Отчёт по лабораторной работе №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений.

Попутников Егор Сергеевич

Содержание

Список иллюстраций

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. 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.
2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) 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 x и a из 7.6.

3 Выполнение лабораторной работы

1. Создадим каталог для программ лабораторной работы № 7, перейдем в него и со- здадим файл lab7-1.asm:(3.1)

```
egor@espoputnikov-dk3n56:~$ mkdir ~/work/arch-pc/lab07
egor@espoputnikov-dk3n56:~$ cd  ~/work/arch-pc/lab07
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ touch lab7-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$
```

Рис. 3.1: Создание файла

2. Инструкция jmp в NASM используется для реализации безусловных перехо- дов. Рассмот- рим пример программы с использованием инструкции jmp. Введем в файл lab7-1.asm текст программы из листинга 7.1.(3.2)

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 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 sprintLF ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintLF ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintLF ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Файл листинга 7.1.

3. Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим:(3.3)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 3.3: 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). Изменим текст программы в соответствии с листингом 7.2.(3.4)

```
%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 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
```

Рис. 3.4: Текст листинга 7.2.

4. Создадим исполняемый файл и проверим его работу.(3.5)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$
```

Рис. 3.5: Создание исполняемого файла

Изменим текст программы добавив или изменив инструкции jmp, чтобы вывод програм- мы был следующим: Сообщение № 3 Сообщение № 2 Сообщение № 1(3.6)

```
%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
```

Рис. 3.6: Измененный текст листинга

5. Создадим исполняемый файл и проверим его работу.(3.7)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ gedit lab7-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 3.7: Создание исполняемого файла

6. Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучим текст программы из листинга 7.3 и введем в lab7-2.asm.(3.8)

```
%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`
```

Рис. 3.8: Текст листинга 7.3.

Создадим исполняемый файл и проверим его работу для разных значений B.(3.9)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 40
Наибольшее число: 50
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
```

Рис. 3.9: Создание исполняемого файла

7. Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создадим файл листинга для программы из файла lab7-2.asm(3.10)

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

Рис. 3.10: Создание файла листинга

Откроем файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit:(3.11)

```
   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
  26 00000011 53                    <1>     push    ebx
  27 00000012 50                    <1>     push    eax
  28 00000013 E8E8FFFFFF            <1>     call    slen
```

Рис. 3.11: Файл листинга

Удалим в файле листинга один операнд:(3.12)

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ gedit lab7-2.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:42: error: invalid combination of opcode and operands
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$
```

```
                                mov [max]
      ******************         error: invalid combination of opcode and operands
                                ; ---------- Вывод результата
                                fin:
```

Рис. 3.12: Ошибка компиляции файла листинга

4 Выполнение самостоятельной работы

(Вариант №4) 1. Напишем программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выберем из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создадим исполняемый файл и проверим его работу.(4.1)

```
%include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
A dd '88'
B dd '8'
C dd '68'
section .bss
min resb 10
section .text
global _start
_start:
; ---------- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ---------- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ---------- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в `max`
; ---------- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C)<B', то переход на 'fin',
```

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ gedit lab7-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 8
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$
```

Рис. 4.1: Задание №1

2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной ра-

боты № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.(4.2)

```
%include 'in_out.asm'
SECTION .data
msg1: DB 'Введите x: ',0
msg2: DB 'Введите a: ',0
msg3: DB 'f(x) = ',0
section .bss
x resb 10
a resb 10
f resb 10
SECTION .text
GLOBAL _start
_start:
;Вывод сообщения и ввод x
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
;Вывод сообщения и ввод a
mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
;преобразоание x из сивола в число
mov eax,x
call atoi
mov [x],eax
;преобразование a из символа в число
mov eax,a
call atoi
mov [a],eax

mov eax,[x]
mov ecx,2
mul ecx ;eax=2x
```

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 0
f(x) = 7
```

Рис. 4.2: Задание №2

5 Выводы

Я изучил команды условного и безусловного переходов. Приобрел навыки написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.