call atoi
mov [x], eax

```



```

; ----- Преобразование 'a' из символа в число
mov eax, a
call atoi
mov [a], eax

mov ecx, [a]
cmp ecx, 3 ;сравниваем a и цифру 3

ja newfunc ;если a больше то идем по метке

mov eax, [a] ;иначе a умножаем на 3
mov ebx, 3
mul ebx
jmp fin

newfunc: mov eax, [x] mov ebx, 1 add eax, ebx
fin: call iprintLF call quit

```

4 Выводы

Я изучила команды условного и безусловного переходов и научилась писать программы с использованием этих переходов.