Отчёт по лабораторной работе №7

Команды безусловного и условного перехода в NASM. Программирование ветвлений.

Скобеева Алиса Алексеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить условный и безусловный переход. Ознакомиться с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Создаем каталог для программ лабораторной работы №7, переходим в него и создаем файл lab7-1.asm:

```
aaskobeeva@fedora:~$ mkdir ~/work/arch-pc/lab07
aaskobeeva@fedora:~$ cd ~/work/arch-pc/lab07
aaskobeeva@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
aaskobeeva@fedora:~/work/arch-pc/lab07$
```

Рис. 2.1: Используем команды mkdir и touch

Вводим в файл текст программы из листинга 7.1:

```
%include 'in_out.asm'
    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
       call sprintLF
     _label2:
       mov eax, msg2
       call sprintLF
     _label3:
       mov eax, msg3
       call sprintLF
     _end:
       call quit
```

Рис. 2.2: Вводим текст

Создаем исполняемый файл и запускаем его:

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

Рис. 2.3: Запускаем файл и смотрим на его работу

Открываем файл для редактирования и изменяем его в соответствии с листингом 7.2:

```
%include 'in_out.asm'
    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
       call sprintLF
       jmp _end
     _label2:
       mov eax, msg2
       call sprintLF
       jmp _label1
     _label3:
       mov eax, msg3
       call sprintLF
     _end:
       call quit
```

Рис. 2.4: Редактируем файл

Создаем исполняемый файл и запускаем его:

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

Рис. 2.5: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его, чтобы произошел данный вывод:

```
%include 'in_out.asm'
    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
       call sprintLF
       jmp _end
     _label2:
       mov eax, msg2
       call sprintLF
       jmp _label1
     _label3:
       mov eax, msg3
       call sprintLF
       jmp _label2
     _end:
       call quit
```

Рис. 2.6: Редактируем файл

Создаем исполняемый файл и запускаем его:

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

Рис. 2.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

Создаем новый файл, открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3:

```
%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:
   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  [max],ecx
check_B:
   mov  eax,max
   call atoi
   mov  [max],eax
   mov  ecx,[max]
   cmp  ecx,[B]
   jg   fin
   mov  ecx,[B]
   mov  [max],ecx
fin:
   mov eax, msg2
   call sprint
   mov eax,[max]
   call iprintLF
   call quit
```

Рис. 2.8: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения B:

```
aaskobeeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aaskobeeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aaskobeeva@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 20
aaskobeeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aaskobeeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aaskobeeva@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 20
aaskobeeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aaskobeeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aaskobeeva@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 20
aaskobeeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aaskobeeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aaskobeeva@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 20
```

Рис. 2.9: Смотрим, как сработали программы

2.2 Изучение структуры файла листинга

Создаем файл листинга для программы lab7-2.asm:

```
aaskobeeva@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
aaskobeeva@fedora:~/work/arch-pc/lab07$
```

Рис. 2.10: Создаем файл листинга

Открываем файл листинга и изучаем его:

```
1                                       %include 'in_out.asm'
1                                   <1> ;---------------   slen  --------------------
2                                   <1> ; Функция вычисления длины сообщения
3                                   <1> slen:
4 00000000 53                       <1>     push    ebx
5 00000001 89C3                     <1>     mov     ebx, eax
6                                   <1>
7                                   <1> nextchar:
8 00000003 803800                   <1>     cmp     byte [eax], 0
9 00000006 7403                     <1>     jz      finished
10 00000008 40                      <1>     inc     eax
11 00000009 EBF8                    <1>     jmp     nextchar
12                                  <1>
13                                  <1> finished:
14 0000000B 29D8                    <1>     sub     eax, ebx
15 0000000D 5B                      <1>     pop     ebx
16 0000000E C3                      <1>     ret
17                                  <1>
18                                  <1>
19                                  <1> ;---------------  sprint  --------------------
20                                  <1> ; Функция печати сообщения
21                                  <1> ; входные данные: mov eax,<message>
22                                  <1> sprint:
23 0000000F 52                      <1>     push    edx
24 00000010 51                      <1>     push    ecx
25 00000011 53                      <1>     push    ebx
26 00000012 50                      <1>     push    eax
27 00000013 E8E8FFFFFF              <1>     call    slen
28                                  <1>
29 00000018 89C2                    <1>     mov     edx, eax
30 0000001A 58                      <1>     pop     eax
```

Рис. 2.11: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, BB01000000-машинный код, mov ebx,1-присвоение переменной ecx значения 1. Строка 34: 00000022-адрес в сегменте кода, B804000000-машинный код, mov eax,4-присвоение eax значения 4. Строка 35: 00000027-адрес в сегменте кода, CD-машинный код, int 80h-вызов ядра.

Открываем файл и удаляем один операндум:

```
%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:
   mov eax,msg1
   call sprint
   mov ecx,B
   mov edx
   call sread
   mov eax,B
   call atoi
   mov  [B],eax
   mov  ecx,[A]
   mov  [max],ecx
check_B:
   mov  eax,max
   call atoi
   mov  [max],eax
   mov  ecx,[max]
   cmp  ecx,[B]
   jg   fin
   mov  ecx,[B]
   mov  [max],ecx
fin:
   mov eax, msg2
   call sprint
   mov eax,[max]
   call iprintLF
   call quit
```

Рис. 2.12: Удаляем операндум из файла

Транслируем с получением файла листинга:

```
aaskobeeva@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:16: error: invalid combination of opcode and operands
aaskobeeva@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
aaskobeeva@fedora:~/work/arch-pc/lab07$
```

Рис. 2.13: Транслируем файл

При трансляции файла, выдается ошибка, но создаются исполнительные файлы lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его:

```
 1                                  %include 'in_out.asm'
 1                              <1> ;---------------  slen  --------------------
 2                              <1> ; Функция вычисления длины сообщения
 3                              <1> slen:
 4 00000000 53                  <1>     push    ebx
 5 00000001 89C3                <1>     mov     ebx, eax
 6                              <1>
 7                              <1> nextchar:
 8 00000003 803800              <1>     cmp     byte [eax], 0
 9 00000006 7403                <1>     jz      finished
10 00000008 40                  <1>     inc     eax
11 00000009 EBF8                <1>     jmp     nextchar
12                              <1>
13                              <1> finished:
14 0000000B 2918                <1>     sub     eax, ebx
15 0000000D 5B                  <1>     pop     ebx
16 0000000E C3                  <1>     ret
17                              <1>
18                              <1>
19                              <1> ;---------------  sprint  --------------------
20                              <1> ; Функция печати сообщения
21                              <1> ; входные данные: mov eax,<message>
22                              <1> sprint:
23 0000000F 52                  <1>     push    edx
24 00000010 51                  <1>     push    ecx
25 00000011 53                  <1>     push    ebx
26 00000012 50                  <1>     push    eax
27 00000013 E8E8FFFFFF          <1>     call    slen
28                              <1>
29 00000018 89C2                <1>     mov     edx, eax
30 0000001A 58                  <1>     pop     eax
31                              <1>
32 0000001B 89C1                <1>     mov     ecx, eax
33 0000001D BB01000000          <1>     mov     ebx, 1
34 00000022 B804000000          <1>     mov     eax, 4
35 00000027 CD80                <1>     int     80h
36                              <1>
37 00000029 5B                  <1>     pop     ebx
38 0000002A 59                  <1>     pop     ecx
39 0000002B 5A                  <1>     pop     edx
40 0000002C C3                  <1>     ret
41                              <1>
42                              <1>
43                              <1> ;---------------  sprintLF  ----------------
44                              <1> ; Функция печати сообщения с переводом строки
45                              <1> ; входные данные: mov eax,<message>
46                              <1> sprintLF:
```

Рис. 2.14: Изучаем файл с ошибкой

2.3 Задание для самостоятельной работы

ВАРИАНТ-17

Задание 1.

Создаем новый файл и открываем его и пишем программу, которая выберет наименьшее число из трех(2 числа уже в программе,3е выводится из консоли):

```
%include 'in_out.asm'
section        .data
   msg1 db 'Введите B: ',0h
   msg2 db "Наименьшее число: ",0h
   A dd '26'
   C dd '68'
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]
   jl check_B
   mov ecx,[C]
   mov [min],ecx
check_B:
   mov  eax,min
   call atoi
   mov  [min],eax
   mov  ecx,[min]
   cmp  ecx,[B]
   jl   fin
   mov  ecx,[B]
   mov  [min],ecx
fin:
   mov eax, msg2
   call sprint
   mov eax,[min]
   call iprintLF
   call quit
```

Рис. 2.15: Пишем программу

Транслируем файл и смотрим на работу программы:

```
aaskobeeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aaskobeeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
aaskobeeva@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 12
Наименьшее число: 12
aaskobeeva@fedora:~/work/arch-pc/lab07$
```

Рис. 2.16: Программа работает корректно

Задание 2.

Создаем новый файл, открываем его и пишем программу, которая решит систему уравнений при данных, введенных в консоль:

```
%include 'in_out.asm'
SECTION .data
   msg1: DB 'Введите x: ',0h
   msg2: DB 'Введите a: ',0h
   otv: DB 'F(x) = ',0h
SECTION .bss
   x: RESB 80
   a: RESB 80
   res: RESB 80
SECTION .text
   GLOBAL _start
_start:
   mov eax,msg1
   call sprint
   mov ecx,x
   mov  edx,80
   call sread
   mov eax,x
   call atoi
   mov [x],eax

   mov eax,msg2
   call sprint
   mov ecx,a
   mov edx,80
   call sread
   mov eax,a
   call atoi
   mov [a],eax

   cmp eax,8
   jl calculate_add

   mov eax,[x]
   imul eax,[a]
   jmp store_result

calculate_add:
   mov eax,[a]
   add eax,8

store_result:
   mov [res],eax

   mov eax,otv
   call sprint
   mov eax,[res]
   call iprintLF
   call quit
```

Рис. 2.17: Пишем программу

Транслируем файл и проверяем его работу при x=3, a=4:

```
aaskobeeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aaskobeeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aaskobeeva@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 4
F(x) = 12
```

Рис. 2.18: Программа работает корректно

Компилируем программу и проверяем для x=2, a=9:

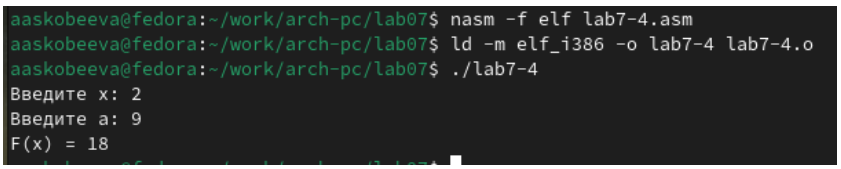

Рис. 2.19: Программа работает корректно

3 Выводы

Мы познакомились со структурой файла листинга, изучили команды условного и безусловного перехода.