Отчет по лабораторной работе №8

Смирнов Дмитрий Романович

Содержание

1 Цель работы

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

2 Выполнение лабораторной работы

Создам каталог для программ лабораторной работы № 8, перейду в него и создам файл lab8-1.asm.

```
[smirnovd_03_22@10 ~]$ mkdir work/arch-pc/lab08
[smirnovd_03_22@10 ~]$ cd work/arch-pc/lab08
[smirnovd_03_22@10 lab08]$ touch lab8-1.asm
[smirnovd_03_22@10 lab08]$
```

Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрю пример программы с использованием инструкции jmp.

```
%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'

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

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

   _end:
     call quit ; вызов подпрограммы завершения
```

Рис. 1: Рис2

Результат:

```
[smirnovd_03_22@10 lab08]$ nasm -f elf lab8-1.asm
[smirnovd_03_22@10 lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[smirnovd_03_22@10 lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[smirnovd_03_22@10 lab08]$
```

Рис. 2: Рис3

Инструкция jmp позволяет осуществлять переходы не только вперед но и назад. Изменю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу

```
%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: Рис4

Результат:

```
[smirnovd_03_22@10 lab08]$ nasm -f elf lab8-2.asm
[smirnovd_03_22@10 lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[smirnovd_03_22@10 lab08]$ ./lab8-2
Сообщение № 2
Сообщение № 1
[smirnovd_03_22@10 lab08]$
```

Рис. 4: Рис5

Изменю текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим: user@dk4n31:~$ ./lab8-1 Сообщение № 3 Сообщение № 2 Сообщение № 1 user@dk4n31:~$

```
%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:


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

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

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

   _end:
     call quit ; вызов подпрограммы завершения
```

Рис. 5: Рис6

```
[smirnovd_03_22@10 lab08]$ nasm -f elf lab8-2,1.asm
[smirnovd_03_22@10 lab08]$ ld -m elf_i386 -o lab8-2,1 lab8-2,1.o
[smirnovd_03_22@10 lab08]$ ./lab8-2,1
Сообщение № 1
Сообщение № 2
Сообщение № 3
[smirnovd_03_22@10 lab08]$
```

Рис. 6: Рис7

Создам файл lab8-3.asm. Введу текст программы из листинга 8.3

```
%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`
; ---------- 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: '
```

Рис. 7: Рис8

Результат:

```
[smirnovd_03_22@10 lab08]$ nasm -f elf lab8-3.asm
[smirnovd_03_22@10 lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[smirnovd_03_22@10 lab08]$ ./lab8-3
Введите B: 45
Наибольшее число: 50
[smirnovd_03_22@10 lab08]$
```

Рис. 8: Рис9

Создам файл lab8-3.lst

```
[smirnovd_03_22@10 lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
[smirnovd_03_22@10 lab08]$
```

Рис. 9: Рис10

Объясним содержимое трёх строк файла.

```
33 0000001B 89C1            <1>     mov     ecx, eax
34 0000001D BB01000000      <1>     mov     ebx, 1
35 00000022 B804000000      <1>     mov     eax, 4
```

Рис. 10: Рис11

1) 33 - номер строки файла листинга, 0000001B - смещение машинного кода от начала текущего сегмента, 89C1 - машинный код, в который ассеблируется данная инструкция в виде шестнадцатиричной последовательности, mov ecx,eax - исходная строка программы.
2) 34 - номер строка файла листинга, 0000001D - смещение машинного кода от начала текущего сегмента, BB01000000 - машинный код, в который ассеблируется данная инструкция в виде шестнадцатиричной последовательности, mov ebx,1 - исходная строка программы.
3) 35 - номер строка файла листинга, 00000022 - смещение машинного кода от начала текущего сегмента, B804000000 - машинный код, в который ассеблируется данная инструкция в виде шестнадцатиричной последовательности, mov eax,4 - исходная строка программы. Если удалю один операнд в программе и выполню трансляцию с получением файла. В файле листинга также выдаст ошибку.

3 Задания для самостоятельной работы:

№1

Открыть

la
~/wor

```
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '81'
C dd '72'
section .bss
min resb 10
B resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [min],ecx

cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [min],ecx

check_B:
mov eax,min
call atoi
mov [min],eax

mov ecx,[min]
cmp ecx,[B]
jb fin
mov ecx,[B]
mov [min],ecx

fin:
mov eax, msg2
```

Рис. 11: Рис12

```
[smirnovd_03_22@10 lab08]$ nasm -f elf lab8-4.asm
[smirnovd_03_22@10 lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[smirnovd_03_22@10 lab08]$  ./lab8-4
Введите B: 22
Наименьшее число: 22
[smirnovd_03_22@10 lab08]$
```

Рис. 12: Рис13

№2

Открыть

```
%include 'in_out.asm'
section .data
msg1 db 'Введите x: ',0h
msg2 db 'Введите a: ',0h
msg3 db 'Функция равна: ',0h

section .bss
x: resb 10
a: resb 10
answer: resb 10

section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,x
mov edx,10
call sread

mov eax,x
call atoi
mov [x],eax

mov eax,msg2
call sprint

mov ecx,a
mov edx,10
call sread

mov eax,x
call atoi
mov [a],eax

mov eax, [x]

cmp eax,0

je Func1
jne Func2

Func1:
mov eax, 3
mov ebx, [a]
```

Рис. 13: Рис14

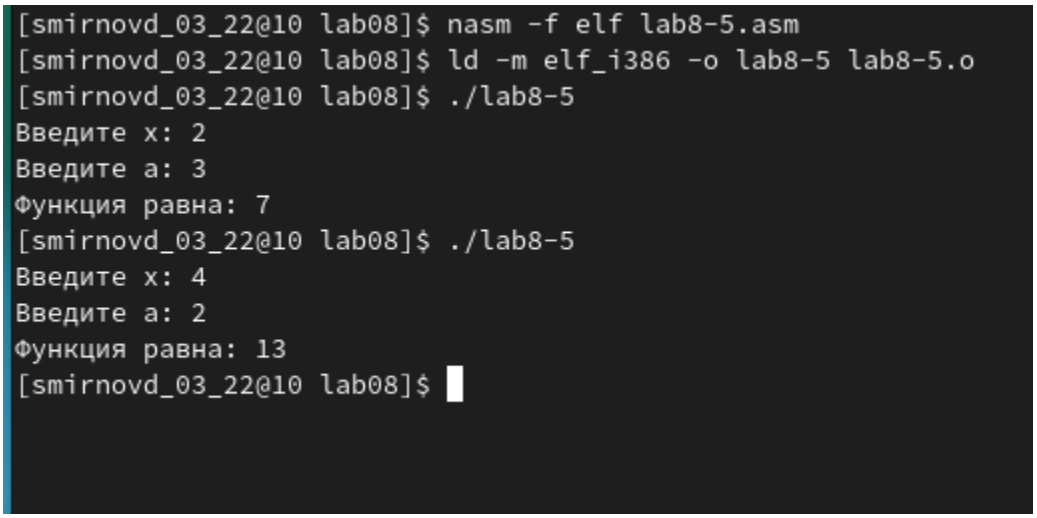

Рис. 14: Рис15

4 Выводы

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