# [Условные и безусловные переходы](http://asmworld.ru/uchebnyj-kurs/016-uslovnye-i-bezuslovnye-perexody/%22%20%5Co%20%22%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D1%8B%D0%B9%20%D0%BA%D1%83%D1%80%D1%81.%20%D0%A7%D0%B0%D1%81%D1%82%D1%8C%2016.%20%D0%A3%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5%20%D0%B8%20%D0%B1%D0%B5%D0%B7%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5%20%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D1%8B)

**Безусловные переходы**

Безусловный переход — это переход, который выполняется всегда. Безусловный переход осуществляется с помощью команды JMP. У этой команды один операнд, который может быть непосредственным адресом (меткой), регистром или ячейкой памяти, содержащей адрес. Существуют также «дальние» переходы — между сегментами, однако здесь мы их рассматривать не будем. Примеры безусловных переходов:

|  |
| --- |
|  **jmp** metka *;Переход на метку* **jmp** **bx** *;Переход по адресу в BX* **jmp** **word**[**bx**] *;Переход по адресу, содержащемуся в памяти по адресу в BX* |

**Условные переходы**

Условный переход осуществляется, если выполняется определённое условие, заданное флагами процессора (кроме одной команды, которая проверяет CX на равенство нулю). Как вы помните, состояние флагов изменяется после выполнения арифметических, логических и некоторых других команд. Если условие не выполняется, то управление переходит к следующей команде.

Существует много команд для различных условных переходов. Также для некоторых команд есть синонимы (например, JZ и JE — это одно и то же). Для наглядности все команды условных переходов приведены в таблице:

|  |  |  |
| --- | --- | --- |
| **Команда** | **Переход, если** | **Условие перехода** |
| JZ/JE | нуль или равно | ZF=1 |
| JNZ/JNE | не нуль или не равно | ZF=0 |
| JC/JNAE/JB | есть переполнение/не выше и не равно/ниже | CF=1 |
| JNC/JAE/JNB | нет переполнения/выше или равно/не ниже | CF=0 |
| JP | число единичных бит чётное | PF=1 |
| JNP | число единичных бит нечётное | PF=0 |
| JS | знак равен 1 | SF=1 |
| JNS | знак равен 0 | SF=0 |
| **JO** | есть переполнение | **OF=1** |
| **JNO** | нет переполнения | **OF=0** |
| JA/JNBE | выше/не ниже и не равно | CF=0 и ZF=0 |
| JNA/JBE | не выше/ниже или равно | CF=1 или ZF=1 |
| JG/JNLE | больше/не меньше и не равно | ZF=0 и SF=OF |
| JGE/JNL | больше или равно/не меньше | SF=OF |
| JL/JNGE | меньше/не больше и не равно | SF≠OF |
| JLE/JNG | меньше или равно/не больше | ZF=1 или SF≠OF |
| JCXZ | содержимое CX равно нулю | CX=0 |

У всех этих команд один операнд — имя метки для перехода. Обратите внимание, что некоторые команды применяются для беззнаковых чисел, а другие — для чисел со знаком. Сравнения «выше» и «ниже» относятся к беззнаковым числам, а «больше» и «меньше» — к числам со знаком. **Для беззнаковых чисел признаком переполнения будет флаг CF**, а соответствующими командами перехода **JC** и **JNC**. **Для чисел со знаком** **о переполнении можно судить по состоянию флага** **OF**, поэтому им соответствуют команды перехода **JO** и **JNO**. Команды переходов не изменяют значения флагов.

В качестве примера я приведу небольшую программу для сложения двух чисел со знаком с проверкой переполнения. В случае переполнения будет выводиться сообщение об ошибке. Вы можете поменять значения объявленных переменных, чтобы переполнение возникало или не возникало при их сложении, и посмотреть, что будет выводить программа.

|  |  |
| --- | --- |
| 123456789101112131415161718192021222324252627282930 | **use16** *;Генерировать 16-битный код*org 100h *;Программа начинается с адреса 100h*  **mov** **al**,[x] *;AL = x* **add** **al**,[y] *;AL = x + y* **jo** error *;Переход, если переполнение* **mov** **ah**,09h *;\* **mov** **dx**,ok\_msg *; > Вывод строки 'OK'* **int** 21h *;/*exit: **mov** **ah**,09h *;\* **mov** **dx**,pak *; > Вывод строки 'Press any key...'* **int** 21h *;/*  **mov** **ah**,08h *;\* **int** 21h *;/ Ввод символа*  **mov** **ax**,4C00h *;\* **int** 21h *;/ Завершение программы*error: **mov** **ah**,09h *;\* **mov** **dx**,err\_msg *; > Вывод сообщения об ошибке* **int** 21h *;/* **jmp** exit *;Переход на метку exit**;----------------------------------------------------------*x **db** -89y **db** -55err\_msg **db** 'Error: overflow detected.',13,10,'$'ok\_msg **db** 'OK',13,10,'$'pak **db** 'Press any key...$' |

**Команды CMP и TEST**

Часто для формирования условий переходов используются команды CMP и TEST. Команда CMP предназначена для сравнения чисел. Она выполняется аналогично команде SUB: из первого операнда вычитается второй, но результат не записывается на место первого операнда, изменяются только значения флагов. Например:

|  |
| --- |
|  **cmp** **al**,5 *;Сравнение AL и 5* **jl** c1 *;Переход, если AL < 5 (числа со знаком)* |
|  **cmp** **al**,5 *;Сравнение AL и 5* **jb** c1 *;Переход, если AL < 5 (числа без знака)* |

Команда TEST работает аналогично команде AND, но также результат не сохраняется, изменяются только флаги. С помощью этой команды можно проверить состояние различных битов операнда. Например:

|  |
| --- |
|  **test** **bl**,00000100b *;Проверить состояние 2-го бита BL* **jz** c2 *;Переход, если 2-й бит равен 0* |

**Пример программы**

Простая программка, которая выводит меню и предлагает пользователю сделать выбор. Для ввода символа используется функция DOS 01h (при вводе символ отображается на экране). В зависимости от введённого символа осуществляется переход на нужный кусок кода. Для разнообразия, я поместил данные в начале программы, а не в конце (кстати, обычно так и делают). Чтобы данные не выполнились как код, перед ними стоит команда безусловного перехода.

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223242526272829303132333435363738394041424344 | **use16** *;Генерировать 16-битный код*org 100h *;Программа начинается с адреса 100h* **jmp** **start** *;Безусловный переход на метку start**;----------------------------------------------------------*menu **db** '1 - Print hello',13,10 **db** '2 - Print go away',13,10 **db** '0 - Exit',13,10,'$'select **db** 13,10,'Select>$'hello **db** 13,10,'Hello!',13,10,13,10,'$'go\_away **db** 13,10,'Go away!',13,10,13,10,'$'*;----------------------------------------------------------***start**: **mov** **ah**,09h *;\* **mov** **dx**,menu *; > Вывод меню* **int** 21h *;/* select\_loop: **mov** **ah**,09h *;\* **mov** **dx**,select *; > Вывод строки 'Select>'* **int** 21h *;/*  **mov** **ah**,01h *;Функция DOS 01h - ввод символа* **int** 21h *;Введённый символ помещается в AL*  **cmp** **al**,'1' *;Сравнение введённого символа с '1'* **je** c1 *;Переход, если равно* **cmp** **al**,'2' *;Сравнение введённого символа с '2'* **je** c2 *;Переход, если равно* **cmp** **al**,'0' *;Сравнение введённого символа с '0'* **je** exit *;Переход, если равно* **jmp** select\_loop *;Безусловный переход*c1: **mov** **ah**,09h *;\* **mov** **dx**,hello *; > Вывод строки 'Hello'* **int** 21h *;/* **jmp** **start** *;Безусловный переход*c2: **mov** **ah**,09h *;\* **mov** **dx**,go\_away *; > Вывод строки 'Go away'* **int** 21h *;/* **jmp** **start** *;Безусловный переход*exit: **mov** **ax**,4C00h *;\* **int** 21h *;/ Завершение программы* |

Скриншот работы программы: