

Отчёт по лабораторной работе № 7

Дисциплина: Архитектура компьютера

Мутале Чали

Содержание

1	Цель работы.....	1
2	Задание	1
3	Выполнение лабораторной работы	1
4	Выводы.....	11
5	Список литературы.....	11

1 Цель работы

Цель этой работы - изучение команд условного и безусловного переходов.
Приобретение навыков написания программ с использованием переходов.
Знакомство с назначением и структурой файла листинга.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга

3 Выполнение лабораторной работы

1. Реализация переходов в NASM

Создаю каталог для программам лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm:

```
(cmutale@cmutale)-[~]  
$ cd ~/work/arch-pc  
  
(cmutale@cmutale)-[~/work/arch-pc]  
$ mkdir -p ~/work/arch-pc/lab07  
  
(cmutale@cmutale)-[~/work/arch-pc]  
$ cd ~/work/arch-pc/lab07  
  
(cmutale@cmutale)-[~/work/arch-pc/lab07]  
$
```

Рис 1

Открываю файл lab7-1.asm и в него вставляю код программы, которая показывает как работает jmp:

```
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 msg1: DB 'Сообщение № 1',0
6 msg2: DB 'Сообщение № 2',0
7 msg3: DB 'Сообщение № 3',0
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12
13 jmp _label2
14
15 _label1:
16 mov eax, msg1
17 call sprintf
18
19 _label2:
20 mov eax, msg2
21 call sprintf
22
23 _label3:
24 mov eax, msg3
25 call sprintf
26
27 _end:
28 call quit
```

Рис 2

Создаю исполняемый файл и запускаю его. Программа выводит “сообщение № 2” и “сообщение № 3”:

```

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-1.asm

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-1 lab7-1.o

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ./lab7-1
Сообщение № 2
Сообщение № 3

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$

```

Рис 3

Изменяю текст программы:

```

1 %include 'in_out.asm'
2
3 SECTION .data
4
5 msg1: DB 'Сообщение № 1',0
6 msg2: DB 'Сообщение № 2',0
7 msg3: DB 'Сообщение № 3',0
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12
13 jmp _label2
14
15 _label1:
16 mov eax, msg1
17 call sprintf
18 jmp _end
19
20 _label2:
21 mov eax, msg2
22 call sprintf
23 jmp _label1
24
25 _label3:
26 mov eax, msg3
27 call sprintf
28
29 _end:
30 call quit

```

Рис 4

Создаю исполняемый файл и запускаю его. Программа выводит “сообщение № 2” и “сообщение № 1”:

```
(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-1.asm

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-1 lab7-1.o

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис 5

Изменяю текст программы, чтобы она выводила “сообщение № 3”, “сообщение № 2” и “сообщение № 1”:

```
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 msg1: DB 'Сообщение № 1',0
6 msg2: DB 'Сообщение № 2',0
7 msg3: DB 'Сообщение № 3',0
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12
13 jmp _label3
14
15 _label1:
16 mov eax, msg1
17 call sprintf
18 jmp _end
19
20 _label2:
21 mov eax, msg2
22 call sprintf
23 jmp _label1
24
25 _label3:
26 mov eax, msg3
27 call sprintf
28
29 _end:
30 call quit
```

Рис 6

Создаю исполняемый файл и запускаю его:

```

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-1.asm

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-1 lab7-1.o

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ./lab7-1
Сообщение № 3

```

Рис 7

Создаю файл lab7-2.asm:

```

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ touch lab7-2.asm

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm

```

Рис 8

В него вставляю код программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных:

```

%include 'in_out.asm'

section .data
msg1 db 'Введите В: ',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 В: '

mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread

```

Рис 9

```

; ----- вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit

```

Рис 10

Создаю исполняемый файл и запускаю его:

```
(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-2.asm

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-2 lab7-2.o

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ./lab7-2
Введите В: 
```

Рис 11

Проверяю работу для разных значений В:

```
(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ./lab7-2
Введите В: 4
Наибольшее число: 50

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ./lab7-2
Введите В: 56
Наибольшее число: 56

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ./lab7-2
Введите В: 30
Наибольшее число: 50
```

Рис 12

2. Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm:

```
(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ nasm -f elf -l lab7-2.lst lab7-2.asm

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst
lab7-1      lab7-1.o   lab7-2.asm  lab7-2.o
```

Рис 13

Открываю файл листинга lab7-2.lst с помощью mcedit:

```

cmutale@cmutale: ~/work/arch-pc/lab07
File Actions Edit View Help
/home/cm~7-2.lst [—] 0 L:[188+24 212/229] *(12808/14146b) 0032 0x[*][X]
13 section .text
14 global _start
15 _start:
16 ; ----- Вывод сообщения 'Введите
17
18 000000E8 B8[00000000] mov eax,msg1
19 000000ED E81DFFFFFF call sprint
20 ; ----- Ввод 'B'
21 000000F2 B9[0A000000] mov ecx,B
22 000000F7 BA0A000000 mov edx,10
23 000000FC E842FFFFFF call sread
24 ; ----- Преобразование 'B' из си
25 00000101 B8[0A000000] mov eax,B
26 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перево
27 0000010B A3[0A000000] mov [B],eax ; запись преобразованного
28 ; ----- Записываем 'A' в перемен
29 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
30 00000116 890D[00000000] mov [max],ecx ; 'max = A'
31 ; ----- Сравниваем 'A' и 'C' (ка
32 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
33 00000122 7F0C jg check_B ; если 'A>C', то переход н
34 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
35 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
36 ; ----- Преобразование 'max(A,C)
37 check_B:

```

Рис 14

Это пример машинного кода сохранен в lab7-2.lst:

```

20 000000F2 B9[0A000000] mov ecx,B
21 000000F7 BA0A000000 mov edx, 10
22 000000FC E842FFFFFF call sread

```

В lab7-2.asm, эти строки предназначены для ввода значения B. 20- номер строки, 000000F2- это смещение машинного кода от начала текущего сегмента (адрес), B9[0A000000]- машинный код и mov ecx,B- исходный текст программы.

Когда я удаляю строку для сравнения A и C, выполняю трансляцию с получением файла листинга, строка удаляется из файла .lst, и ничего не добавляется:

```

26 mov [B],eax
27
28 mov ecx,[A]
29 mov [max],ecx
30
31
32 jg check_B
33 mov ecx,[C]
34

```

Рис 15


```

33 00000122 7F0C          jg check_B ; если 'A>C', то переход н
34 00000124 8B0D[39000000]  mov ecx,[C] ; иначе 'ecx = C'
35 0000012A 890D[00000000]  mov [max],ecx ; 'max = C'
36                      ; ----- Преобразование 'max(A,C)
37                      check_B:

```

Рис 16

#Выполнение задания для самостоятельной работы

Создаю файл task1.asm:

```

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ touch task1.asm

```

Рис 17

В него вставляю код программы, которая определяет наименьшей из 3 целочисленных переменных a,b и c:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg1 db "Наименьшее число: ",0h
```

```
A dd '41'
```

```
B dd '35'
```

```
C dd '62'
```

```
SECTION .bss
```

```
min resb 10
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax,B
```

```
call atoi
```

```
mov [B],eax
```

```
mov ecx,[A]
```

```
mov [min],ecx
```

```
cmp ecx,[C]
```

```
j1 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,msg1
call sprint
mov eax,[min]
call iprintLF
call quit

```

```

mc [cmutale@cmutale]:~/work/arch-pc/lab07
File Actions Edit View Help
GNU nano 7.2 /home/cmutale/work/arch-pc/lab07/task1.asm
#include 'in_out.asm'

SECTION .data
msg1 db "Наименьшее число: ",0h
A da '41'
B dd '62'
C dd '35'

SECTION .bss
min resb 10

SECTION .text
GLOBAL _start

_start:

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

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

```

Создаю исполняемый файл и проверяю его работу для значения переменных из варианта 10:

```
(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ nasm -f elf task1.asm

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o task1 task1.o

(cmutale@cmutale)-[~/work/arch-pc/lab07]
$ ./task1
Наименьшее число: 35
```

Рис 19

4 Выводы

При выполнении лабораторной работы, я изучила команд условного и безусловного переходов в NASM.

5 Список литературы

Архитектура ЭВМ