Лабораторная работа №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений

Коровкин Никита Михайлович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Понять принцип работы условных и безусловных переходов в Ассемблере и научиться писать программы с командами, отвечающими за переходы. Научиться работать с файлами листинга и уметь их читать.

2 Выполнение лабораторной работы

Для начала нам необходимо создать рабочую папку lab07 и файл lab7-1.asm (рис.1):

```
liveuser@localhost-live:~$ cd ~/work/arch-pc/
liveuser@localhost-live:~/work/arch-pc$ mkdir lab07
liveuser@localhost-live:~/work/arch-pc$ touch lab7-1.asm
liveuser@localhost-live:~/work/arch-pc$ mv lab7-1.asm lab07
liveuser@localhost-live:~/work/arch-pc$
```

Рис. 2.1: Создание папки и файла lab7-1.asm

Создав файл, вставим туда код из листинга.(рис.2)

```
%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
```

Рис. 2.2: Вставляем код

После этого перенесем в нужную папку файл in_out.asm, чтобы код работал.(рис.3)

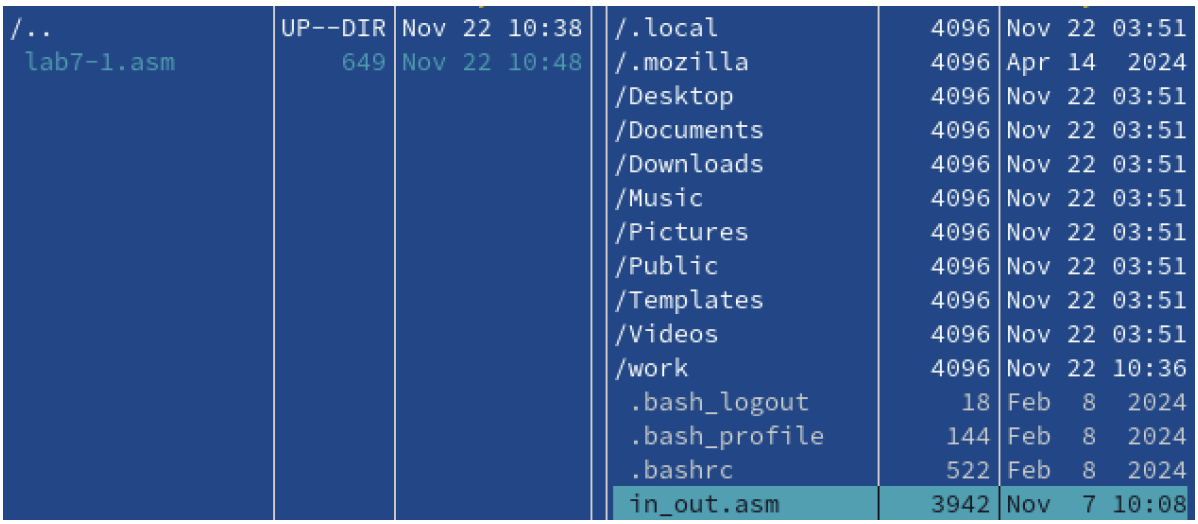

Рис. 2.3: перенос файла в нужную директорию

Соберем нашу программу и запустим ее.(рис.4)

```
liveuser@localhost-live:~/work/arch-pc$ nasm -f elf lab7-1.asm
nasm: fatal: unable to open input file `lab7-1.asm' No such file or directory
liveuser@localhost-live:~/work/arch-pc$ cd lab07
liveuser@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
liveuser@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
liveuser@localhost-live:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
liveuser@localhost-live:~/work/arch-pc/lab07$
```

Рис. 2.4: Запуск кода

После этого отредактируем файл, как это сделано во втором листинге.(рис.5)

```
%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
```

Рис. 2.5: Редактирование кода

Снова запустим наш код.(рис.6)

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

Рис. 2.6: Повторный запуск кода

После этого перепишем код, чтобы он выводил сообщения в обратном порядке - от 3 сообщения к первому.(рис.7)

```
%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
```

Рис. 2.7: Повторное изменение кода

Проверим и запустим код.(рис.8)

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

Рис. 2.8: проверка работоспособности кода

Код работает верно.

Теперь создадим второй файл.(рис.9)

```
liveuser@localhost-live:~/work/arch-pc/lab07$ touch lab7-2.asm
liveuser@localhost-live:~/work/arch-pc/lab07$
```

Рис. 2.9: Создание второго файла

Вставим туда код из третьего листинга и запустим файл, проверяя, как он работает.(рис.10)

```
liveuser@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
liveuser@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
liveuser@localhost-live:~/work/arch-pc/lab07$ ./lab7-2.asm
bash: ./lab7-2.asm: Permission denied
liveuser@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
liveuser@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 15
Наибольшее число: 50
liveuser@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 65
Наибольшее число: 65
```

Рис. 2.10: Проверка кода второго файла

Теперь создадим файл листинга при сборке файла.(рис.11)

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

Рис. 2.11: Создание файла листинга

Откроем файл.(рис.12)

```
liveuser@localhost-live:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 2.12: Открытие файла

Перед нами появляется такое окно.(рис.13)

```
lab7-2.lst        [----]  0 L:[  1+ 0   1/225] *(0    /14451b) 0032 0x020        [*][
    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.....................
```

Рис. 2.13: Открытый файл

Программа будет находиться ниже.(рис.14)

```
  170 000000E7 C3               <1>     ret
    2                               section .data
    3 00000000 D092D0B2D0B5D0B4D0-  msg1 db 'Введите B: ',0h
    3 00000009 B8D182D0B520423A20-
    3 00000012 00................
    4 00000013 D09DD0B0D0B8D0B1D0-  msg2 db "Наибольшее число: ",0h
    4 0000001C BED0BBD18CD188D0B5-
    4 00000025 D0B520D187D0B8D181-
    4 0000002E D0BBD0BE3A2000.....
    5 00000035 32300000             A dd '20'
    6 00000039 35300000             C dd '50'
    7                               section .bss
    8 00000000 <res Ah>             max resb 10
    9 0000000A <res Ah>             B resb 10
   10                               section .text
   11                               global _start
   12                               _start:
   13                               ; ---------- Вывод сообщения 'Введите B: '
```

Рис. 2.14: Машинный код и текст программы

Попробуем разобрать несколько строк кода.

1) Строка 3 отвечает за содержимое сообщения.

2) Строка 11 отвечает за начало основной рабочей части кода

3) 14 строка отправляет сообщение в нужный регистр

Теперь допустим ошибку в нашем коде.(рис.15)

```
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,
mov edx,10
call sread
```

Рис. 2.15: Допущение ошибки

Создадим файл листинга еще раз.(рис.16)

```
liveuser@localhost-live:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
liveuser@localhost-live:~/work/arch-pc/lab07$
```

Рис. 2.16: Повторное создание файла листинга

У нас возникает ошибка. Откроем файл и найдем ошибку.(рис.17)

```
  5 00000035 32300000                A dd '20'
  6 00000039 35300000                C dd '50'
  7                                  section .bss
  8 00000000 <res Ah>                max resb 10
  9 0000000A <res Ah>                B resb 10
 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                                  mov ecx,
 17                ******************  error: invalid combination of opcode and oper
 18 000000F2 BA0A000000              mov edx,10
```

Рис. 2.17: Ошибка в файле

Как можно заметить, в листинге прописана ошибка.

#Выполнение Самостоятельной работы

Для начала создадим первый файл.(рис.18)

```
liveuser@localhost-live:~/work/arch-pc/lab07$ touch task1v16.asm
liveuser@localhost-live:~/work/arch-pc/lab07$
```

Рис. 2.18: Создание первого файла

Запишем в него код, который будет находить наименьшее число среди трех. Вариант выполняемого задания - 16.(рис.19)

```
A dd '44'
B dd '74'
C dd '17'
section .bss
min resb 10
section .text
global _start
_start:

mov eax, B
call atoi
mov [B],eax ; запись преобразованного числа в 'B'

mov eax,A
call atoi ; Вызов подпрограммы перевода символа в число
mov [A],eax ; запись преобразованного числа в `A`
```

Рис. 2.19: Код первого файла

Запустим файл. На выходе получаем число 17. Это верный ответ.(рис.20)

```
liveuser@localhost-live:~/work/arch-pc/lab07$ nasm -f elf task1v16.asm
liveuser@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o task1v16 task1v16.o
liveuser@localhost-live:~/work/arch-pc/lab07$ ./task1v16
Наименьшее число: 17
liveuser@localhost-live:~/work/arch-pc/lab07$
```

Рис. 2.20: Проверка файла

Теперь создадим второй файл и запишем туда следующий код.(рис.21-23)

```
%include 'in_out.asm'

section .data
msg1 DB "Введите x: ",0h
msg2 DB "Введите a: ",0h
msg3 DB "Ответ=",0h

section .bss
x: RESB 80
a: RESB 80
ans: 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

mov eax, [x]
cmp eax, [a]
jl xsa

imul eax, [a]
jmp ansv

xsa:

add eax, 4

ansv:
mov [ans],eax
mov eax,msg3
call sprint
```

Запустим файл и попробуем две пары чисел.(рис.24)

```
liveuser@localhost-live:~/work/arch-pc/lab07$ nasm -f elf task2v16.asm
liveuser@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o task2v16  task2v16.o
liveuser@localhost-live:~/work/arch-pc/lab07$ ./task2v16
Введите x: 1
Введите a: 1
Ответ=1
liveuser@localhost-live:~/work/arch-pc/lab07$ ./task2v16
Введите x: 7
Введите a: 1
Ответ=7
liveuser@localhost-live:~/work/arch-pc/lab07$
```

Рис. 2.21: Проверка кода разными числами

Код работает верно. Самостоятельная работа выполнена правильно.

3 Выводы

В ходе выполнения лабораторной работы были получены навыки работы с командами условных и безусловных переходов и написаны программы для закрепления материала