

Отчет по лабораторной работе №8

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

Попова Елизавета Сергеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Самостоятельная работа	13
4	Выводы	17
	Список литературы	18

Список иллюстраций

2.1	Создание файла lab8-1.asm	6
2.2	Файл lab8-1.asm	7
2.3	Запуск программы lab8-1.asm	7
2.4	Изменение текста программы	8
2.5	Проверка работы программы	8
2.6	Изменение текста программы	9
2.7	Запуск программы	9
2.8	Текст программы lab8-2.asm	10
2.9	Программа lab8-2.asm	11
2.10	Файл листинга lab8-2.lst	11
2.11	Первая строка	11
2.12	Вторая строка	12
2.13	Третья строка	12
2.14	Попытка создать файл с ошибкой	12
2.15	Ошибка в файле листинга	12
3.1	Текст программы	14
3.2	Результат работы программы	15
3.3	Текст программы	15
3.4	Результат работы программы	16

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.

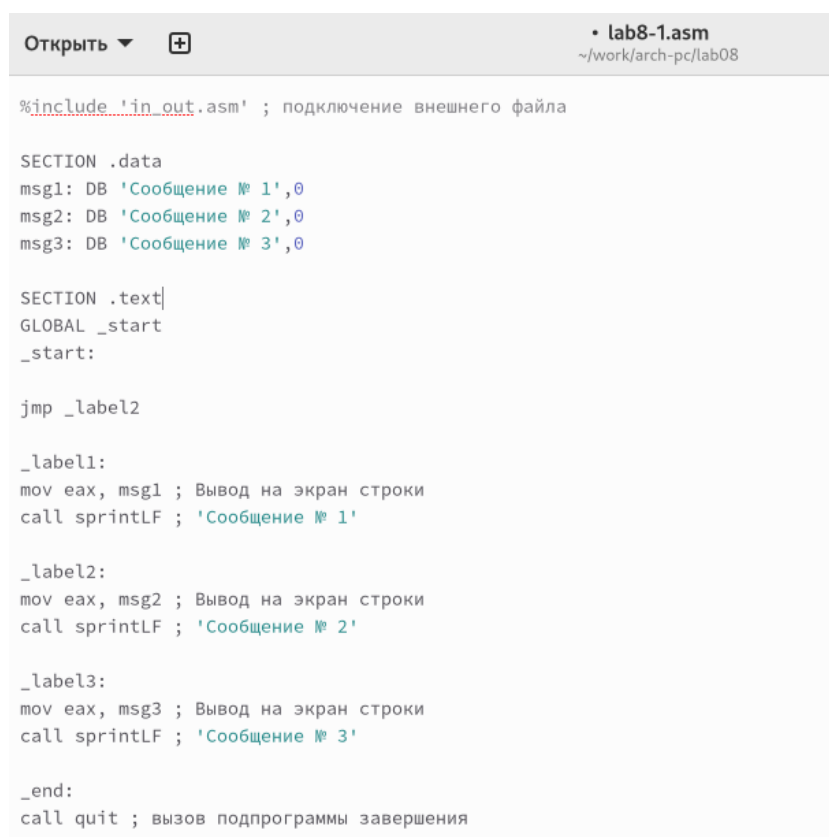
2 Выполнение лабораторной работы

- 1) Я создала каталог lab08, где создала файл lab8-1.asm

```
[laisacrete@10 ~]$ cd work/arch-pc  
[laisacrete@10 arch-pc]$ mkdir lab08  
[laisacrete@10 arch-pc]$ cd lab08  
[laisacrete@10 lab08]$ touch lab8-1.asm  
[laisacrete@10 lab08]$ ls  
lab8-1.asm  
[laisacrete@10 lab08]$ pwd  
/home/laisacrete/work/arch-pc/lab08
```

Рис. 2.1: Создание файла lab8-1.asm

- 2) Я ввела нужный текст программы в файл и затем запустила его.




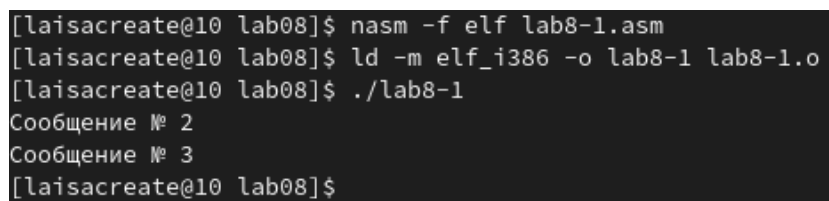
```
Открыть ▾  • lab8-1.asm  
~/work/arch-pc/lab08  
  
%include 'in_out.asm' ; подключение внешнего файла  
  
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
  
SECTION .text  
GLOBAL _start  
_start:  
  
jmp _label2  
  
_label1:  
mov eax, msg1 ; 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: Файл lab8-1.asm

- 3) Я создала исполняемый файл и запустила его. Результат работы программы верный.



```
[laisacrete@10 lab08]$ nasm -f elf lab8-1.asm  
[laisacrete@10 lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o  
[laisacrete@10 lab08]$ ./lab8-1  
Сообщение № 2  
Сообщение № 3  
[laisacrete@10 lab08]$
```

Рис. 2.3: Запуск программы lab8-1.asm

- 4) Я изменила текст программы так, чтобы выводился нужный ответ и снова создала исполняемый файл.

```

%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 текста программы

```

[laisacreate@10 lab08]$ nasm -f elf lab8-1.asm
[laisacreate@10 lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[laisacreate@10 lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1

```

Рис. 2.5: Проверка работы программы

- 5) Я изменила текст программы так, чтобы сперва выводилось сообщение №3, потом сообщение №2 и в конце сообщение №1.


```

%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: Изменение текста программы

6) Я запустила программу и проверила корректность результата.

```

[laisacreate@10 lab08]$ nasm -f elf lab8-1.asm
[laisacreate@10 lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[laisacreate@10 lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 2.7: Запуск программы

7) Я создала файл lab8-2.asm и написала нужный текст программы.



```
Открыть ▾  • lab8-2.asm
~/work/arch-pc/lab08

%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
```

Рис. 2.8: Текст программы lab8-2.asm

8) Я ввела несколько различных чисел для проверки программы.


```
35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
```

Рис. 2.12: Вторая строка

- 13) Данная строка находится на 47 месте. Она имеет адрес 00000163, ее машинный код - A1[00000000], исходный текст программы - `mov eax,[max]`; исходный код означает, что число, которое хранилось в переменной `max` записывается в регистр `eax`.

```
47 00000163 A1[00000000] mov eax,[max]
```

Рис. 2.13: Третья строка

- 14) В строке `mov ecx,[B]` я убрала `[B]` и попробовала создать файл. Выдало ошибку, так как для программы нужно два операнда.

```
[laisacrete@10 lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:41: error: invalid combination of opcode and operands
```

Рис. 2.14: Попытка создать файл с ошибкой


- 15) В файле листинга также показывается ошибка.

```
41 mov ecx ; иначе 'ecx = B'
41 ***** error: invalid combination of opcode and operands
```

Рис. 2.15: Ошибка в файле листинга

3 Самостоятельная работа

- 1) Я написала программу для нахождения меньшего из трех чисел. Для 15 варианта нужны следующие значения: 32, 6, 54. Программа вывела правильное значение - 6.

Открыть ▾ 

• lab8-3.asm
~/work/arch-pc/lab08

```
%include 'in_out.asm'

section .data
rem db 'Результат:',0h

section .bss
min resb 10
A RESB 20
B RESB 20
C RESB 20

section .text
global _start
_start:

mov eax,32
mov [A],eax
xor eax,eax

mov eax,6
mov [B],eax
xor eax,eax

mov eax,54
mov [C],eax
xor eax,eax

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

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

check_C:

mov ecx,[min]
cmp ecx,[C]
jl fin
```

Рис. 3.1: Текст программы

```
[laisacrete@10 lab08]$ nasm -f elf lab8-3.asm
[laisacrete@10 lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[laisacrete@10 lab08]$ ./lab8-3
Результат:6
```

Рис. 3.2: Результат работы программы

- 2) Я написала программу для вычисления значения выражения при введенных X и A. Мой вариант 15: $(a + 10, x < a \ x + 10, x \geq a)$

```
Открыть ▾ + lab8-4.asm
~/work/arch-pc/lab08

#include 'in_out.asm'

SECTION .data
X1 DB 'Введите значение X:',0
A1 DB 'Введите значение a:',0
otv DB 'Ответ: ',0

SECTION .bss
X RESB 20
A RESB 20
F RESB 20
SECTION .text
GLOBAL _start
_start:

mov eax,X1
call sprint

mov ecx,X
mov edx,10
call sread

mov eax,X
call atoi
mov [X],eax

mov eax,A1
call sprint

mov ecx,A
mov edx,10
call sread

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

mov ecx,A

cmp eax, [X]
```

Рис. 3.3: Текст программы

3) Результаты также оказались верными.

```
[laisacrete@10 lab08]$ ./lab8-4
Введите значение X:2
Введите значение a:3
Ответ: 13
[laisacrete@10 lab08]$ ./lab8-4
Введите значение X:4
Введите значение a:2
Ответ: 14
[laisacrete@10 lab08]$
```

Рис. 3.4: Результат работы программы

4 Выводы

Я изучила команды условного и безусловного перехода, а также научилась писать программы с переходами.

Список литературы