Шаблон отчёта по лабораторной работе 7

Простейший вариант

Абдуллахи Бахара

Содержание

Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

Задание

7.3. Порядок выполнения лабораторной работы

7.3.1. Реализация переходов в NASM

1- Создайте каталог для программам лабораторной работы № 7, перейдите в него и со- здайте файл lab7-1.asm:

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~$ cd ~/work/
arch-pc/
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c$ mkdir lab07
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c$ ls
lab04  lab05  lab06  lab07
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c$ cd lab07
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ touch lab7-1.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ls
lab7-1.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$
```

2- Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмот- рим пример программы с использованием инструкции jmp. Введите в файл lab7-1.asm текст программы из листинга 7.1.

```
%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 sprintLF       ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2       ; Вывод на экран строки
call sprintLF       ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3       ; Вывод на экран строки
call sprintLF       ; 'Сообщение № 3'
jmp _label2

_end:
call quit           ; вызов подпрограммы завершения
```

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

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ nasm -f elf lab7-1.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$
```

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 7.2.

```
SECTION .text
GLOBAL _start
_start:

jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'

_end:
call quit     ; вызов подпрограммы завершения

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ nasm -f elf lab7-1.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$
```

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:~$

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ nasm -f elf lab7-1.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$
```

3- Использование инструкции jmp приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход дол- жен 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 вводиться 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.

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ touch lab7-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$
```

```
/home/bahara123-virtualbox/work/arch-pc/lab07/lab7-2.asm
%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

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify
```

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 преобра- зования 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.

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ nasm -f elf lab7-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$
```

7.3.2. 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

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst
lab7-1      lab7-1.o    lab7-2.asm  lab7-2.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$
```

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

```
/home/wa~7-2.lst   [----] 43 L:[  1+ 0   1/226] *(43  /14545b) 0099 0x063 [*][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>
   23                                <1> sprint:
   24 0000000F 52                    <1>     push    edx
   25 00000010 51                    <1>     push    ecx
1Help  2Save  3Mark  4Replac 5Copy  6Move  7Search 8Delete 9PullDn10Quit
```

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержи- мое 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:

```
/home/bahara123-virtualbox/work/arch-pc/lab07/lab7-2.asm
%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

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify
```

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:24: error: invalid combination of opcode and operands
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-p
c/lab07$
```

7.4. 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.

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab07$ touch lab7-hw.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab07$ mc
```

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-hw.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-hw lab7-hw.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab07$ ./lab7-hw
Введите B: 6
наименьшей число: 6
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab07$
```

```
%include 'in_out.asm'
section        .data
  msg1 db 'Введите B: ',0h
  msg2 db "наименьшей число: ",0h
  A dd '32'
  C dd '54'
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 ; 'min = A'
; ----------  Сравниваем 'A' и 'C' (как символы)
   cmp  [C],ecx   ; 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:
```

#

Теоретическое введение

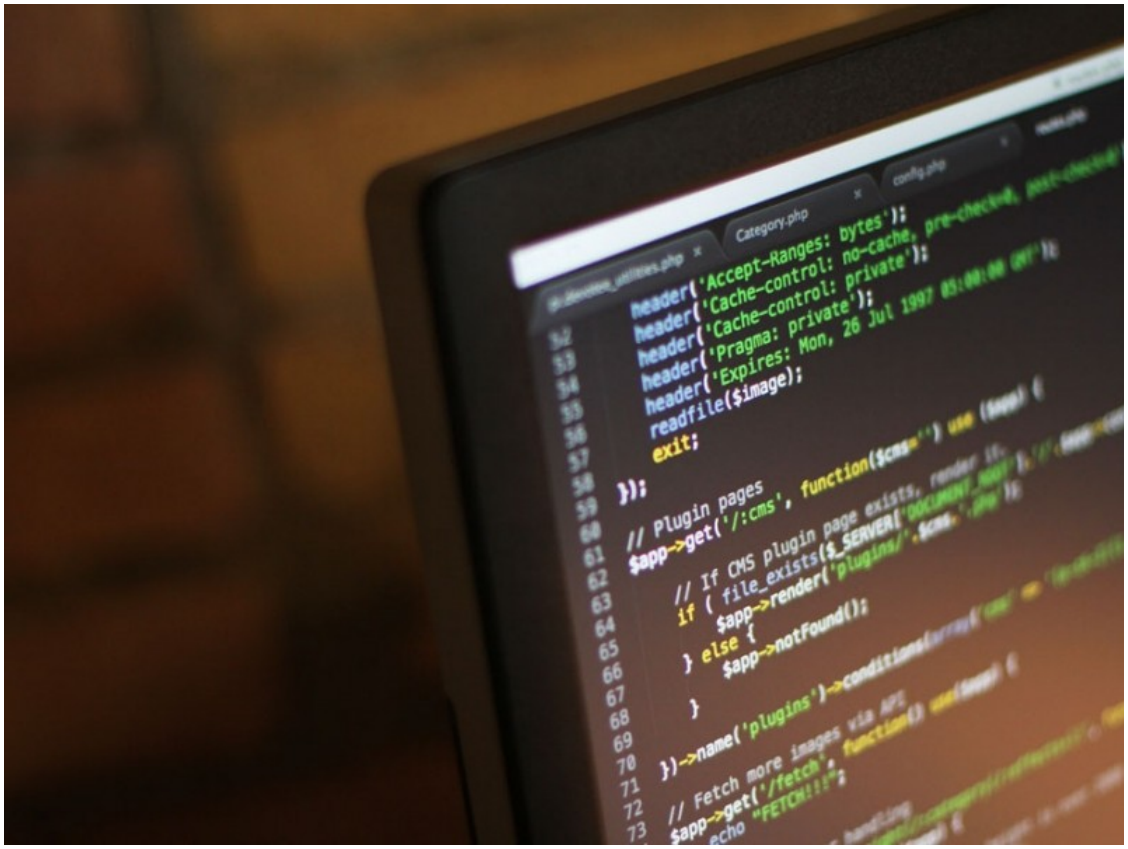
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

• условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.

• безусловный переход – выполнение передачи управления в определенную точку про- граммы без каких-либо условий.

Выполнение лабораторной работы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. @fig:001).



Название рисунка

Выводы

Здесь кратко описываются итоги проделанной работы.

Список литературы