

Отчёт по лабораторной работе №7

Простейший вариант

Сахно Алёна Юрьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Задание для самостоятельной работы	15
6	Выводы	17
	Список литературы	18

Список иллюстраций

4.1	Сохдание каталога	8
4.2	Программа с использованием инструкции jmp	9
4.3	Создание исполняемого файла	9
4.4	Программа с использованием инструкции jmp	10
4.5	Создание исполняемого файла	11
4.6	Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C.	12
4.7	Вот какие файлы создаются в этом случае	13
4.8	Вот какие файлы создаются в этом случае	14
5.1	Результат для 1 задания	15
5.2	Результат для 2 задания	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

1. Теоретическое введение
2. Выполнение Лабораторной работы
3. Самостоятельная работа
4. Вывод

3 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.

Команды безусловного перехода

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

В следующем примере рассмотрим использование инструкции `jmp`:

`label:`

`... ; ... ; команды ... ; jmp label`

Команды условного перехода

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

4 Выполнение лабораторной работы

Реализация переходов в NASM

1. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm:

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

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

```
touch lab7-1.asm
```

(рис.1 4.1).

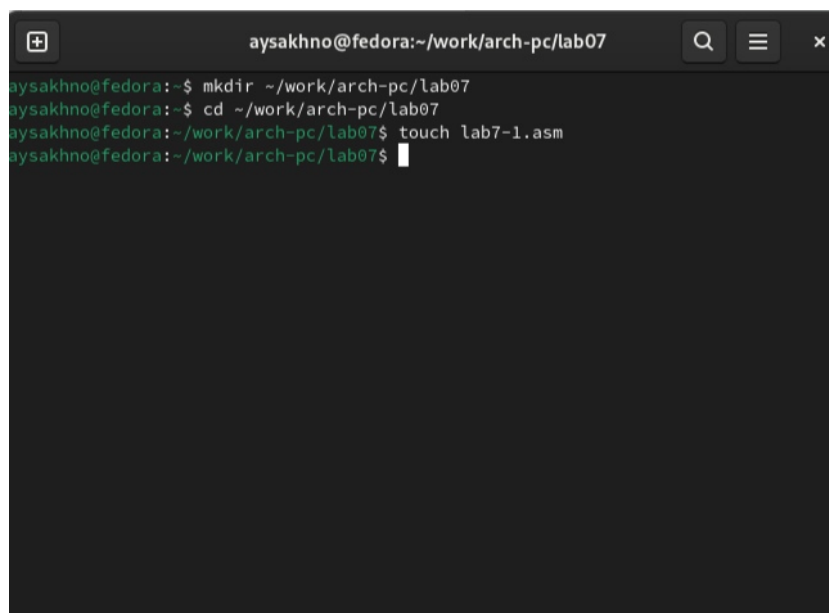
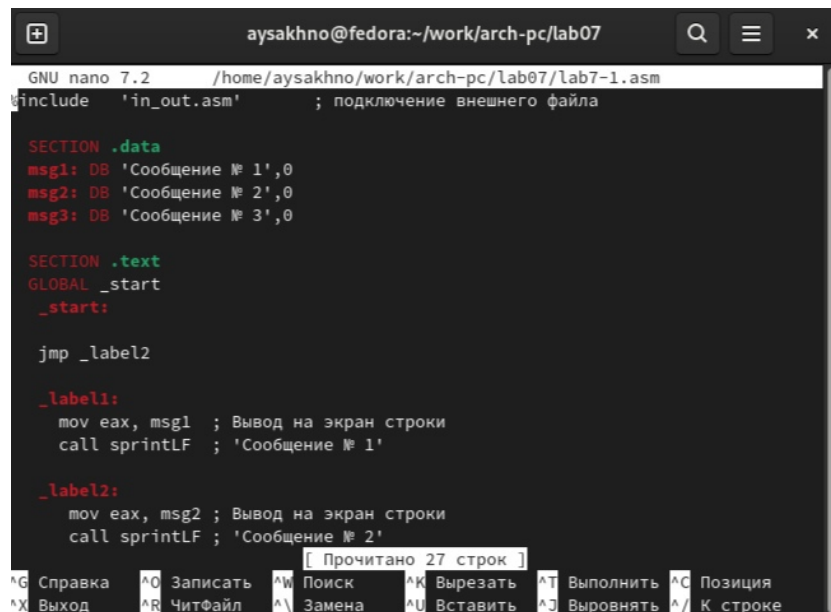


Рис. 4.1: Сохдание каталога

2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`.

Введите в файл lab7-1.asm текст программы из листинга

(рис.2 4.2).



```
GNU nano 7.2 /home/aysakhno/work/arch-pc/lab07/lab7-1.asm
#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'

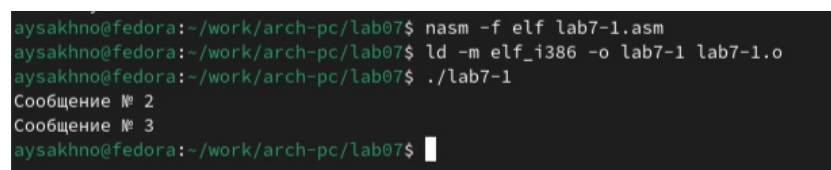
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
```

Рис. 4.2: Программа с использованием инструкции jmp

Создайте исполняемый файл и запустите его. Результат работы данной программы будет следующим:

```
user@dk4n31:~$ ./lab7-1
Сообщение № 2
Сообщение № 3
user@dk4n31:~$
```

(рис.3 4.3).



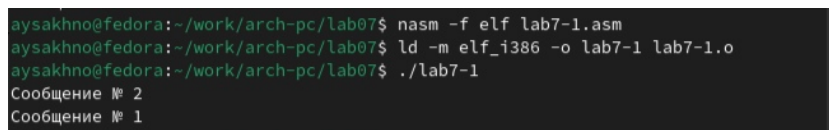
```
aysakhno@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aysakhno@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aysakhno@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
aysakhno@fedora:~/work/arch-pc/lab07$
```

Рис. 4.3: Создание исполняемого файла

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 № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом

(рис.4 4.4).



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

Рис. 4.4: Программа с использованием инструкции `jmp`

Создайте исполняемый файл и проверьте его работу. Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим:

```
user@dk4n31:~$ ./lab7-1
```

```
Сообщение № 3 Сообщение № 2 Сообщение № 1
```

```
user@dk4n31:~$
```

(рис.5 4.5).

```

GNU nano 7.2 /home/aysakhno/work/arch-pc/lab07/lab7-2.asm Изменён
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
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'

```

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

Рис. 4.5: Создание исполняемого файла

- 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: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры. Создайте файл `lab7-2.asm` в каталоге `~/work/arch-pc/lab07`. Внимательно

изучите текст программы из листинга 7.3 и введите в lab7-2.asm.

(рис.6 4.6).

```
aysakhno@fedora:~/work/arch-pc/lab07$ touch ~/work/arch-pc/lab06/lab7-2.asm
aysakhno@fedora:~/work/arch-pc/lab07$ mc
aysakhno@fedora:~/work/arch-pc/lab07$ mc
aysakhno@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aysakhno@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aysakhno@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
aysakhno@fedora:~/work/arch-pc/lab07$
```

Рис. 4.6: Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C.

Создайте исполняемый файл и проверьте его работу для разных значений B. 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.

Изучение структуры файлы листинга

4. 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 `lab7-2.asm`

```
nasm -f elf -l lab7-2.lst lab7-2.asm
```

Откройте файл листинга `lab7-2.lst` с помощью любого текстового редактора, например `mcedit`:

```
mcedit lab7-2.lst
```

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору. Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга:

```
nasm -f elf -l lab7-2.lst lab7-2.asm
```

Какие выходные файлы создаются в этом случае? Что добавляется в листинге? (рис.7 4.7).

```

lab7-2.lst [B---] 79 L:[191+ 6 197/224] *(12185/14432b) 1086 0x43E[*][X]
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из симво
21 00000101 E896FFFFFF call atoi
22 00000106 A3[0A000000] mov [B],eax ; запись преобразованного
23 ; ----- Записываем 'A' в переменную
24 00000108 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
25 00000111 890D[00000000] mov [max],ecx ; 'max = A'
26 ; ----- Сравниваем 'A' и 'C' (как с
27 00000117 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
28 0000011D 7F0C jg check_B ; если 'A>C', то переход н
29 0000011F 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
30 00000125 890D[00000000] mov [max],ecx ; 'max = C'
31 ; ----- Преобразование 'max(A,C)' и
32 check_B:
33 0000012B B8[00000000] mov eax,max
34 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перево
35 00000135 A3[00000000] mov [max],eax ; запись преобразованно
36 ; ----- Сравниваем 'max(A,C)' и 'B'
37 0000013A 8B0D[00000000] mov ecx,[max]
38 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и
39 00000146 7F0C jg fin ; если 'max(A,C)>B', то перехо
40 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
41 0000014E 890D[00000000] mov [max],ecx
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход

```

Рис. 4.7: Вот какие файлы создаются в этом случае

(рис.8 4.8).

```

aysakhno@fedora:~/work/arch-pc/lab07
ab7-2.lst [B---] 79 L:[185+13 198/225] *(12238/14485b) 1086 0x43E[*][X]
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из симво
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного
24 ; ----- Записываем 'A' в переменную
25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как с
28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C jg check_B ; если 'A>C', то переход н
30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' и
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перево
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC 10Выход

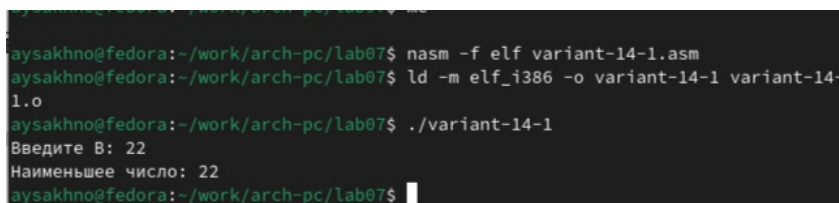
```

Рис. 4.8: Вот какие файлы создаются в этом случае

5 Задание для самостоятельной работы

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a , b и c . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

(рис.9 5.1).

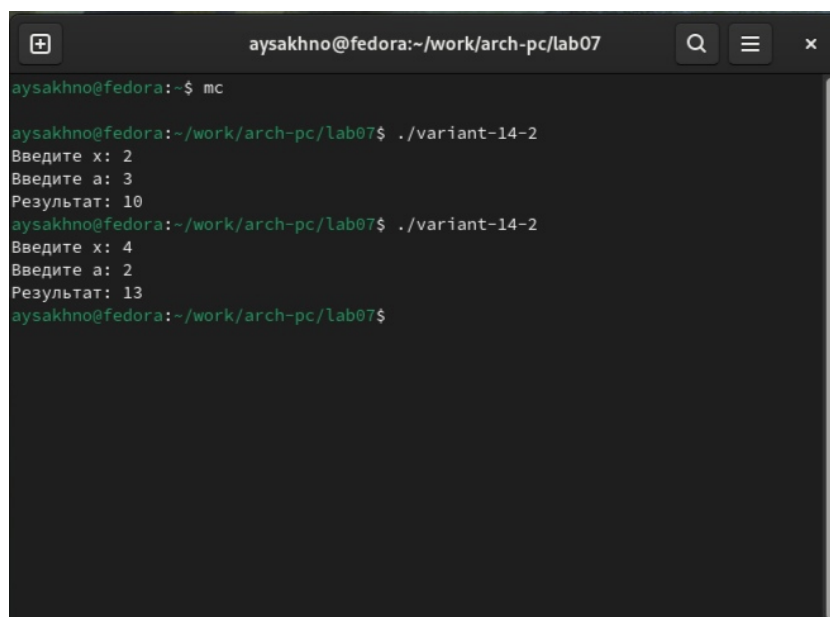


```
aysakhno@fedora:~/work/arch-pc/lab07$ nasm -f elf variant-14-1.asm
aysakhno@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o variant-14-1 variant-14-1.o
aysakhno@fedora:~/work/arch-pc/lab07$ ./variant-14-1
Введите B: 22
Наименьшее число: 22
aysakhno@fedora:~/work/arch-pc/lab07$
```

Рис. 5.1: Результат для 1 задания

2. Напишите программу, которая для введенных с клавиатуры значений a и b вычисляет значение заданной функции $f(a, b)$ и выводит результат вычислений. Вид функции $f(a, b)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений a и b из 7.6

(рис.10 5.2).



The image shows a terminal window with a dark background. The title bar at the top reads 'aysakhno@fedora:~/work/arch-pc/lab07'. The terminal content shows the following sequence of commands and outputs:

```
aysakhno@fedora:~$ mc
aysakhno@fedora:~/work/arch-pc/lab07$ ./variant-14-2
Введите x: 2
Введите a: 3
Результат: 10
aysakhno@fedora:~/work/arch-pc/lab07$ ./variant-14-2
Введите x: 4
Введите a: 2
Результат: 13
aysakhno@fedora:~/work/arch-pc/lab07$
```

Рис. 5.2: Результат для 2 задания

6 Выводы

Я изучила команды условного и безусловного переходов. Приобрела навыков написания программ с использованием переходов. Познакомилась с назначением и структурой файла листинга.

Список литературы

:::{#refs}:::https://esystem.rudn.ru/pluginfile.php/2089087/mod_resource/content/0