

Лабораторная работа №7

НКАБд-06-23

Улитина Мария Максимовна

Содержание

1 Цель работы.....	1
2 Задание	1
3 Теоретическое введение	2
3.1 Команды безусловного перехода	2
3.2 Команды условного перехода.....	2
3.2.1 Регистр флагов.....	2
3.2.2 Описание инструкции стр.....	2
3.2.3 Описание команд условного перехода.....	3
3.3 Файл листинга и его структура	3
4 Выполнение лабораторной работы	3
4.1 Задания для самостоятельной работы	9
5 Выводы	9
6 Список литературы.....	9

1 Цель 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 файла листинга.

3. Написание программы нахождения наименьшей из 3 целочисленных переменных.
4. Написание программы для вычисления заданной функции.

3 Теоретическое введение

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.

3.1 Команды безусловного перехода

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление: `jmp <адрес_перехода>`. Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре.

3.2 Команды условного перехода

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

3.2.1 Регистр флагов

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора. В следующей таблице указано положение битовых флагов в регистре флагов.

3.2.2 Описание инструкции `cmp`

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: `cmp <операнд_1>, <операнд_2>`. Команда `cmp`, так же как и команда вычитания, выполняет вычитание, но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов.

3.2.3 Описание команд условного перехода

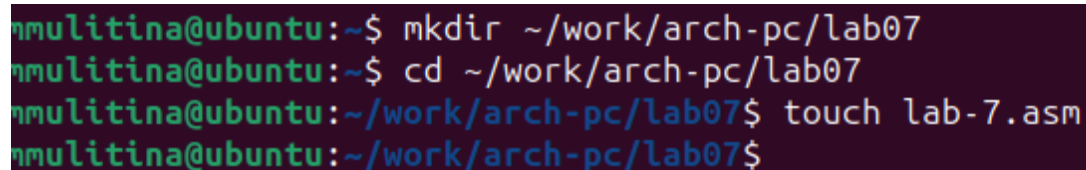
Команда условного перехода имеет вид `j < мнемоника перехода > label`. Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов. В табл. 7.3. представлены команды условного перехода, которые обычно ставятся после команды сравнения `str`. В их мнемокодах указывается тот результат сравнения, при котором надо делать переход. Мнемоники, идентичные по своему действию, написаны в таблице через дробь (например, `ja` и `jnb`). Программист выбирает, какую из них применить, чтобы получить более простой для понимания текст программы.

3.3 Файл листинга и его структура

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.

4 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы №7, перейдём в него и создадим файл `lab7-1.asm` (рис. 1).



```
mmulitina@ubuntu:~$ mkdir ~/work/arch-pc/lab07
mmulitina@ubuntu:~$ cd ~/work/arch-pc/lab07
mmulitina@ubuntu:~/work/arch-pc/lab07$ touch lab-7.asm
mmulitina@ubuntu:~/work/arch-pc/lab07$
```

Figure 1: Создание файла

Введём в файл текст программы (рис. 2).

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения

```

Figure 2: Текст программы

Создадим исполняемый файл и запустим его (рис. 3).

```

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

```

Figure 3: Работа программы

Изменим текст программы (рис. 4).

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:

```

Figure 4: Измененный текст программы

Запустим программу и проверим его работу (рис. 5).

```

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

```

Figure 5: Работа программы

Изменим текст программы (рис. 6).

```

GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Figure 6: Изменение текста программы

Запустим программу (рис. 7).

```

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

```

Figure 7: Работа программы

Создадим файл lab7-2.asm (рис. 8).

```

mmulitina@ubuntu:~/work/arch-pc/lab07$ touch lab7-2.asm

```

Figure 8: Создание файла

и введём в неё предложенный текст (рис. 9).

```

#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

```

Figure 9: Ввод текста

Проверим работу программы с разными значениями (рис. 10).

```

mmulitina@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mmulitina@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mmulitina@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 5
Наибольшее число: 50
mmulitina@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 5555.
Наибольшее число: 5555
mmulitina@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60

```

Figure 10: Проверка

Получим объектный файл, указав ключ -l (рис. 11).

```

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>

```

Figure 11: Листинг

Откроем файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалим один операнд. (рис. 12).

```

; ----- Сравниваем 'A' и 'C' (как символы)
cmp esx ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov esx,[C] ; иначе 'esx = C'
mov [max],esx ; 'max = C'

```

Figure 12: Изменение программы

Откроем файл листинга (рис. 13).

```

28          *****
28          *****
and operands
29 0000011C 7F0C          cmp esx ; Сравниваем 'A' и 'C'
метку 'check_B',          error: invalid combination of opcode
                          jg check_B ; если 'A>C', то переход на

```

Figure 13: Листинг

(рис. 14).

```

mmulitina@ubuntu:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands

```

Figure 14: Программа

4.1 Задания для самостоятельной работы

Напишем программу для нахождения наименьшей из 3 целочисленных переменных (рис. 15).

```
mmulitina@ubuntu:~/work/arch-pc/lab07$ nasm -f elf prog1.asm
mmulitina@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o prog1 prog1.o
mmulitina@ubuntu:~/work/arch-pc/lab07$ ./prog1
Наименьшее число: 5
```

Figure 15: Программа

Напишем программу для вычисления функции для введенных с клавиатуры значений (рис. 16).

```
mmulitina@ubuntu:~/work/arch-pc/lab07$ nasm -f elf prog2.asm
mmulitina@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o prog2 prog2.o
mmulitina@ubuntu:~/work/arch-pc/lab07$ ./prog2
1
4
5
mmulitina@ubuntu:~/work/arch-pc/lab07$ ./prog2
3
9
```

Figure 16: Программа

5 Выводы

В процессе выполнения работы были изучены команды условного и безусловного переходов. Приобретены навыки написания программ с использованием переходов.

6 Список литературы

Архитектура ЭВМ. Лабораторная работа №7.