

Отчёт по лабораторной работе 7

Архитектура компьютеров

Махкамов Рауфджон НММбд-04-24

Содержание

1	Цель работы	5
2	Теоретическое введение	6
2.1	Команды перехода	6
2.2	Листинг	7
3	Выполнение лабораторной работы	8
3.1	Реализация переходов в NASM	8
3.2	Изучение структуры файла листинга	13
3.3	Самостоятельное задание	15
4	Выводы	19

Список иллюстраций

Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Теоретическое введение

2.1 Команды перехода

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление.

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

2.2 Листинг

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

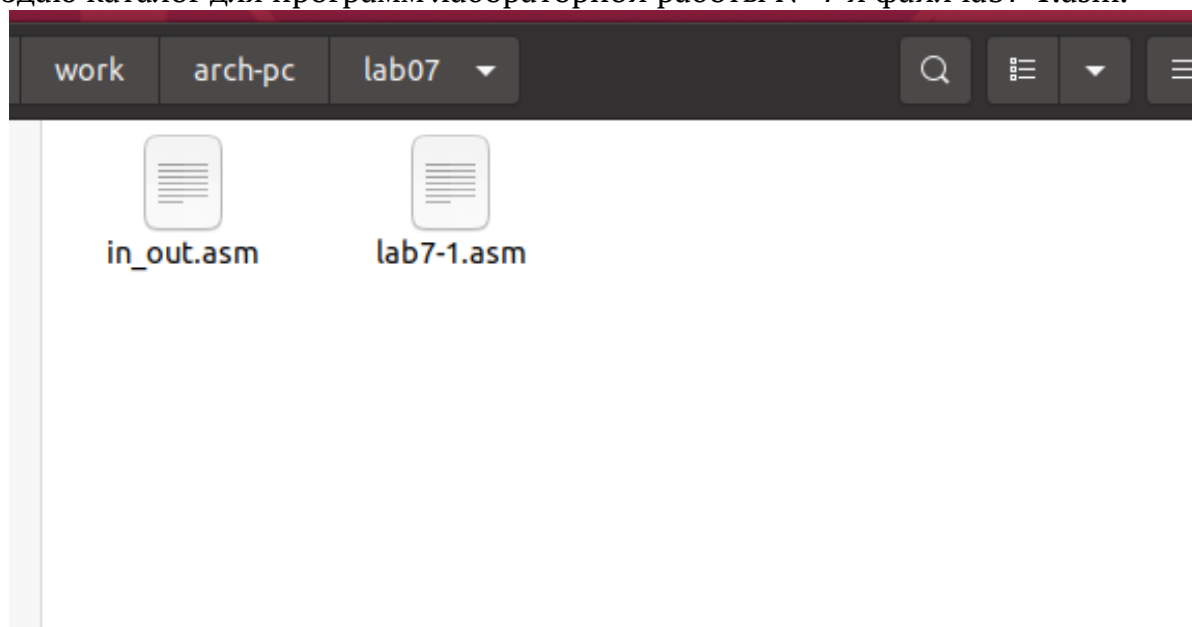
Итак, структура листинга:

- номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);
- адрес — это смещение машинного кода от начала текущего сегмента;
- машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция `int 80h` начинается по смещению `00000020` в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `int 80h` ассемблируется в `CD80` (в шестнадцатеричном представлении); `CD80` — это инструкция на машинном языке, вызывающая прерывание ядра)
- исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается)

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7 и файл lab7-1.asm.



(рис.

)

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл `lab7-1.asm` текст программы из листинга 7.1. (рис.


```

/home/rm~7-1.asm [----] 9 L:[ 1+24 25/
%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

```

```

rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/
7-1
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/
Сообщение № 2
Сообщение № 3
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/

```

Создаю исполняемый файл и запускаю его. (рис.

Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Изменяю программу таким образом, чтобы она выводила сначала «Сообщение № 2», затем «Сообщение № 1», и завершала работу. Для этого после вывода сообщения № 2 добавляю инструкцию `jmp` с меткой `_label1` (переход к

инструкциям вывода сообщения № 1), и после вывода сообщения № 1 добавляю инструкцию `jmp` с меткой `_end` (переход к инструкции `call quit`).

```
rmahkamov@Ubuntu-VirtualBox:
rmahkamov@Ubuntu-VirtualBox:
7-1
rmahkamov@Ubuntu-VirtualBox:
Сообщение № 2
Сообщение № 3
rmahkamov@Ubuntu-VirtualBox:
```

Изменяю текст программы в соответствии с листингом 7.2. (рис.

```
/home/rm~7-1.asm  [----]  5 L:[ 1+25 26/ 28
%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.)

После изменений программа выводит следующее: Сообщение № 3 Сообщение № 2 Сообщение № 1

```

/home/rm~7-1.asm [----] 0 L:[ 1+15 16/ 29]
%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.

) (рис.

```

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

```

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, то есть переход должен осуществляться только при выполнении определенного

условия. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшее из трех целочисленных переменных: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создаю исполняемый файл и проверяю его работу для различных значений B

```
/home/gm~7-2.asm [----] 0 L:[ 19+ 6 25/ 49] *(533 /1056b)[*][X
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A].
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C].
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprintf
mov eax,[max]
call iprintfLF
call quit
```

(рис.

```

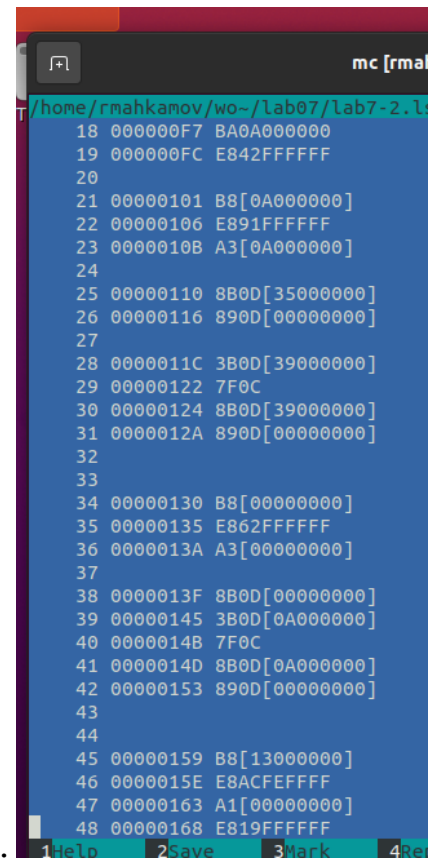
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$

```

(рис.

3.2 Изучение структуры файла листинга

Обычно nasm создает в результате ассемблирования только объектный файл. Чтобы получить файл листинга, необходимо указать ключ -l и задать имя файла листинга в командной строке.



```

/home/rmahkamov/wo~/lab07/lab7-2.l
18 000000F7 BA0A000000
19 000000FC E842FFFFFF
20
21 00000101 B8[0A000000]
22 00000106 E891FFFFFF
23 0000010B A3[0A000000]
24
25 00000110 8B0D[35000000]
26 00000116 890D[00000000]
27
28 0000011C 3B0D[39000000]
29 00000122 7F0C
30 00000124 8B0D[39000000]
31 0000012A 890D[00000000]
32
33
34 00000130 B8[00000000]
35 00000135 E862FFFFFF
36 0000013A A3[00000000]
37
38 0000013F 8B0D[00000000]
39 00000145 3B0D[0A000000]
40 0000014B 7F0C
41 0000014D 8B0D[0A000000]
42 00000153 890D[00000000]
43
44
45 00000159 B8[13000000]
46 0000015E E8ACFFFFFF
47 00000163 A1[00000000]
48 00000168 E819FFFFFF

```

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

Ознакомимся с его форматом и содержимым.

- строка 211:

- 34 - номер строки
 - 0000012E - адрес
 - B8[00000000] - машинный код
 - mov eax, max - код программы
- строка 212:
 - 35 - номер строки
 - 00000133 - адрес
 - E864FFFFFF - машинный код
 - call atoi - код программы
 - строка 213:
 - 36 - номер строки
 - 00000138 - адрес
 - A3[00000000] - машинный код
 - mov [max], eax - код программы

Открываю файл с программой lab7-2.asm и удаляю один операнд из инструкции с двумя операндами. Затем выполняю трансляцию с получением

```
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
-2.lst
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
-2.lst
lab7-2.asm:39: error: invalid combination of opcode and operands
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$
```

файла листинга. (рис. 2.10)

```
/home/rmahkamov/wo~/lab07/lab7-2.lst [----] 0 L:[196+30 226/226] *(13859/13859b) <EOF> [*][X]
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A].
26 00000116 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C jg check_B
30 00000124 8B0D[39000000] mov ecx,[C].
31 0000012A 890D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi
36 0000013A A3[00000000] mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000] mov ecx,[max]
39 cmp ecx,
39 *****
40 00000145 7F0C jg fin
41 00000147 8B0D[0A000000] mov ecx,[B]
42 0000014D 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000153 B8[13000000] mov eax, msg2
46 00000158 E8B2FEFFFF call sprint
47 0000015D A1[00000000] mov eax,[max]
48 00000162 E81FFFFFFF call iprintlnLF
49 00000167 E86FFFFFFF call quit
```

(рис.

Объектный файл не смог создаться из-за ошибки, но файл листинга с выделенным местом ошибки был получен.

3.3 Самостоятельное задание

Напишите программу нахождения наименьшей из трех целочисленных переменных a, b и c. Значения переменных выбрать из таблицы 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создаю исполняемый файл и проверяю его работу (рис.

```

/home/rm~7-3.asm  [----]  0 L:[ 40+ 1 41/ 70
mov edx,80
call sread.
mov eax,C
call atoi
mov [C],eax...
....
mov ecx,[A]
mov [min],ecx

cmp ecx, [B]
jl check_C
mov ecx, [B]
mov [min], ecx

check_C:
cmp ecx, [C]
jl finish
mov ecx,[C]
mov [min],ecx.

finish:
mov eax,answer
call sprint

mov eax, [min]
call iprintLF

call quit

```

) (рис.

```

rmahkamov@Ubuntu-VirtualBo
rmahkamov@Ubuntu-VirtualBo
rmahkamov@Ubuntu-VirtualBo
rmahkamov@Ubuntu-VirtualBo
7-3
rmahkamov@Ubuntu-VirtualBo
Input A: 99
Input B: 29
Input C: 26
Smallest: 26
rmahkamov@Ubuntu-VirtualBo

```

Для варианта 12 - 99,29,26

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. Создаю исполняемый файл и проверяю его работу для значений X и a


```

/home/rmahkamov/lab7-4.asm [13:21:20] 13 L: [21+20 47/3]
    call atoi.
    mov [A],eax

    mov eax,msgX
    call sprint
    mov ecx,X
    mov edx,80
    call sread.
    mov eax,X
    call atoi
    mov [X],eax...

    mov ebx, [X]
    mov edx, 5
    cmp ebx, edx
    jb first
    jmp second

first:
    mov eax,[A]
    mov ebx,[X]
    mul ebx
    call iprintLF.
    call quit
second:
    mov eax,[X]
    sub eax,5
    call iprintLF.
    call quit

```

из 7.6. (рис.

) (рис.

```

rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab
7-4
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Input A: 7
Input X: 3
21
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Input A: 4
Input X: 6
1
rmahkamov@Ubuntu-VirtualBox:~/work/arch-pc/lab07$

```

Для варианта 12:

$$\begin{cases} xa, & x < 5 \\ x - 5, & x \geq 5 \end{cases}$$

При (x = 3, a = 7) получается 21

При (x = 6, a = 4) получается 1

4 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.