

# **Лабораторная работа №8**

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

Медникова Екатерина Михайловна

# Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выводы	12

## Список иллюстраций

3.1	Создание каталога и файла . . . . .	7
3.2	Ввод текста из листинга 8.1 . . . . .	7
3.3	Создание и запуск файла . . . . .	8
3.4	Изменение текста программы . . . . .	8
3.5	Создание и проверка работы файла . . . . .	8
3.6	Изменение текста программы . . . . .	9
3.7	Создание и проверка работы файла . . . . .	9
3.8	Создание файла lab8-2.asm . . . . .	9
3.9	Ввод программы из листинга 8.3 . . . . .	10
3.10	Создание исполняемого файла и проверка его работы . . . . .	10
3.11	Создание файла листинга . . . . .	10
3.12	push eax . . . . .	11
3.13	pop eax . . . . .	11
3.14	dec ecx . . . . .	11
3.15	Удаление max . . . . .	11
3.16	Выполнение трансляции и получение ошибки . . . . .	11

## 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 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление:

`jmp`

Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре.

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора.

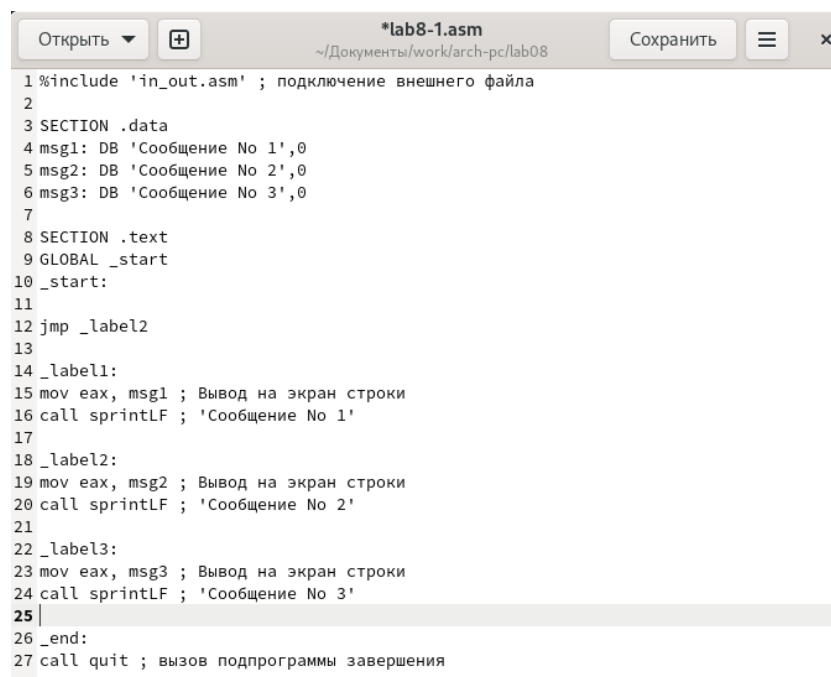
### 3 Выполнение лабораторной работы

1. Создала каталог для программ лабораторной работы No 8, перешла в него и создала файл lab8-1.asm:

```
[emmednikova@fedora arch-pc]$ mkdir lab08
[emmednikova@fedora arch-pc]$ cd lab08
[emmednikova@fedora lab08]$ touch lab8-1.asm
[emmednikova@fedora lab08]$
```

Рис. 3.1: Создание каталога и файла

2. Ввела в файл lab8-1.asm текст программы из листинга 8.1.



```
*lab8-1.asm
~/Документы/work/arch-pc/lab08
Сохранить

1 %include 'in_out.asm' ; подключение внешнего файла
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 sprintf ; 'Сообщение No 1'
17
18 _label2:
19 mov eax, msg2 ; Вывод на экран строки
20 call sprintf ; 'Сообщение No 2'
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call sprintf ; 'Сообщение No 3'
25 |
26 _end:
27 call quit ; вызов подпрограммы завершения
```

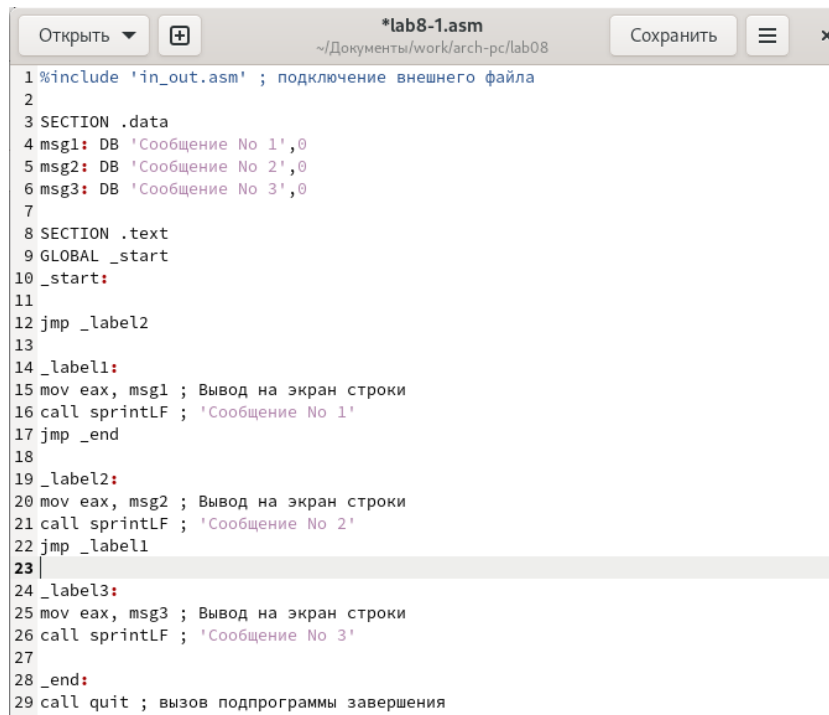
Рис. 3.2: Ввод текста из листинга 8.1

Создала исполняемый файл и запустила его.

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

Рис. 3.3: Создание и запуск файла

Изменила текст программы в соответствии с листингом 8.2.



```
*lab8-1.asm
~/Документы/work/arch-pc/lab08
Сохранить

1 %include 'in_out.asm' ; подключение внешнего файла
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 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
28 _end:
29 call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Изменение текста программы

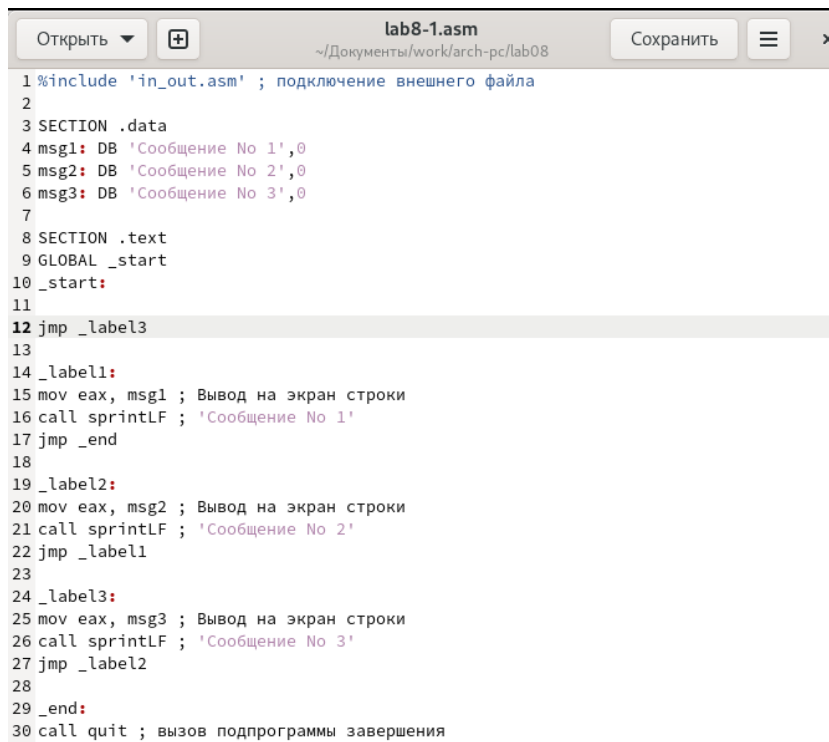
Создала исполняемый файл и проверила его работу.

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

Рис. 3.5: Создание и проверка работы файла



Изменила текст программы так, чтобы её вывод соответствовал инструкции, прописанной в лабораторной работе.

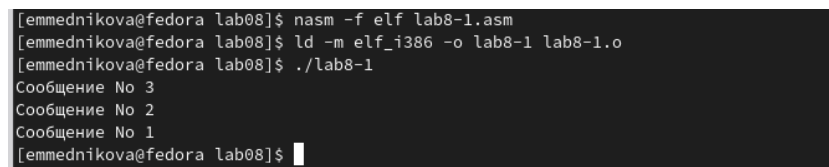


```
lab8-1.asm
~/Документы/work/arch-pc/lab08
Сохранить

1 %include 'in_out.asm' ; подключение внешнего файла
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 _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
```

Рис. 3.6: Изменение текста программы

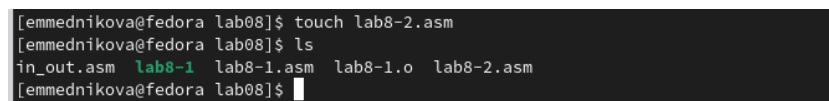
Создала исполняемый файл и проверила его работу.



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

Рис. 3.7: Создание и проверка работы файла

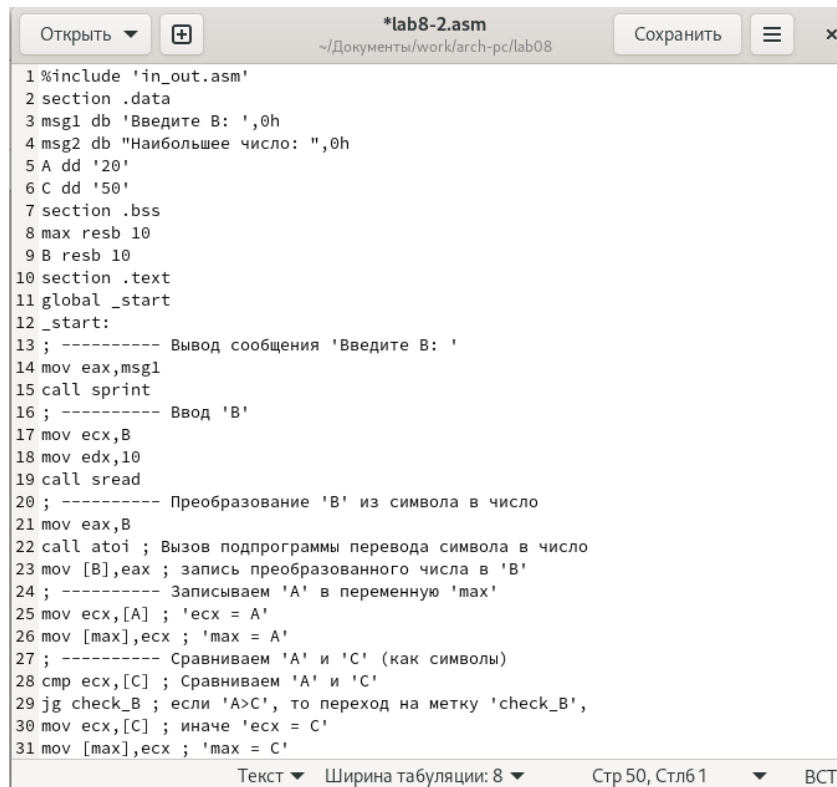
3. Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08.



```
[emmednikova@fedora lab08]$ touch lab8-2.asm
[emmednikova@fedora lab08]$ ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o  lab8-2.asm
[emmednikova@fedora lab08]$
```

Рис. 3.8: Создание файла lab8-2.asm

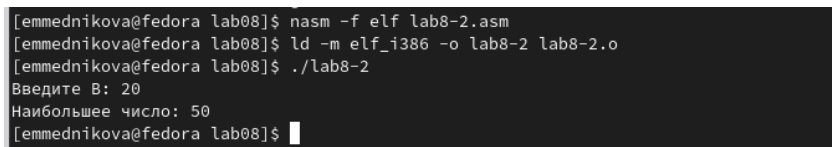
Ввела в созданный файл текст программы из листинга 8.3.



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

Рис. 3.9: Ввод программы из листинга 8.3

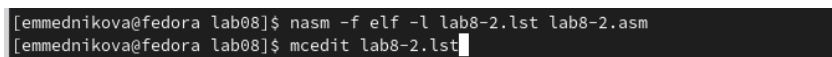
Создала исполняемый файл и проверила его работу.



```
[emmednikova@fedora lab08]$ nasm -f elf lab8-2.asm
[emmednikova@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[emmednikova@fedora lab08]$ ./lab8-2
Введите B: 20
Наибольшее число: 50
[emmednikova@fedora lab08]$
```

Рис. 3.10: Создание исполняемого файла и проверка его работы

4. Создала файл листинга для программы из файла lab8-2.asm. Открыла файл листинга lab8-2.lst с помощью текстового редактора mcedit.



```
[emmednikova@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
[emmednikova@fedora lab08]$ mcedit lab8-2.lst
```

Рис. 3.11: Создание файла листинга

27 строка. Адрес 00000012. Машинный код 50. Push eax (исходный текст программы) выделяет место наверху стека и помещает туда значение из регистра eax.

```
27 00000012 50      <1>      push      eax
```

Рис. 3.12: push eax

55 строка. Адрес 00000040. Машинный код 58. Pop eax (исходный текст программы) переносит любые данные из верхней части стека в eax и освобождает эту область памяти.

```
55 00000040 58      <1>      pop       eax
```

Рис. 3.13: pop eax

95 строка. Адрес 00000073. Машинный код 49. Dec ecx (исходный текст программы) уменьшает значение ecx на единицу.

```
95 00000073 49      <1>      dec       ecx.....
```

Рис. 3.14: dec ecx

Открыла файл с программой lab8-2.asm и удалила один операнд - max.

```
34 mov eax,max
```

Рис. 3.15: Удаление max

Выполнила трансляцию с получением файла листинга. Выдалась ошибка.

```
[emmednikova@fedora lab08]$ nasm -f elf lab8-2.asm
lab8-2.asm:34: error: invalid combination of opcode and operands
[emmednikova@fedora lab08]$
```

Рис. 3.16: Выполнение трансляции и получение ошибки

## 4 Выводы

Изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Познакомилась с назначением и структурой файла листинга.