

Учебный курс. Часть 16. Условные и безусловные переходы

Автор: xrnd | Рубрика: [Учебный курс](#) | 27-04-2010 |  [Распечатать запись](#)

Наконец-то мы добрались и до переходов! В этой части научимся программировать условные и безусловные переходы. Вообще, трудно представить себе программу без проверки условий и переходов. С их помощью в программе реализуются различные управляющие конструкции, ветвления и даже циклы.

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 ВХ
jmp word[bx]   ;Переход по адресу, содержащемуся в памяти по адресу в ВХ
```

Условные переходы

Условный переход осуществляется, если выполняется определённое условие, заданное флагами процессора (кроме одной команды, которая проверяет СХ 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.

```

1  use16                ;Генерировать 16-битный код
2  org 100h             ;Программа начинается с адреса 100h
3
4  mov al,[x]           ;AL = x
5  add al,[y]           ;AL = x + y
6  jo error            ;Переход, если переполнение
7  mov ah,09h          ;\
8  mov dx,ok_msg        ; > Вывод строки 'OK'
9  int 21h              ;/
10 exit:
11  mov ah,09h          ;\
12  mov dx,pak          ; > Вывод строки 'Press any key...'
13  int 21h              ;/
14
15  mov ah,08h          ;\
16  int 21h              ;/ Ввод символа
17
18  mov ax,4C00h         ;\
19  int 21h              ;/ Завершение программы
20 error:
21  mov ah,09h          ;\
22  mov dx,err_msg      ; > Вывод сообщения об ошибке
23  int 21h              ;/
24  jmp exit            ;Переход на метку exit
25 ;-----
26 x      db -89
27 y      db -55
28 err_msg db 'Error: overflow detected.',13,10,'$'
29 ok_msg  db 'OK',13,10,'$'
30 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
j1 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.

```
1 use16                ;Генерировать 16-битный код
2 org 100h              ;Программа начинается с адреса 100h
3     jmp start         ;Безусловный переход на метку start
4 ;-----
5 menu    db '1 - Print hello',13,10
6         db '2 - Print go away',13,10
7         db '0 - Exit',13,10,'$'
8 select  db 13,10,'Select>$'
9 hello   db 13,10,'Hello!',13,10,13,10,'$'
10 go_away db 13,10,'Go away!',13,10,13,10,'$'
11 ;-----
12 start:
13     mov ah,09h        ;\
14     mov dx,menu       ; > Вывод меню
15     int 21h           ;/
16
17 select_loop:
18     mov ah,09h        ;\
19     mov dx,select     ; > Вывод строки 'Select>'
20     int 21h           ;/
21
22     mov ah,01h        ;Функция DOS 01h - ввод символа
23     int 21h           ;Введённый символ помещается в AL
24
25     cmp al,'1'        ;Сравнение введённого символа с '1'
26     je c1             ;Переход, если равно
27     cmp al,'2'        ;Сравнение введённого символа с '2'
28     je c2             ;Переход, если равно
29     cmp al,'0'        ;Сравнение введённого символа с '0'
30     je exit           ;Переход, если равно
31     jmp select_loop   ;Безусловный переход
32 c1:
33     mov ah,09h        ;\

```

```

34     mov dx,hello      ; > Вывод строки 'Hello'
35     int 21h           ;/
36     jmp start         ;Безусловный переход
37 c2:
38     mov ah,09h        ;\
39     mov dx,go_away    ; > Вывод строки 'Go away'
40     int 21h           ;/
41     jmp start         ;Безусловный переход
42 exit:
43     mov ax,4C00h      ;\
44     int 21h           ;/ Завершение программы

```

Скриншот работы программы:

```

C:\jumps.com
1 - Print hello
2 - Print go away
0 - Exit

Select>2
Go away!

1 - Print hello
2 - Print go away
0 - Exit

Select>1
Hello!

1 - Print hello
2 - Print go away
0 - Exit

Select>3
Select>8
Select>_

```

Упражнение

Упражнение простое. Напишите программу для сравнения двух переменных со знаком a и b . В зависимости от результатов сравнения выведите « $a < b$ », « $a > b$ » или « $a = b$ ». Проверьте работу программы в отладчике. Результаты можете выкладывать в комментариях.

[Следующая часть »](#)

Комментарии:

RoverWWorm
13-05-2010 12:15

use16
org 100h

jmp start

```

;_____
a db -55
b db -23

```

```
bolshe db 'a>b$'  
menshe db 'a<b$'  
ravno db 'a=b$'
```

```
;_____
```

```
start:
```

```
mov ah,[a]
```

```
mov al,[b]
```

```
cmp ah,al
```

```
jg a_bolshe_b
```

```
jl a_menshe_b
```

```
jz a_ravno_b
```

```
a_bolshe_b:
```

```
mov ah,09
```

```
mov dx,bolshe
```

```
int 21h
```

```
jmp exit
```

```
a_menshe_b:
```

```
mov ah,09
```

```
mov dx,menshe
```

```
int 21h
```

```
jmp exit
```

```
a_ravno_b:
```

```
mov ah,09
```

```
mov dx,ravno
```

```
int 21h
```

```
exit:
```

```
mov ah,08h
```

```
int 21h
```

```
mov ax,4c00h
```

```
int 21h
```

[\[Ответить\]](#)

[xrnd](#)

13-05-2010 14:01

Всё правильно! Но можно немного оптимизировать. Например, команду

```
jg a_bolshe_b
```

можно вообще убрать. Если меньше, то переход. Если равно, то переход. Остаётся только случай, что больше.

[\[Ответить\]](#)

RoverWWorm

13-05-2010 14:19

Точно 😊

[\[Ответить\]](#)

IgorKing
15-09-2010 15:47

А что делает
mov ah,08h
int 21h
что-то не помню такого.

И для общего развития: что происходит, если не указывается конец строки. Я нечаянно перепутал '\$' и 'S'... чуть не умер от неожиданности.

[\[Ответить\]](#)

[xrnd](#)
20-09-2010 16:30

mov ah,08h
int 21h
Это вызов функции 08h DOS — ввод символа с клавиатуры без эха (то есть символ вводится, но не отображается на экране). Здесь это используется, чтобы программа не закрылась сразу после выполнения и можно было посмотреть результат.

Если не указывается конец строки, то на консоль будет выводиться содержимое памяти за строкой: другие данные, код, всякий мусор. До тех пор, пока где-нибудь случайно не встретится символ '\$'

[\[Ответить\]](#)

IgorKing
06-12-2010 15:25

А где тут ошибка: cmp byte[bx+di],byte[si] ? Нельзя сравнивать когда операнды в памяти?

[\[Ответить\]](#)

[xrnd](#)
06-12-2010 16:14

Да. Хотя бы один из операндов должен находится в регистре, либо быть непосредственным значением.

[\[Ответить\]](#)

[Борис](#)
24-12-2010 17:04

еще раз пробую больше не буду
use16 ;gen 16-bit cod
org 100h ;begin 100h
jmp start ;to start
;-----
hello db 'Hello friend',13,10
exitres db 'Press key for exit!!!',13,10

```
amoreb db 'a > b',13,10
bmorea db 'a < b',13,10
aeqb db 'a = b',13,10
```

```
a dw -500
b dw -500
```

```
;-----
start:
```

```
mov ah,09h
mov dx,hello ;Hello out'
int 21h
```

```
mov ax, [a]
cmp ax, [b] ; a-b
```

```
jg rezmores
jl rezlitle
je rezzero
```

```
rezmores:
mov ah,09h
mov dx,amoresb
int 21h
jmp beforequit
```

```
rezlitle:
mov ah,09h
mov dx,bmoresa
int 21h
jmp beforequit
```

```
rezzero:
mov ah,09h
mov dx,aeqb
int 21h
```

```
beforequit:
mov ah,09h
mov dx,exitres
int 21h
mov ah,01h
int 21h
```

```
exit:
mov ax,4C00h
int 21h ; End
```

[\[Ответить\]](#)

[Борис](#)

24-12-2010 17:11

а разобрался \$ забыл

[\[Ответить\]](#)

[xrnd](#)

24-12-2010 20:00

Да, без баксов дос ничего не выводит 😊

< b превращалось в HTML код в комментариях.

Программа написана верно.

Можно ещё убрать одну команду перехода:

```
jg rezmore
```

Либо больше, либо равно, либо меньше. Всего 3 варианта. Для двух — команды переходов. Для третьего — выполнение программы продолжается без перехода.

[\[Ответить\]](#)

Гость

15-01-2011 17:07

```
use16
```

```
org 100h
```

```
jmp start
```

```
;
```

```
x db 'rezultat sravhehnia a b' ,13,10,'$'
```

```
a db 'a<b $'
```

```
b db 'a>b $'
```

```
c db 'a=b $'
```

```
a1 db -5
```

```
b1 db -5
```

```
;
```

```
start:
```

```
mov ah,09h
```

```
mov dx,x
```

```
int 21h
```

```
mov bl,[a1]
```

```
cmp bl,[b1]
```

```
jl amb
```

```
cmp bl,[b1]
```

```
jg abob
```

```
mov dx,c ;a=b вы вести строку c
```

```
int 21h
```

```
jmp exit ;выход из програмы
```

```
abob:
```

```
mov dx,b
```

```
int 21h
```

```
jmp exit
```

```
amb:
```

```
mov dx,a
```

```
int 21h
```



```
jmp exit
exit:
mov ah,08h ;Функция DOS 08h — ввод символа без эха
int 21h ;Обращение к функции DOS
mov ax,4C00h
int 21h
```

[\[Ответить\]](#)

[xrnd](#)

18-01-2011 21:30

Я немного подправил 😊

Движок сайта плохо обрабатывает знаки больше и меньше, думает что это HTML.

Программа написана хорошо и правильно.

Можно убрать вторую команду

```
cmp b1,[b1]
```

Команды условных переходов не изменяют флаги, а только анализируют, поэтому достаточно выполнить сравнение один раз для нескольких переходов.

[\[Ответить\]](#)

123123

12-01-2013 16:42

```
alert(«hello world»)
```

[\[Ответить\]](#)

Гость

26-01-2011 17:28

Привет , подскажи какие операции могут изменять флаг S ?

Я пробовал не деление не умножение ,не сложение не вычитание его не трогают.

[\[Ответить\]](#)

[xrnd](#)

26-01-2011 22:52

Скорее всего, у тебя получался положительный результат.

Флаг S — это флаг знака результата. Он равен знаковому биту результата.

```
xor ax,ax
dec ax ;ax = 0FFFFh, SF=1
```

[\[Ответить\]](#)

Гость

27-01-2011 22:26

Ясна , делить на отрицательное число невозможно , а складывать вычитать можно
и результат корректный и SF=1 меняется , деление спутало ,)

[\[Ответить\]](#)

Knight212

09-02-2011 23:35

```
use16
org 100h
jmp start
```

```
a db -65
b db -65
s1 db 'a>b$'
s2 db 'a<b$'
s3 db 'a=b$'
press db 0Dh, 0Ah, 'Press any key...$'
```

```
start:
mov al, [a]
cmp al, [b]
jg c1
jl c2
```

```
mov ah, 09h
mov dx, s3
int 21h
jmp exit
c1:
```

```
mov ah, 09h
mov dx, s1
int 21h
jmp exit
```

```
c2:
mov ah, 09h
mov dx, s2
int 21h
jmp exit
```

```
exit:
mov ah, 09h
mov dx, press
int 21h
```

```
mov ah, 08h
int 21h
```

```
mov ax, 4C00h
int 21h
```

[\[Ответить\]](#)

[xrnd](#)

10-02-2011 15:28

Программа написана без ошибок, хотя её можно сделать короче.

Во-первых, в этом коде переход делать не обязательно:

```
    jmp exit
exit:
```

Следующая команда и так выполнится по порядку.

Во-вторых, в любом случае выполняется команда «mov ah,09h» и команда «int 21h». Различие только в адресе строки в DX. Можно «mov ah,09h» сделать в начале, а «int 21h» после метки exit.

```
...
cmp al, [b]
mov ah,09h ;Команда не изменяет флаги
jg c1
jl c2
mov dx,s3
jmp exit
c1:
mov dx,s1
jmp exit
c2:
mov dx,s2
exit:
int 21h
...
```

[\[Ответить\]](#)

Knight212

10-02-2011 18:59

Спасибо.

[\[Ответить\]](#)

A13R42

28-02-2011 18:43

Привет!

```
;_____
use16
org 0x100
```

```
jmp start
;_____
x db -4
y db 4
a db 'x>y',13,10,'$'
b db 'x=y',13,10,'$'
c db 'x<y',13,10,'$'
```

```
exit db 'Press any key...$'
```

```
;
```

```
start:
```

```
mov ah,0x09
```

```
mov bh,[x]
```

```
mov bl,[y]
```

```
cmp bh,bl
```

```
jl xmy
```

```
jg xby
```

```
mov dx,b
```

```
int 0x21
```

```
jmp exit1
```

```
xmy:
```

```
mov dx,c
```

```
int 0x21
```

```
jmp exit1
```

```
xby:
```

```
mov dx,a
```

```
int 0x21
```

```
jmp exit1
```

```
exit1:
```

```
mov dx,exit
```

```
int 0x21
```

```
mov ah,0x08
```

```
int 0x21
```

```
mov ax, 0x4c00
```

```
int 0x21
```

[\[Ответить\]](#)

A13R42

28-02-2011 22:17

```
xby:
```

```
mov dx,a
```

```
int 0x21
```

```
jmp exit1 ;тут jump, в принципе, не нужен, хотя он повышает читаемость и структурность
```

[\[Ответить\]](#)

[xrnd](#)

01-03-2011 16:00

Написано всё правильно, хотя можно немного оптимизировать.

Последний `jmp exit1` лучше убрать. Он увеличивает время выполнения кода и размер программы. Для читаемости можно добавить комментарий 😊

```
mov bl,[y]
cmp bh,bl
```

Загружать у в регистр не обязательно.

```
cmp bh,[y]
```

Ещё можно int 0x21 разместить сразу после метки exit1. Так как во всех случаях эта команда выполняется. Вместо трёх команд будет одна.

[\[Ответить\]](#)

A13R42

01-03-2011 16:02

Хорошо, учту, спасибо!

[\[Ответить\]](#)

Shov

30-03-2011 23:07

```
use16
org 100h
jmp start
;_____
i_a db 13,10,'Enter A:$'
i_b db 13,10,'Enter B:$'
abb db 13,10,'a > b$'
arb db 13,10,'a = b$'
amb db 13,10,'a < b$'
a db ?
b db ?
;_____
start:
mov dx,i_a
mov ah,09h
int 21h
get_a:
mov ah,01h
int 21h
cmp al,13
je get_b_t
add [a],al
jmp get_a
get_b_t:
mov dx,i_b
mov ah,09h
int 21h
get_b:
mov ah,01h
int 21h
cmp al,13
je analiz
add [b],al
jmp get_b
```

```
analiz:  
mov bh,[a]  
mov bl,[b]  
cmp bh,bl
```

```
je ravno  
jl menshe  
jmp bolshe  
ravno:  
mov dx,arb  
mov ah,09h  
int 21h  
jmp endd  
menshe:  
mov dx,amb  
mov ah,09h  
int 21h  
jmp endd  
bolshe:  
mov dx,abb  
mov ah,09h  
int 21h
```

```
endd:  
mov ah,08h  
int 21h  
mov ax,4C00h  
int 21h
```

[\[Ответить\]](#)

[xrnd](#)

01-04-2011 15:48

Интересный способ ввода переменных.

Вообще всё верно, но можно сделать программу короче, если перенести

```
mov ah,09h  
int 21h
```

после метки endd.

[\[Ответить\]](#)

annihilator

27-05-2011 11:47

пытался организовать ввод чисел с клавиатуры, но оказалось что функция DOS 01h позволяет ввести лишь один символ, всё получилось но только с одним символом (цифра). Видел в следующих уроках есть ввод строки но не стал заморачиваться, всему своё время, успею ещё...

Поэтому сделал по-другому:

```

use16 ;Генерировать 16-битный код
org 100h ;Программа начинается с адреса 100h
jmp start ;Безусловный переход на метку start
;-----
a dw -128
b dw 127
a_more_b db 13,10,'a>b',13,10,'$'
b_more_a db 13,10,'a<b',13,10,'$'
a_equal_b db 13,10,'a=b',13,10,'$'
err_msg db 'Error: overflow detected.',13,10,'$'
pak db 'Press any key...$'
;-----
start:

```

```

cmp [a], -128 ;поскольку число со знаком, то в AL мин число -128(-128..127)
jl error ;переход к обработке ошибки если переполнение
cmp [a], 127 ;поскольку число со знаком, то в AL макс число 127(-128..127)
jg error ;переход к обработке ошибки если переполнение
mov al, byte[a]

```

```

cmp [b], -128 ;поскольку число со знаком, то в AL мин число -128(-128..127)
jl error ;переход к обработке ошибки если переполнение
cmp [b], 127 ;поскольку число со знаком, то в AL макс число 127(-128..127)
jg error ;переход к обработке ошибки если переполнение

```

```

cmp al, byte[b] ;сравнение a и b
jg lb1 ;Переход, a>b
jl lb2 ;Переход, a<b
mov ah, 09h
mov dx, a_equal_b ;Вывод строки 'a=b'
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
lb1:
mov ah, 09h ;\
mov dx, a_more_b ; > Вывод строки 'a>b'
int 21h ;/
jmp exit ;Безусловный переход
lb2:
mov ah, 09h ;\

```

```
mov dx,b_more_a ; > Вывод строки 'a<b';  
int 21h ;/  
jmp exit ;Безусловный переход
```

Можно ещё как-то улучшить эту программу?

[\[Ответить\]](#)

[xrnd](#)

23-06-2011 15:39

Тут есть глюк с символом '<' в комментариях 😊

В начале проверки на «переполнение» не имеют смысла. Байт никак не может быть меньше -128 или больше 127. К тому же ты проверяешь 16-битные переменные, а потом всё равно сравниваешь только их младшие байты. Лучше сразу их объявить как байты а и b.

Ещё можно немного оптимизировать, если команды «mov ah,09h» и «int 21h» поместить в общей части программы. У тебя 3 варианта выбора и в каждом варианте эти команды повторяются. Различие только в адресе строки в DX.

Можно сделать так:

```
...  
cmp al,byte[b]  
jg lb1  
jl lb2  
mov dx,a_equal_b  
jmp out_str  
lb1:  
mov dx,a_more_b  
jmp out_str  
lb2:  
mov dx,b_more_a  
out_str:  
mov ah,09h  
int 21h ; Вывод нужной строки  
exit:  
...
```

[\[Ответить\]](#)

annihilator

23-06-2011 23:29

Спасибо. Насчёт чисел, это просто осталось от попыток написать программу со вводом чисел через функцию дос с клавиатуры (это для случая когда введенное число окажется больше чем может сравнить программа). Я это оставил «на потом»)))

[\[Ответить\]](#)

Daedalus

30-08-2011 00:44

Привет у меня вопрос возник по поводу флага OF.

Решил потестить , и совсем запутался ,)

Вот пример


```
mov al,192
mov ah,192
; cF,sF=1 ; перенос , отрицательный результат
```

```
add al,ah ; al = 384 1 1000 0000
```

;А если

```
mov al,191
mov ah,192
; cF,oF=1 ; перенос , переполнение
```

```
add al,ah ; al = 383 1 0111 1111
```

Старший байт , в одном случае 0 , а в другом 1 ,)
Это в принципе поясняет почему флаг sF то равен 1 то нет .
Но понять почему oF , то равен то нет я не как не могу.
То есть если число 1 1 — — , то флаг oF = 0 а sF=1
И наоборот
1 0 — — , то флаг oF = 1 а sF=0
Просто флаг oF , какойто хитрый оказался.
Или я чегота не допонял.

[\[Ответить\]](#)

Daedalus
30-08-2011 13:04

Привет , правильно я понял .
Почему так происходит и за границы результата(байт) он должен уместиться от -127 до 128 .
То есть от (255..129) — воспринимаются всегда как отрицательные , (1 ..128)
воспринимаются всегда как положительные.
И этот хитрый алгоритм не видит нечего страшного в том чтобы 255+255 , так как для него
это -1+-1 =-2 , что вполне в деапозоне , а отражает это только флаг oF
Поэтому тут cF=1 , sF=1 , oF = 0.
Это говорит о том, что если число , со знаком то результат получился корректным
и флаг cF=1 , можно проигнореть.

А вот при попытке 64+65 — результат выходит за деопозон значений
cF=1 , sF=0 , oF = 1.
А это говорит атом что , если число со знаком то результат вышел , не корректный.

В общем я прешёл к выводу что
cF , используется для чисел без знака , для анализа корректности результата.
sF , oF для чисел со знаком , флаги в заимо исключают друг друга.
Хотелось бы узнать такли это ?

[\[Ответить\]](#)

[xrnd](#)
17-09-2011 17:25

Именно так. Для чисел без знака переполнение определяется по флагу CF.
Если произошёл перенос из старшего разряда, значит результат не влез в разумные пределы.

$192+192 = 384$ CF=1 значит переполнение, результат не влез в байт.

Если записать эти числа как числа со знаком, то получится:

$-64+(-64)=-128$ результат отрицательный, но переполнения нет. SF=1 OF=0

[\[Ответить\]](#)

алекс

12-03-2012 20:53

use16

org 100h

jmp start ;безусловный переход на начало

lowe db 'ab\$',13,10

equal db 'a=b\$',13,10

press db 13,10,'Press any key...\$'

a dw 230 ;описание переменных

b dw 230

start: ;начало программы

mov ax,[a] ;помещаем в ax значение a

cmp ax,[b] ;сравниваем ax с b

je eu ;если равно переходим на метку eu

cmp ax,[b] ;сравниваем ax с b

js lw ;если ax меньше переходим на lw

mov ah,09h ;здесь выполняем вывод строки

mov dx,hige ;hige, если ax не равно b и не

int 21h ;меньше b, то есть $a > b$

jmp exit ;переход на метку exit

eu: ;метка eu

mov ah,09h ;здесь выполняем вывод строки

mov dx,equal ;equal, когда $a = b$

int 21h

;переход на метку exit

jmp exit

lw: ;метка lw

mov ah,09h ;здесь выполняем вывод строки

mov dx,lowe ;lowe, когда $a < b$

int 21h

;метка exit, завершение программы

exit:

mov ah,09h ;вывод сообщения press

mov dx,press

int 21h

mov ah,08h ;ввод символа

int 21h

mov ax,4c00h ;завершение программы
int 21h

[\[Ответить\]](#)

andrew.NET
10-06-2012 12:17

use16
org 0x100

mov ah, [a]
mov al, [b]
cmp ah, al
jl cp1
jg cp2
je cp3

cp1:
mov ah, 09
mov dx, less
jmp exit

cp2:
mov ah, 09
mov dx, more
jmp exit

cp3:
mov ah, 09
mov dx, rav
jmp exit

exit:
int 0x21
mov ah, 0x08
int 0x21
mov ax, 0x4C00
int 0x21

a db -5
b db -5
less db «a b\$»
rav db «a = b\$»

[\[Ответить\]](#)

алексей
15-10-2012 17:15

Ребят помогите с условными переходами 3,4,8
умр метка умр адрес безусловный переход
sub...

СМР ПР.,ИСТЗ ЗНАКА РАВНО ПР-ИСТ =НЕ РАВНО ПР.=ПР-ИСТ

JZ если не 0 пр.=ист
JNZ если не 0 прист
JC есть перенос пр=ист
JS есть знак пр=0

[\[Ответить\]](#)

алексей
15-10-2012 17:23

Есть блок схема

[\[Ответить\]](#)

Денис
02-12-2012 15:14

```
use16
org 100h
jmp start
a db 15
b db 15
rav db 'a=b$'
bol db 'a>b$'
men db 'a<b$'
start:
mov al,[a]
sub al,[b]
jz nol
jnz ne_nol
```

```
nol:
mov ah,09h
mov dx, rav
int 21h
```

```
jmp exit
ne_nol:
ja bolshe
jna menshe
bolshe:
mov ah,09h
mov dx,bol
int 21h
jmp exit
menshe:
mov ah,09h
mov dx,men
int 21h
jmp exit
exit:
mov ah,08h
int 21h
```

```
mov ax,4c00h
int 21h
```

[\[Ответить\]](#)

Денис
02-12-2012 15:15

ой, косяк с табуляцией

[\[Ответить\]](#)

enlacsys
07-12-2012 06:21

```
use16
;;-----
Label Python style
;;-----
org 100h

jmp __init__ ; go start label
;-----
a dw -1
b dw 10

eq_str db «a=b», '$'
lt_str db «ab», '$'
;-----
__init__::programm start here
xor bx,bx
mov bx,[a]
cmp bx,[b]

jg __gt__
jl __lt__
je __eq__

__gt__::great than
mov ah,0x9
mov dx, gt_str
int 0x21
jmp __del__

__lt__::less than
mov ah,0x9
mov dx, lt_str
int 0x21
jmp __del__

__eq__::equal
mov ah,0x9
mov dx, eq_str
int 0x21
jmp __del__
```

```
__del__::exit programm  
mov ax,0x4c00  
int 0x21
```

[\[Ответить\]](#)

REDPROJRAMMER
03-07-2013 22:33

Кроме команды CMP есть конструкция ветвления, вот я не понимаю как ее использовать, она находится в include\macro\if.inc. кроме конструкции ветвления есть команды until и repeat. Вот пытаюсь ее использовать, а пока пользуюсь cmp

[\[Ответить\]](#)

vitta116rus
18-12-2015 22:52

Такой вопрос, нужно чтобы выводило месяц по счету,когда вводишь цифру от 1 до 9 но почему-то он всегда выдает «сентябрь». что не так с моим кодом?

```
data segment  
mes1 db 10, 13, 'Jan$'  
mes2 db 10, 13, 'Fed$'  
mes3 db 10, 13, 'Mar$'  
mes4 db 10, 13, 'Apr$'  
mes5 db 10, 13, 'May$'  
mes6 db 10, 13, 'Jun$'  
mes7 db 10, 13, 'Jul$'  
mes8 db 10, 13, 'aug$'  
mes9 db 10, 13, 'sep$'  
data ends
```

```
code segment  
start:  
assume cs:code, ds: data  
mov ax, data  
mov ds, ax  
mov ah, 01  
int 21h  
cmp al, '1'  
je .ibone  
.ibone:  
lea dx,mes1
```

```
cmp al, '2'  
je .ibtwo  
.ibtwo:  
lea dx, mes2
```

```
cmp al, '3'  
je .ibthree  
.ibthree:  
lea dx, mes3
```

```
cmp al, '4'  
je .ibfour  
.ibfour:  
lea dx, mes4
```

```
cmp al, '5'  
je .ibfive  
.ibfive:  
lea dx, mes5
```

```
cmp al, '6'  
je .ibsix  
.ibsix:  
lea dx, mes6
```

```
cmp al, '7'  
je .ibseven  
.ibseven:  
lea dx, mes7
```

```
cmp al, '8'  
je .ibeight  
.ibeight:  
lea dx, mes8
```

```
cmp al, '9'  
je .ibnine  
.ibnine:  
lea dx, mes9  
jmp vivod
```

```
vivod:  
mov ah, 09  
int 21h
```

```
mov ax, 4c00h  
int 21h  
code ends  
end start
```

[\[Ответить\]](#)

Андрей
11-04-2017 20:18

Кто может составить блок схему ??

[\[Ответить\]](#)

mmd
19-05-2017 09:08

Добрый день, хотелось бы обратиться к вам со следующим вопросом. При написании программы довольно странно работает cmp. При сравнении выводит весь последующий текст

начиная с указанной метки не беря во внимание конец строки. Решила для интереса воспользоваться кодом предложенным в примере для вывода строк, но тоже самое. Подскажите, пожалуйста, в чем может быть причина?

[\[Ответить\]](#)

Rafael
17-08-2017 19:18

```
use16
org 100h
jmp start
;-----
a dw 2h
b dw -2h
menshe db 'a b$'
ravno db 'a = b$'
;-----
start:
mov ax, [a]
mov bx, [b]
cmp ax, bx
jg bo
jl me
jz ra
bo:
mov dx, bolshe
jmp s
me:
mov dx, menshe
jmp s
ra:
mov dx, ravno
jmp s
s:
mov ax, 0900h
int 21h
mov ax, 0800h
int 21h
jmp exit
exit:
mov ax, 4C00h
int 21h
```

[\[Ответить\]](#)

Rafael
17-08-2017 19:21

```
;-----
a dw 2h
b dw -2h
menshe db 'a b$'
```


ravno db 'a = b\$'

;_____

[\[Ответить\]](#)

Илья

18-12-2017 01:47

как вернуться на место откуда был совершен переход

[\[Ответить\]](#)

Ваш комментарий

Имя *

Почта (скрыта) *

Сайт

Добавить

☐ Уведомить меня о новых комментариях по email.

☐ Уведомлять меня о новых записях почтой.