

Лабораторная работа №8

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

Вершинина Ангелина Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файлы листинга	12
4.3	Задание для самостоятельной работы	14
5	Выводы	17
	Список литературы	18

Список иллюстраций

4.1	Создание каталога	8
4.2	Текст программы	9
4.3	Создание исполняемого файла и его запуск	9
4.4	Текст программы	10
4.5	Создание исполняемого файла и его запуск	10
4.6	Текст программы	11
4.7	Создание исполняемого файла и его запуск	11
4.8	Создание файла	12
4.9	Создание исполняемого файла и его запуск	12
4.10	Создание файла листинга	12
4.11	Открытие файла листинга	13
4.12	Строка 6	13
4.13	Строка 11	13
4.14	Строка 54	14
4.15	Удаление операнда	14
4.16	Трансляция файла	14
4.17	Текст программы	15
4.18	Создание исполняемого файла и его запуск	15
4.19	Создание исполняемого файла и его запуск	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 Задание

Написание программ, используя полученные знания

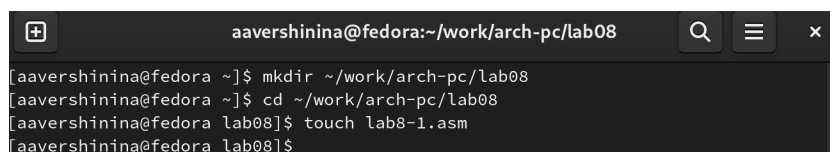
3 Теоретическое введение

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

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

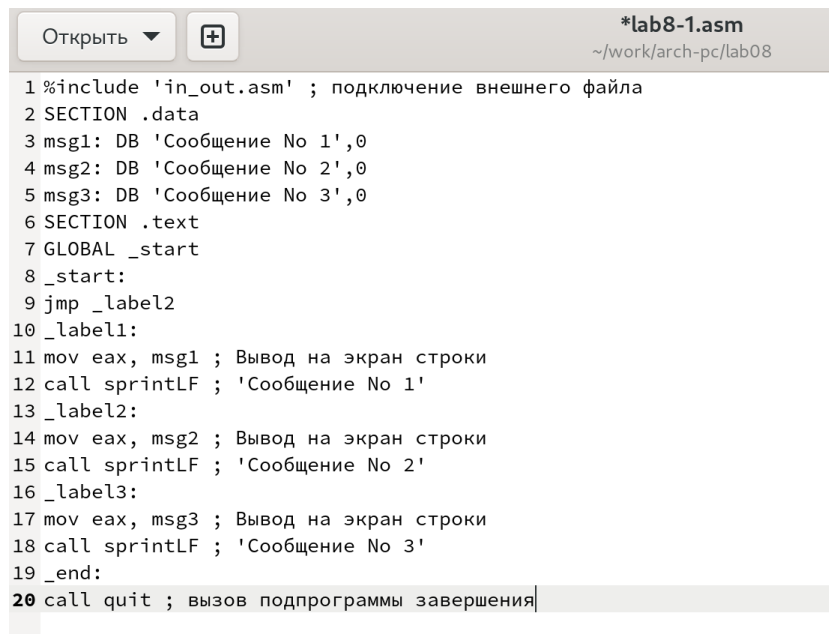
Создайте каталог для программ лабораторной работы No 8, перейдите в него и создайте файл lab8-1.asm (рис. 4.1)



```
aavershinina@fedora:~/work/arch-pc/lab08
[aavershinina@fedora ~]$ mkdir ~/work/arch-pc/lab08
[aavershinina@fedora ~]$ cd ~/work/arch-pc/lab08
[aavershinina@fedora lab08]$ touch lab8-1.asm
[aavershinina@fedora lab08]$
```

Рис. 4.1: Создание каталога

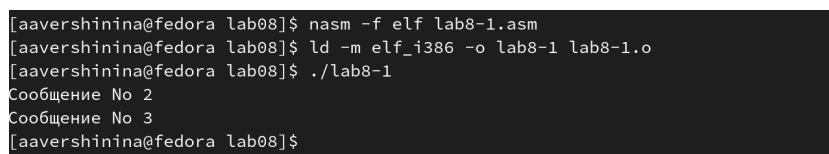
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введите в файл `lab8-1.asm` текст программы из листинга 8.1 (рис. 4.2)



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

Рис. 4.2: Текст программы

Создайте исполняемый файл и запустите его (рис. 4.3)

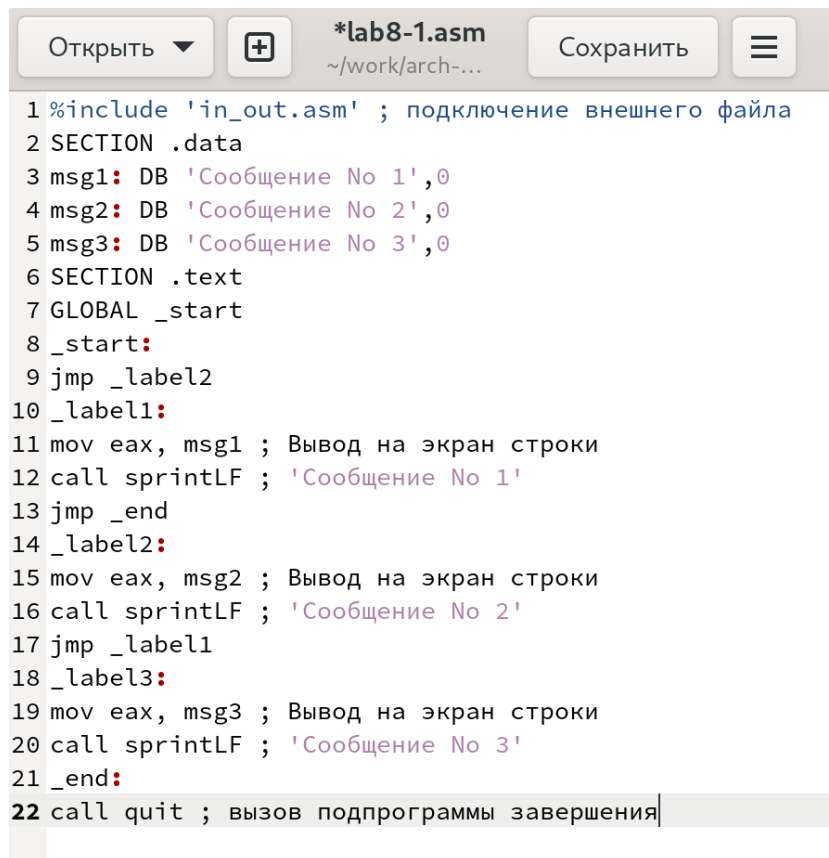


```
[aavershinina@fedora lab08]$ nasm -f elf lab8-1.asm
[aavershinina@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[aavershinina@fedora lab08]$ ./lab8-1
Сообщение No 2
Сообщение No 3
[aavershinina@fedora lab08]$
```

Рис. 4.3: Создание исполняемого файла и его запуск

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

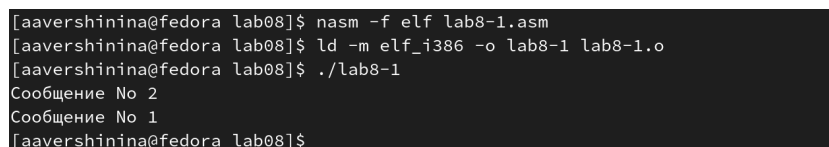
Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу. Для этого в текст программы после вывода сообщения No 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения No 1) и после вывода сообщения No 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом 8.2. (рис. 4.3)



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

Рис. 4.4: Текст программы

Создайте исполняемый файл и запустите его (рис. 4.5)



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

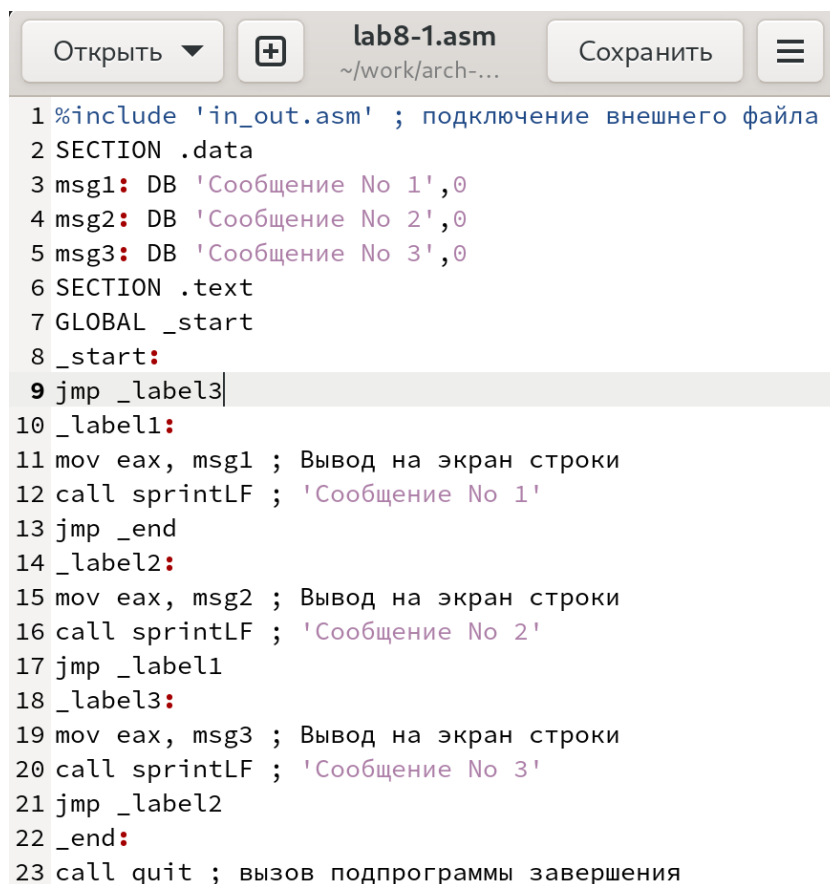
Рис. 4.5: Создание исполняемого файла и его запуск

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим:(рис. 4.6)

Сообщение No 3

Сообщение No 2

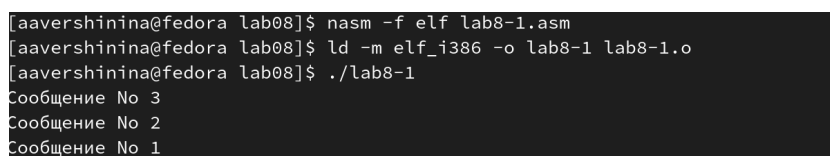
Сообщение No 1



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

Рис. 4.6: Текст программы

Создайте исполняемый файл и запустите его (рис. 4.7)



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

Рис. 4.7: Создание исполняемого файла и его запуск

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08. Внимательно изучите текст программы из листинга 8.3 и введите в lab8-2.asm. (рис. 4.8)

```
[aavershinina@fedora lab08]$ touch lab8-2.asm
```

Рис. 4.8: Создание файла

Создайте исполняемый файл и запустите его (рис. 4.9)

```
[aavershinina@fedora lab08]$ nasm -f elf lab8-2.asm
[aavershinina@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[aavershinina@fedora lab08]$ ./lab8-2
Введите B: 23
Наибольшее число: 50
```

Рис. 4.9: Создание исполняемого файла и его запуск

Обратите внимание, в данном примере переменные A и C сравниваются как символы, а переменная B и максимум из A и C как числа (для этого используется функция atoi преобразования символа в число). Это сделано для демонстрации того, как сравниваются данные. Данную программу можно упростить и сравнивать все 3 переменные как символы (т.е. не использовать функцию atoi). Однако если переменные преобразовать из символов числа, над ними можно корректно проводить арифметические операции.

4.2 Изучение структуры файлы листинга

Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создайте файл листинга для программы из файла lab8-2.asm (рис. 4.10)

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

Рис. 4.10: Создание файла листинга

Откройте файл листинга lab8-2.lst с помощью любого текстового редактора, например mcedit (рис. 4.11)

```

lab8-2.lst  [----]  0 L: [ 1+ 0 1/226] *(0 /14499b) 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>.
20                                     <1> ;----- sprint -----
21                                     <1> ; Функция печати сообщения
22                                     <1> ; входные данные: mov eax, <message>

```

Рис. 4.11: Открытие файла листинга

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

1. 6(номер строки) 00000001(адрес в сегменте кода) 89C3(машинный код) mov ebx,eax(исходный текст программы) (рис. 4.12)
2. 11(номер строки) 00000008(адрес в сегменте кода) 40(машинный код) inc eax(исходный текст программы) (рис. 4.13)
3. 54(номер строки) 0000003B(адрес в сегменте кода) E8CFFFFFFF(машинный код) call sprint(исходный текст программы) (рис. 4.14)

```

6 00000001 89C3                       <1> mov     ebx, eax.....

```

Рис. 4.12: Строка 6

```

11 00000008 40                        <1> inc     eax.....

```

Рис. 4.13: Строка 11

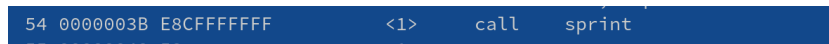


Рис. 4.14: Строка 54

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалите один операнд.(рис. 4.15) Выполните трансляцию с получением файла листинга (рис. 4.16). В результате система выдает ошибку и создает файлы lab8-2 и lab8-2.lst.

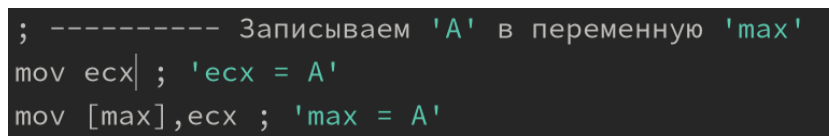


Рис. 4.15: Удаление операнда

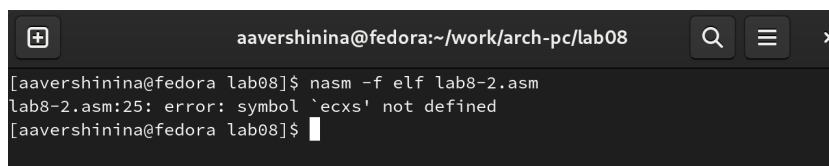


Рис. 4.16: Трансляция файла

4.3 Задание для самостоятельной работы

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных а, b и с. Значения переменных выбрать из табл. 8.5 в соответствии с вариантом 12, полученным при выполнении лабораторной работы No 7. (рис. 4.17) Создайте исполняемый файл и проверьте его работу. (рис. 4.18)

```

6 C dd '26'
7 section .bss
8 min 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 'min'
25 mov ecx, [A] ; 'ecx = A'
26 mov [min],ecx ; 'min = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'

```

Рис. 4.17: Текст программы

```

[aavershinina@fedora lab08]$ nasm -f elf sr1.asm
[aavershinina@fedora lab08]$ ld -m elf_i386 -o sr1 sr1.o
[aavershinina@fedora lab08]$ ./sr1
Введите B: 29
Наименьшее число: 26

```

Рис. 4.18: Создание исполняемого файла и его запуск

2. Напишите программу, которая для введенных с клавиатуры значений x и y вычисляет значение заданной функции $f(x, y)$ и выводит результат вычислений. Вид функции $f(x, y)$ выбрать из таблицы 8.6 вариантов заданий в соответствии с вариантом 12, полученным при выполнении лабораторной работы No 7. Создайте исполняемый файл и проверьте его работу для значений x и y из 8.6. (рис. 4.19)

```
[aavershinina@fedora lab08]$ nasm -f elf sr2.asm
[aavershinina@fedora lab08]$ ld -m elf_i386 -o sr2 sr2.o
[aavershinina@fedora lab08]$ ./sr2
Введите x: 3
Введите a: 7
F(x): 21
[aavershinina@fedora lab08]$ ./sr2
Введите x: 6
Введите a: 4
F(x): 1
```

Рис. 4.19: Создание исполняемого файла и его запуск

5 Выводы

В результате выполнения лабораторной работы я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Ознакомилась с назначением и структурой файла листинга.

Список литературы