**Лекция А5**
**БГТУ, ФИТ, ПОИТ, 2 семестр, Языки программирования**
**Введение в язык Ассемблер**

## 1. Регистр флагов EFLAGS:

- ✓ инструкции процессора или специальные команды устанавливают флаги регистра **EFLAGS**;
- ✓ непосредственно регистр не доступен программисту;
- ✓ програмист может проверять состояние флагов.

**Флаг установлен:**
значение соответствующего ему бита регистра EFLAGS равно 1.

**Флаг сброшен:**
значение соответствующего ему бита регистра EFLAGS равно 0.

Регистр флагов **EFLAGS –** это 32-разрядный регистр.
Старшие 16 разрядов используются при работе в защищённом режиме, и мы их рассматривать не будем.
К младшим 16 разрядам этого регистра можно обращаться как к отдельному регистру с именем FLAGS.
Все неиспользуемые биты помечены серым цветом и равны нулю за исключением 1-го бита, который всегда равен единице.

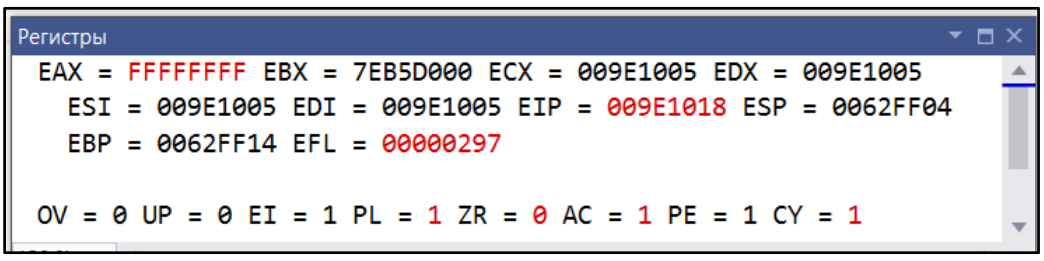
Флаги, находящиеся в младших 16 разрядах регистра **EFLAGS**:

| 15 | | | | | | | | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | 0 | | 0 | | 1 | |
| | NT | IOPL | | OF | DF | IF | TF | SF | ZF | | AF | | PF | | CF |
| x | x | x | | s | c | x | x | s | s | | s | | s | | s |

s – состояние; x – системный; c – управляющий.

| Бит | Обозначение Intel | Название | Описание | Обозначение в окне Registers VS |
|---|---|---|---|---|
| 0 | **CF** | Carry Flag | **Флаг переноса.**<br>Устанавливается в 1, когда арифметическая операция генерирует перенос или выход за разрядную сетку результата.<br>Сбрасывается в 0 в противном случае. | **CY** |
| 1 | 1 | | **Зарезервирован** | |
| 2 | **PF** | Parity Flag | **Флаг чётности.**<br>Устанавливается в 1, если результат последней операции имеет четное число единиц. | **PE** |
| 3 | 0 | | **Зарезервирован** | |
| 4 | **AF** | Auxiliary Carry Flag | **Вспомогательный флаг переноса**.<br>Устанавливается в 1, если арифметическая операция генерирует перенос из 3 бита в 4.<br>Сбрасывается в 0 в противном случае. Этот флаг используется в двоично-десятичной арифметике. | **AC** |
| 5 | 0 | | **Зарезервирован** | |
| 6 | **ZF** | Zero Flag | **Флаг нуля.**<br>Устанавливается в 1, если результат нулевой.<br>Сбрасывается в 0 в противном случае. | **ZR** |
| 7 | **SF** | Sign Flag | **Флаг знака.**<br>Устанавливается равным старшему биту результата, который определяет знак в знаковых целочисленных операциях (0 – положительное число, 1 – отрицательное число). | **PL** |
| 8 | **TF** | Trap Flag | **Флаг трассировки** (пошаговое выполнение). | |
| 9 | **IF** | Interrupt Enable Flag | **Флаг разрешения прерываний.**<br>При значении 1 микропроцессор реагирует на внешние аппаратные прерывания. | **El** |
| 10 | **DF** | Direction Flag | **Флаг направления.** | **UP** |
| 11 | **OF** | Overflow Flag | **Флаг переполнения.**<br>Устанавливается в 1, если целочисленный результат слишком длинный для размещения в целевом операнде. Этот флаг показывает наличие переполнения в знаковой целочисленной арифметике. | **OV** |
| 12 | **IOPL** | I/O Privilege Level | **Уровень приоритета ввода-вывода.** | |
| 13 | | | | |
| 14 | **NT** | Nested Task | **Флаг вложенности задач.** | |
| 15 | 0 | | **Зарезервирован** | |

## Окно регистров отладчика VS:



## 2. Команды условного перехода:

Синтаксис команды условного перехода:

| J<условие> | <метка_перехода> |
|---|---|

Команда условного перехода передает управление по указанной метке, если установлен соответствующий флаг состояния процессора.

Если флаг сброшен, то выполняется следующая за ней команда.

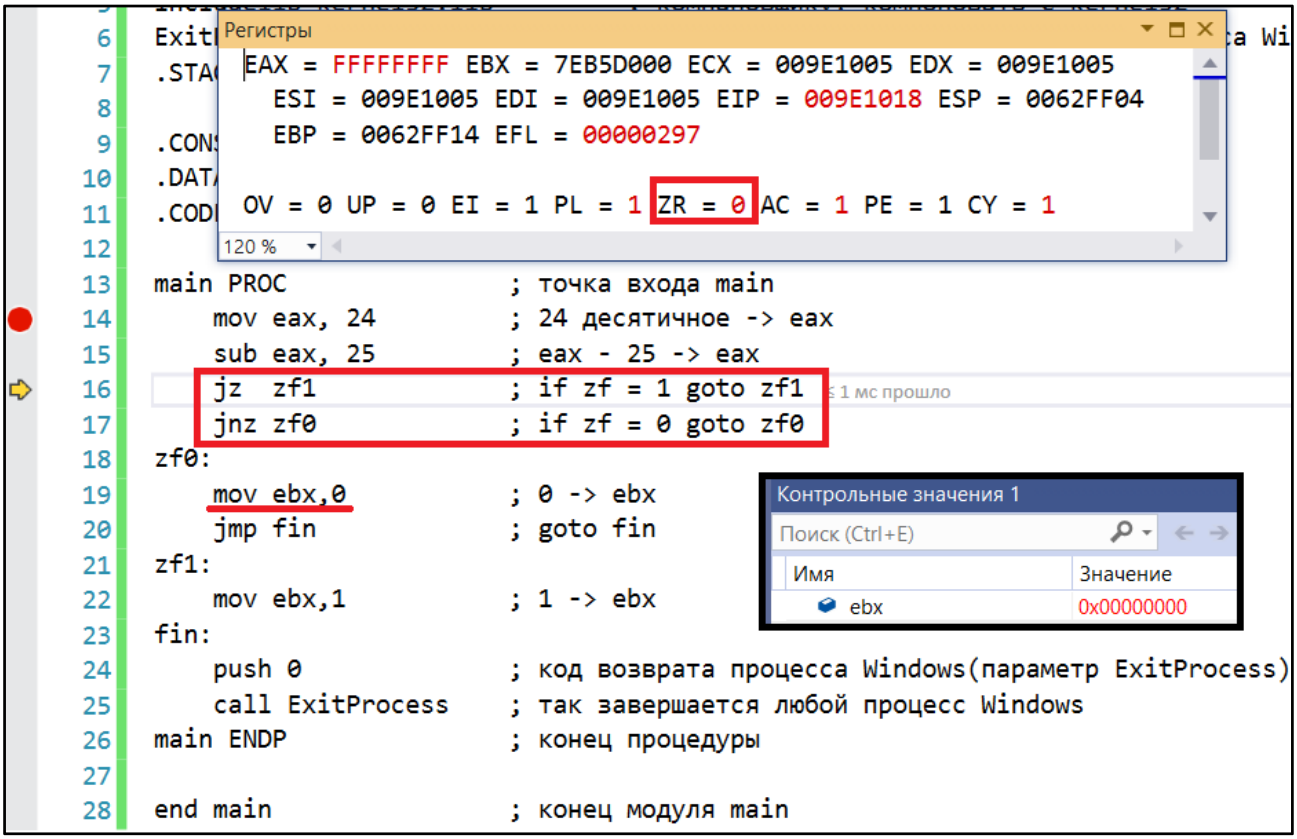
### *2.1 Команды перехода в зависимости от значения флагов состояния процессора:*

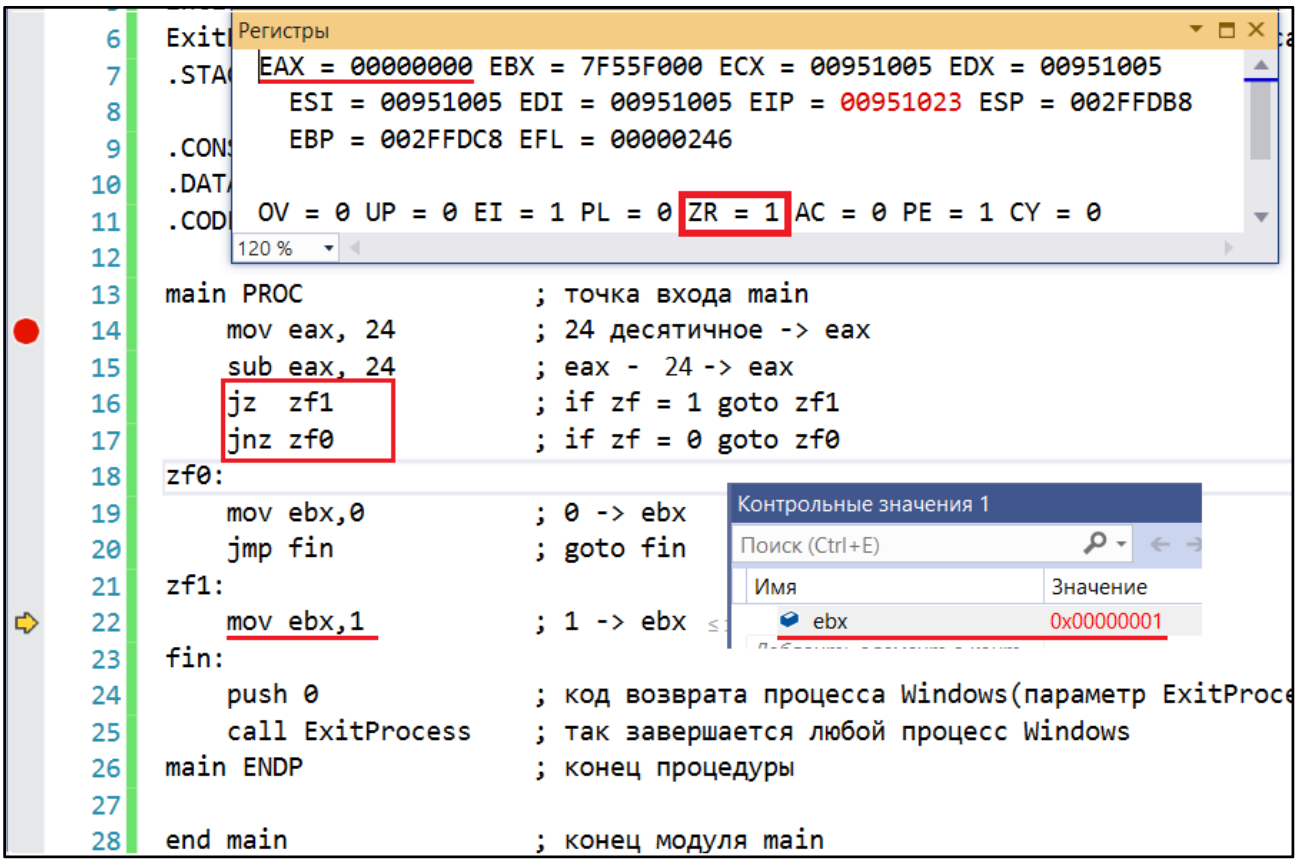
**Флаг нуля:**

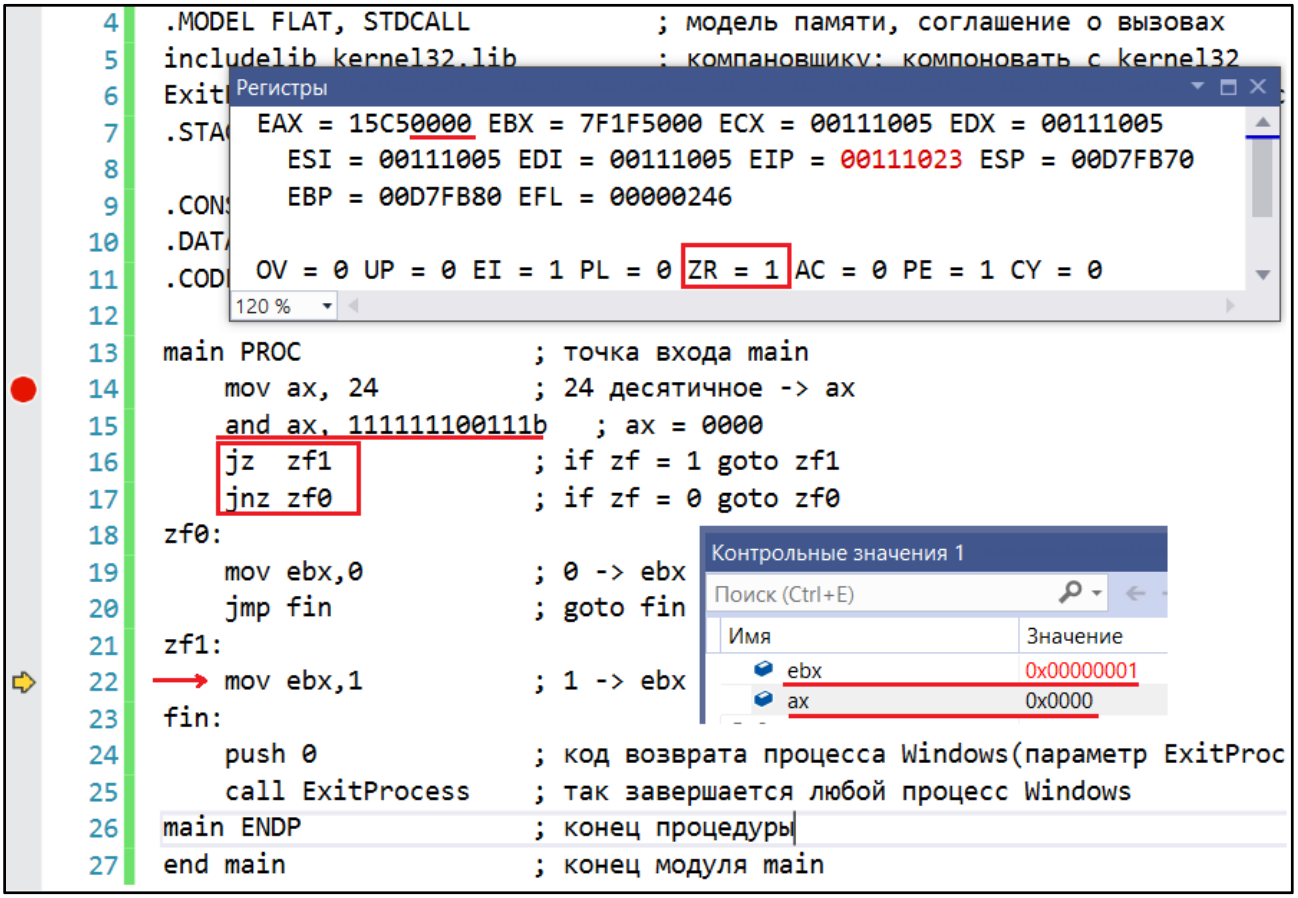
ZF (Zero Flag) Устанавливается в 1, если результат нулевой. **(ZR)**
Сбрасывается в 0 в противном случае.

| Команда | Описание | Состояние флага |
|---|---|---|
| JZ | Переход по метке, если флаг нуля установлен | ZF=1 |
| JNZ | Переход по метке, если флаг нуля сброшен | ZF=0 |

В противном случае выполняется команда, следующая за этой.




Регистры
EAX = 00000000 EBX = 7F55F000 ECX = 00951005 EDX = 00951005
ESI = 00951005 EDI = 00951005 EIP = 00951023 ESP = 002FFDB8
EBP = 002FFDC8 EFL = 00000246
OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
120 %
main PROC ; точка входа main
mov eax, 24 ; 24 десятичное -> eax
sub eax, 24 ; eax - 24 -> eax
jz zf1 ; if zf = 1 goto zf1
jnz zf0 ; if zf = 0 goto zf0
zf0:
mov ebx,0 ; 0 -> ebx
jmp fin ; goto fin
zf1:
mov ebx,1 ; 1 -> ebx
fin:
push 0 ; код возврата процесса Windows(параметр ExitProce
call ExitProcess ; так завершается любой процесс Windows
main ENDP ; конец процедуры
end main ; конец модуля main
Контрольные значения 1
Поиск (Ctrl+E)
Имя
Значение
ebx
0x00000001

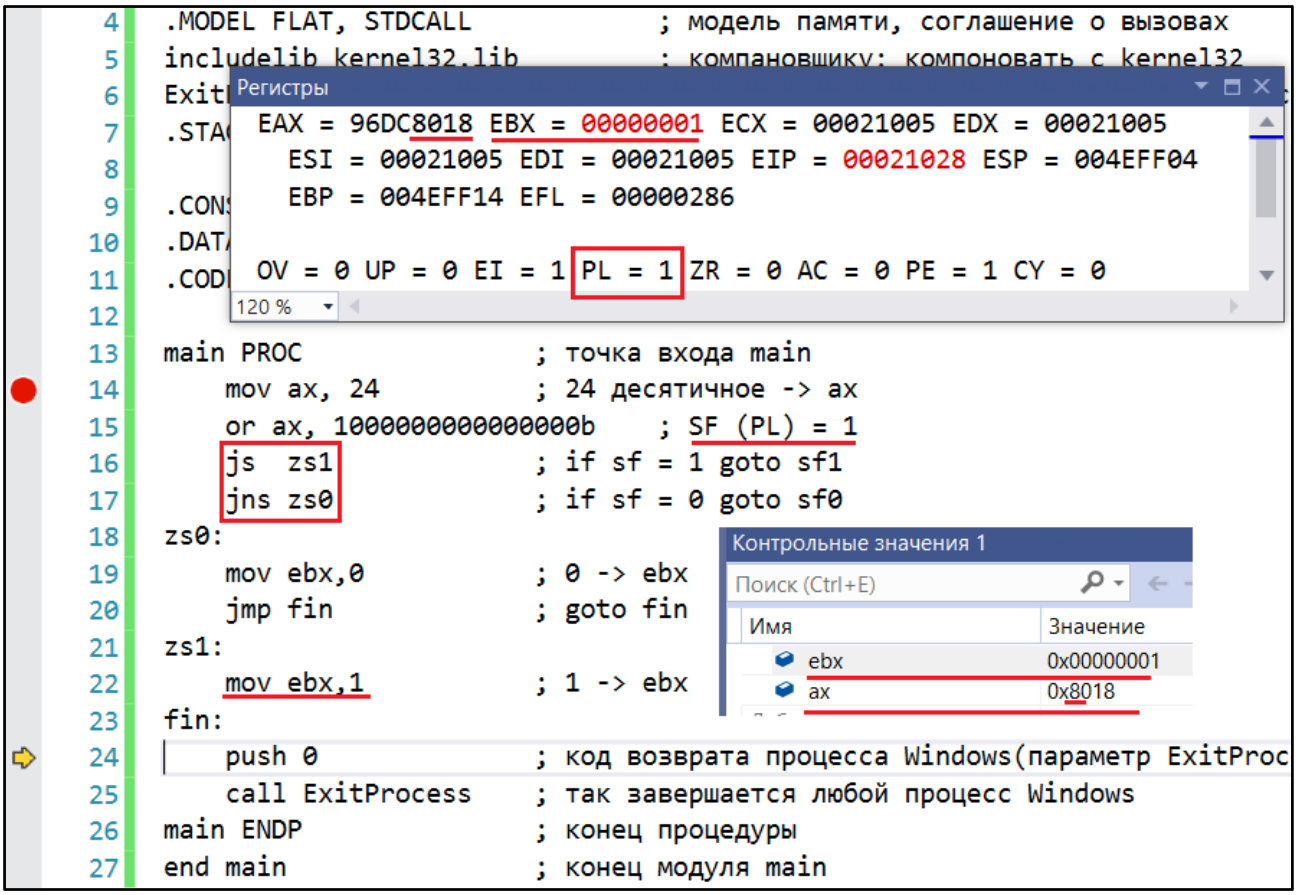

.MODEL FLAT, STDCALL ; модель памяти, соглашение о вызовах
Регистры
EAX = 15C50000 EBX = 7F1F5000 ECX = 00111005 EDX = 00111005
ESI = 00111005 EDI = 00111005 EIP = 00111023 ESP = 00D7FB70
EBP = 00D7FB80 EFL = 00000246
OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
120 %
main PROC ; точка входа main
mov ax, 24 ; 24 десятичное -> ax
and ax, 111111100111b ; ax = 0000
jz zf1 ; if zf = 1 goto zf1
jnz zf0 ; if zf = 0 goto zf0
zf0:
mov ebx,0 ; 0 -> ebx
jmp fin ; goto fin
zf1:
mov ebx,1 ; 1 -> ebx
fin:
push 0 ; код возврата процесса Windows(параметр ExitProc
call ExitProcess ; так завершается любой процесс Windows
main ENDP ; конец процедуры
end main ; конец модуля main
Контрольные значения 1
Поиск (Ctrl+E)
Имя
Значение
ebx
0x00000001
ax
0x0000

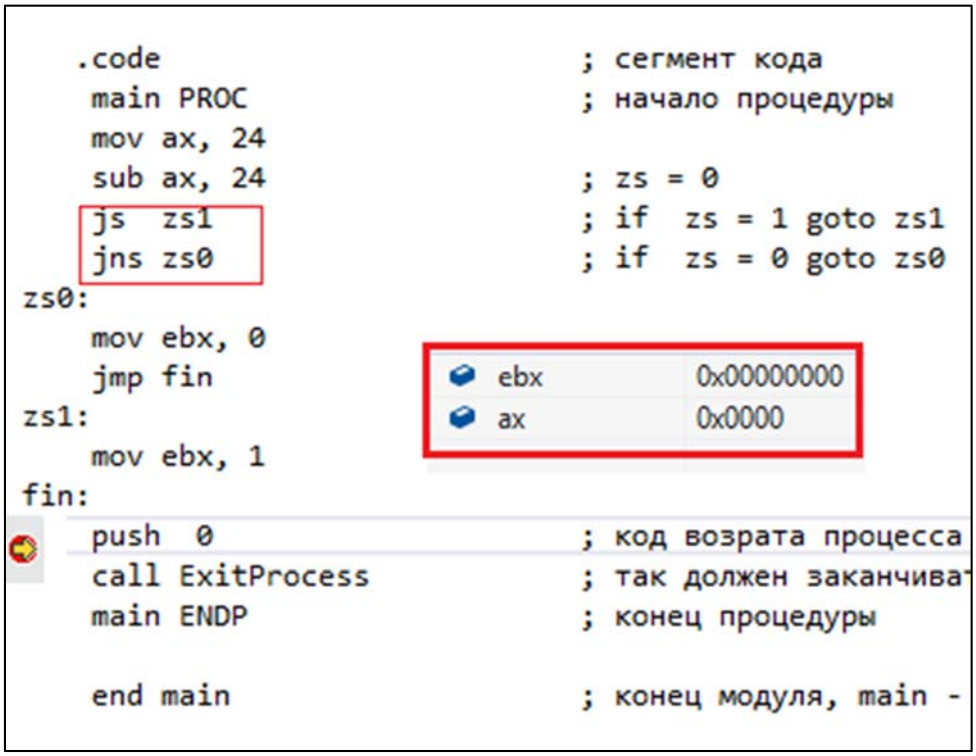
## *2.2Команды перехода в зависимости от значения флагов состояния процессора:*

**Флаг знака:**

SF (Sign Flag) (PL) Устанавливается равным старшему биту результата, который определяет знак в знаковых целочисленных операциях (0 – положительное число, 1 – отрицательное число).

| Команда | Описание | Состояние |
|---|---|---|
| JS | переход, если флаг знака установлен | SF=1 |
| JNS | переход, если флаг знака сброшен | SF =0 |



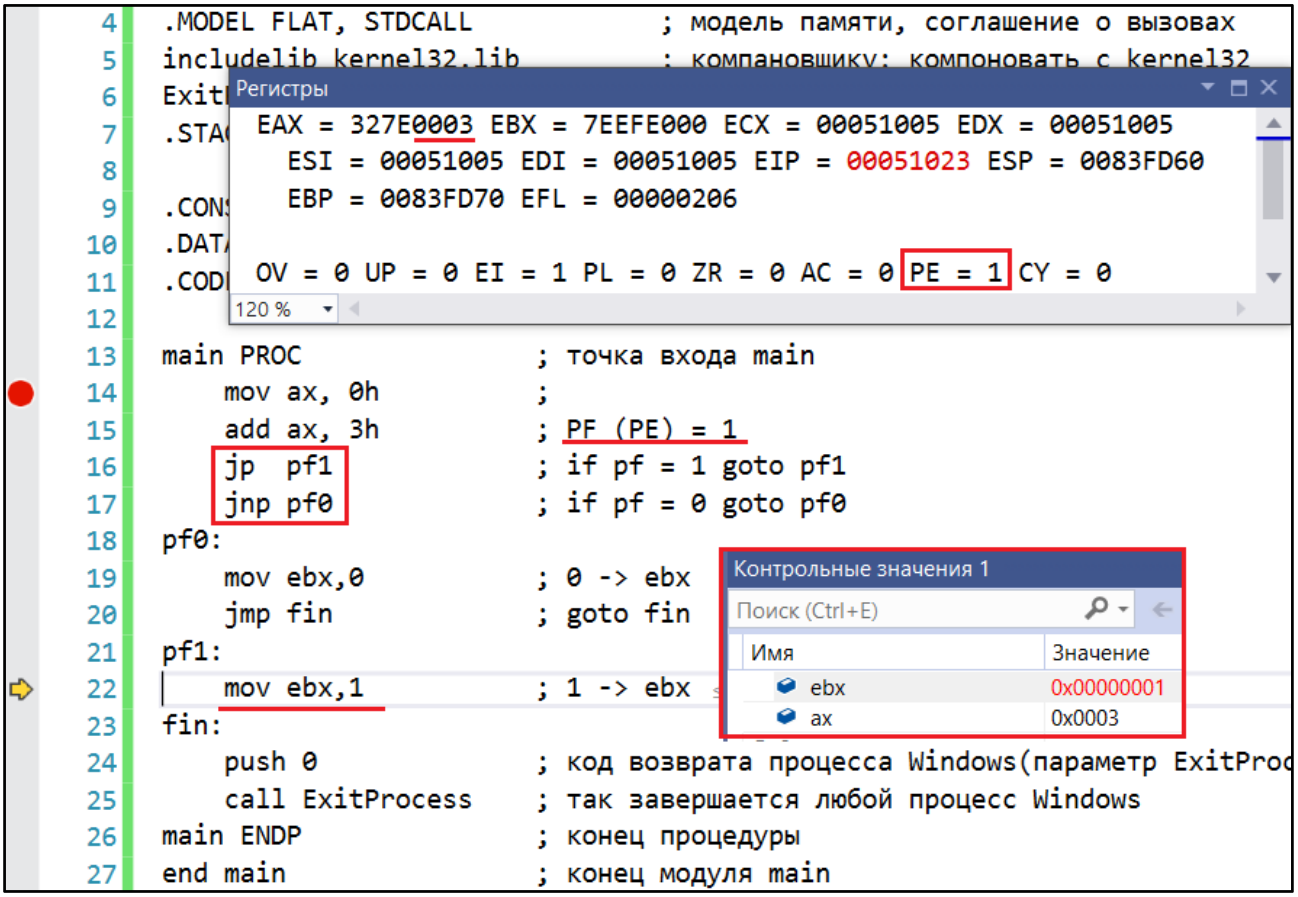

.code ; сегмент кода
main PROC ; начало процедуры
mov ax, 24
sub ax, 24 ; zs = 0
js zs1 ; if zs = 1 goto zs1
jns zs0 ; if zs = 0 goto zs0
zs0:
mov ebx, 0
jmp fin
zs1:
mov ebx, 1
fin:
push 0 ; код возрата процесса
call ExitProcess ; так должен заканчива
main ENDP ; конец процедуры
end main ; конец модуля, main -
ebx 0x00000000
ax 0x0000

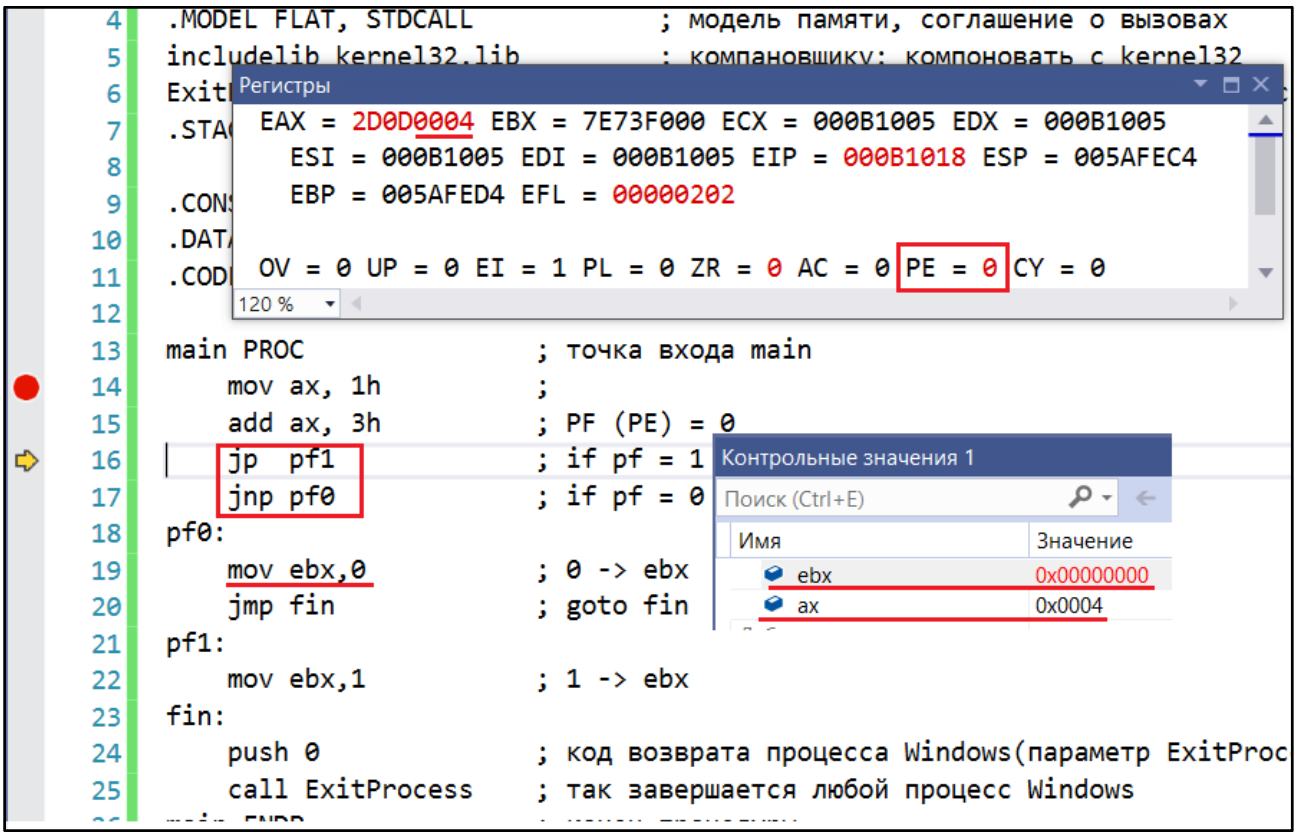
## *2.3 Команды перехода в зависимости от значения флагов состояния процессора:*

**Флаг чётности:**

PF (PE) (Parity Flag) Устанавливается в 1, если результат последней операции имеет четное число единиц.

| Команда | Описание | Состояние |
|---|---|---|
| JP | переход, если флаг четности установлен | PF=1 |
| JNP | переход, если флаг четности сброшен | PF =0 |



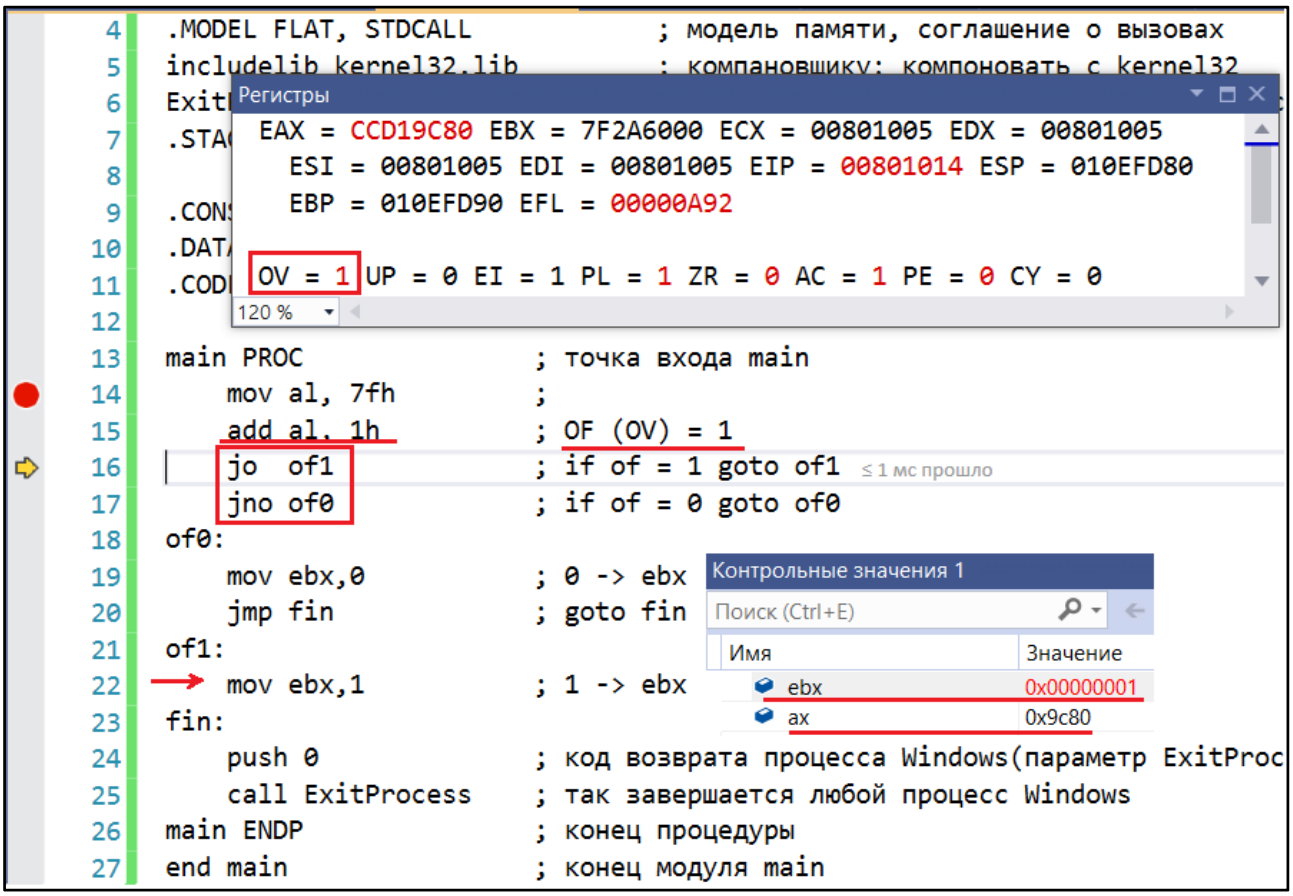

Регистры
EAX = 2D0D0004 EBX = 7E73F000 ECX = 000B1005 EDX = 000B1005
ESI = 000B1005 EDI = 000B1005 EIP = 000B1018 ESP = 005AFEC4
EBP = 005AFED4 EFL = 00000202
OV = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0 PE = 0 CY = 0
120 %
.MODEL FLAT, STDCALL ; модель памяти, соглашение о вызовах
includelib kernel32.lib ; компановщику: компоновать с kernel32
main PROC ; точка входа main
mov ax, 1h ;
add ax, 3h ; PF (PE) = 0
jp pf1 ; if pf = 1
jnp pf0 ; if pf = 0
pf0:
mov ebx,0 ; 0 -> ebx
jmp fin ; goto fin
pf1:
mov ebx,1 ; 1 -> ebx
fin:
push 0 ; код возврата процесса Windows(параметр ExitProc
call ExitProcess ; так завершается любой процесс Windows
Контрольные значения 1
Поиск (Ctrl+E)
Имя
Значение
ebx
0x00000000
ax
0x0004

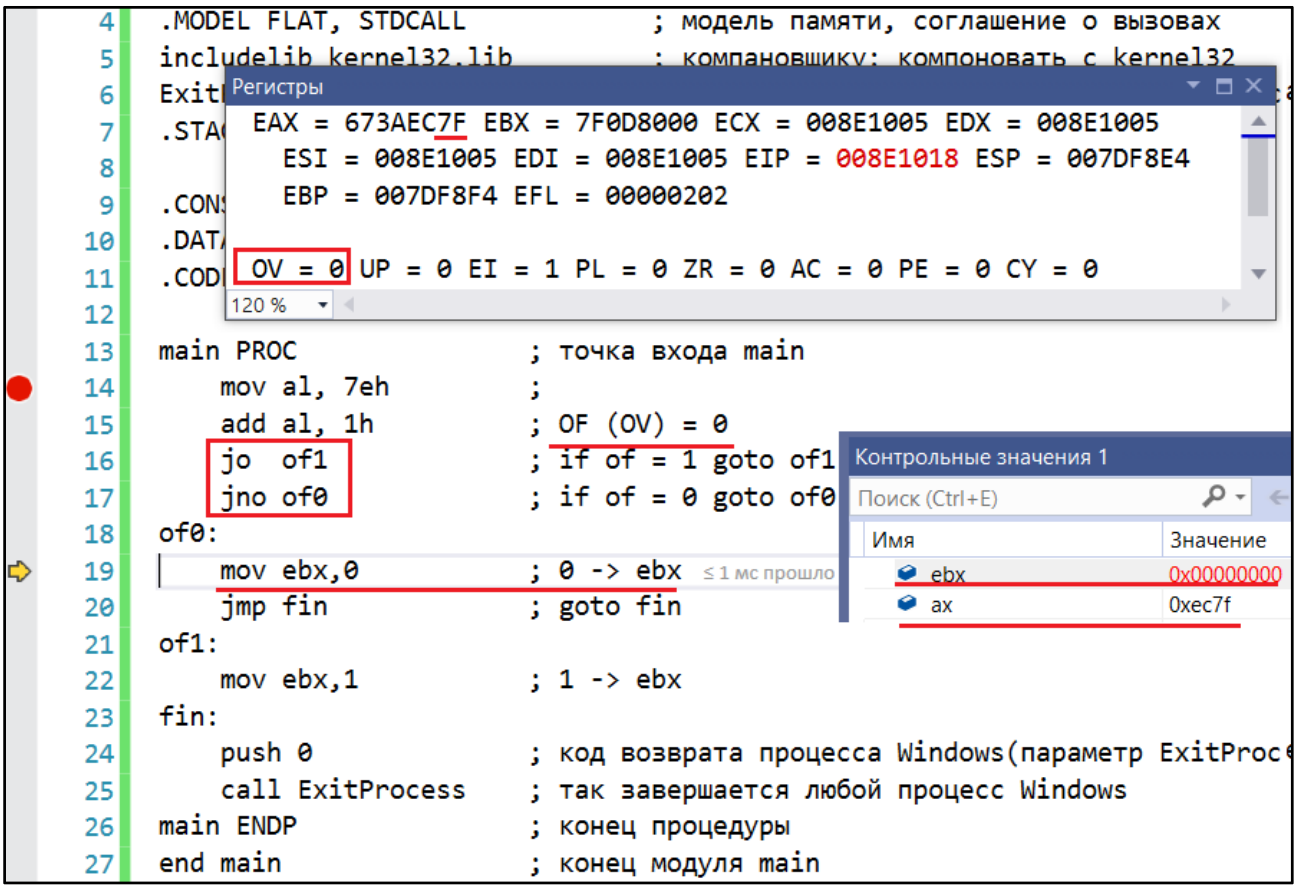
## *2.4Команды перехода в зависимости от значения флагов состояния процессора:*

**Флаг переполнения:**

OF (OV) (Overflow Flag) Устанавливается в 1, если целочисленный результат слишком длинный для размещения в целевом операнде. Этот флаг показывает наличие переполнения в знаковой целочисленной арифметике.

| Команда | Описание | Состояние |
|---|---|---|
| JO | переход, если возникло переполнение | OF=1 |
| JNO | переход, если переполнения нет | OF =0 |




Регистры
EAX = 673AEC7F EBX = 7F0D8000 ECX = 008E1005 EDX = 008E1005
ESI = 008E1005 EDI = 008E1005 EIP = 008E1018 ESP = 007DF8E4
EBP = 007DF8F4 EFL = 00000202
OV = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0 PE = 0 CY = 0
120 %
4 .MODEL FLAT, STDCALL ; модель памяти, соглашение о вызовах
5 includelib kernel32.lib ; компановщику: компоновать с kernel32
6 Exit
7 .STA
8
9 .CON
10 .DAT
11 .COD
12
13 main PROC ; точка входа main
14 mov al, 7eh ;
15 add al, 1h ; OF (OV) = 0
16 jo of1 ; if of = 1 goto of1
17 jno of0 ; if of = 0 goto of0
18 of0:
19 mov ebx,0 ; 0 -> ebx ≤ 1 мс прошло
20 jmp fin ; goto fin
21 of1:
22 mov ebx,1 ; 1 -> ebx
23 fin:
24 push 0 ; код возврата процесса Windows(параметр ExitProc
25 call ExitProcess ; так завершается любой процесс Windows
26 main ENDP ; конец процедуры
27 end main ; конец модуля main
Контрольные значения 1
Поиск (Ctrl+E)
Имя
Значение
ebx
0x00000000
ax
0xec7f

## *3. Команды сравнения*

### Комада TEST

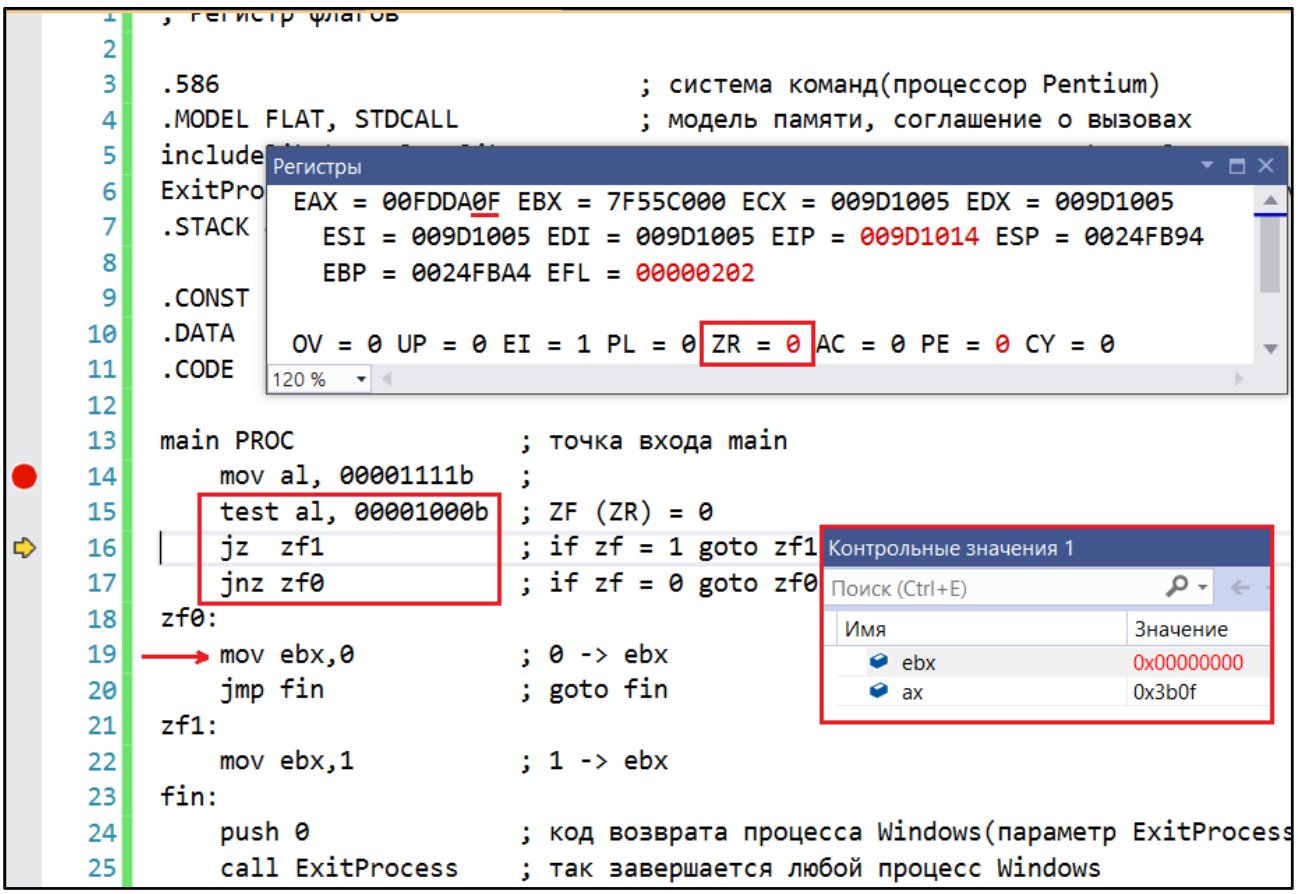
Выпоняет операцию поразрядного логического И между соответствующими парами битов двух операндов.

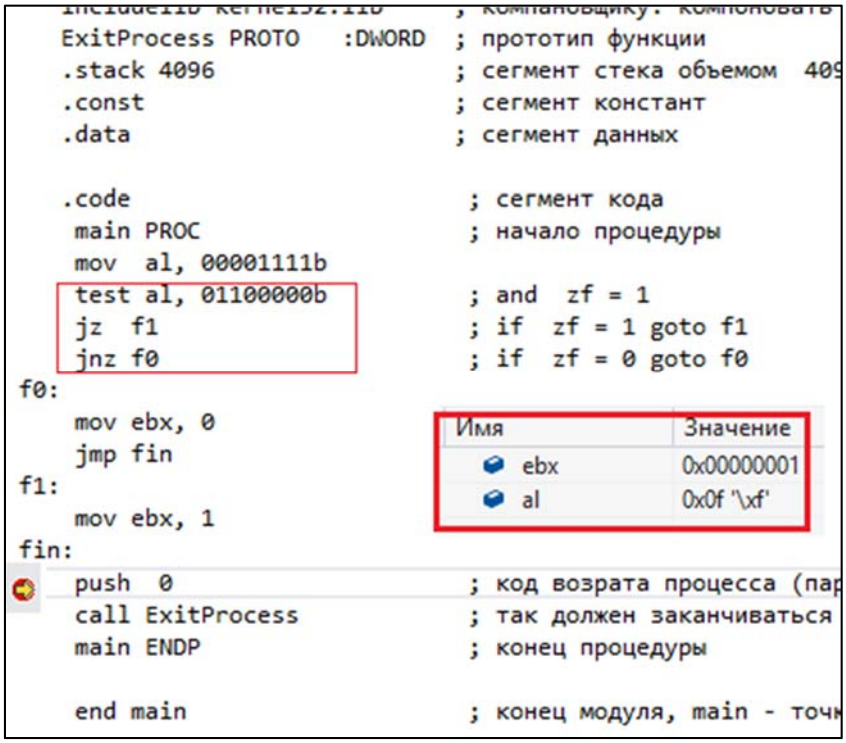
В зависимости от полученного результата устанавливает флаги состояния процессора.

Значение операнда-получателя не изменяется.

**Флаг нуля:**

ZF (Zero Flag) (ZR) Устанавливается в 1, если результат нулевой. Сбрасывается в 0 в противном случае.



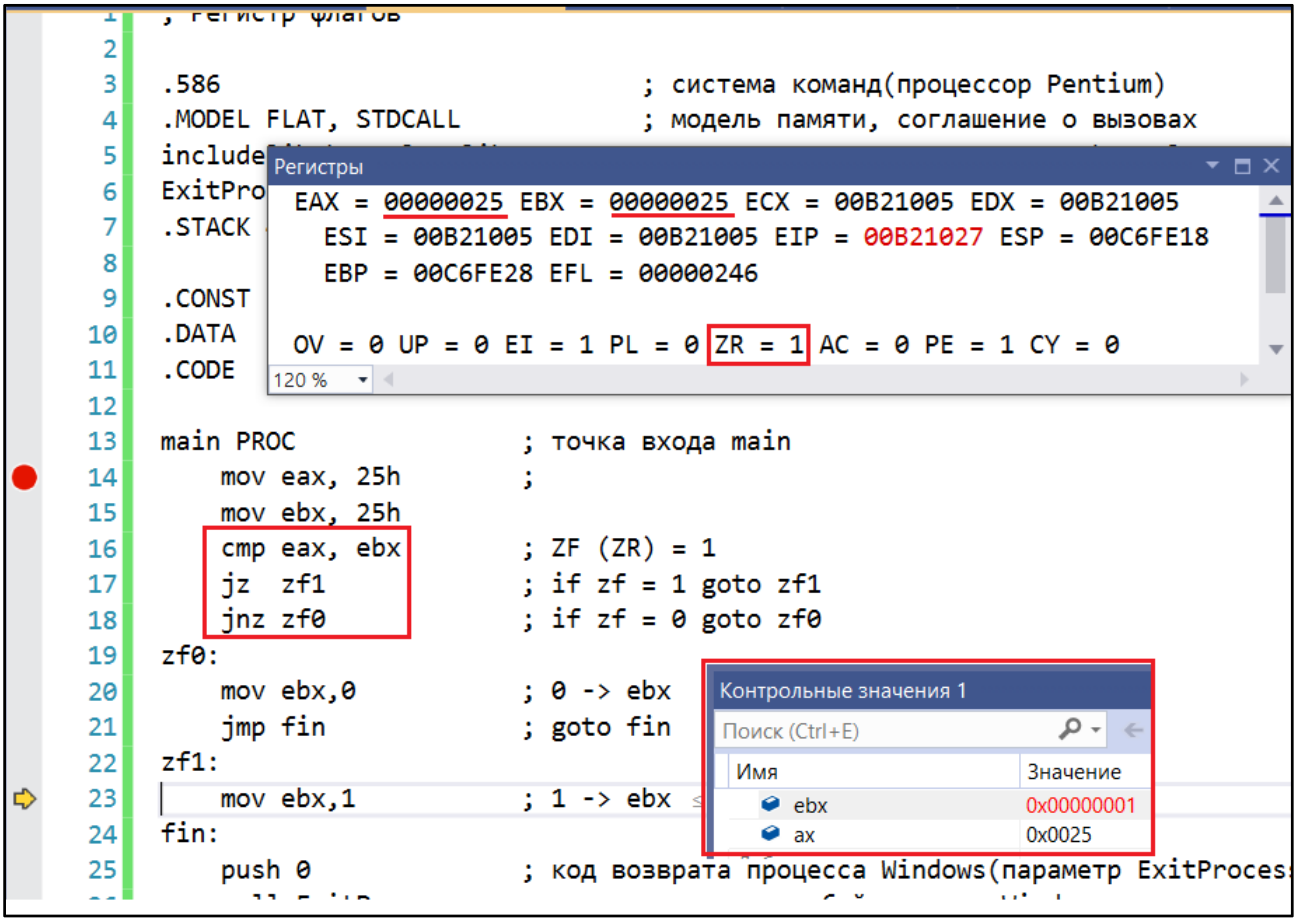

ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом
.const ; сегмент констант
.data ; сегмент данных
.code ; сегмент кода
main PROC ; начало процедуры
mov al, 00001111b
test al, 01100000b ; and zf = 1
jz f1 ; if zf = 1 goto f1
jnz f0 ; if zf = 0 goto f0
f0:
mov ebx, 0
jmp fin
f1:
mov ebx, 1
fin:
push 0 ; код возрата процесса
call ExitProcess ; так должен заканчиваться
main ENDP ; конец процедуры
end main ; конец модуля, main - точк
Имя
Значение
ebx
0x00000001
al
0x0f '\xf'

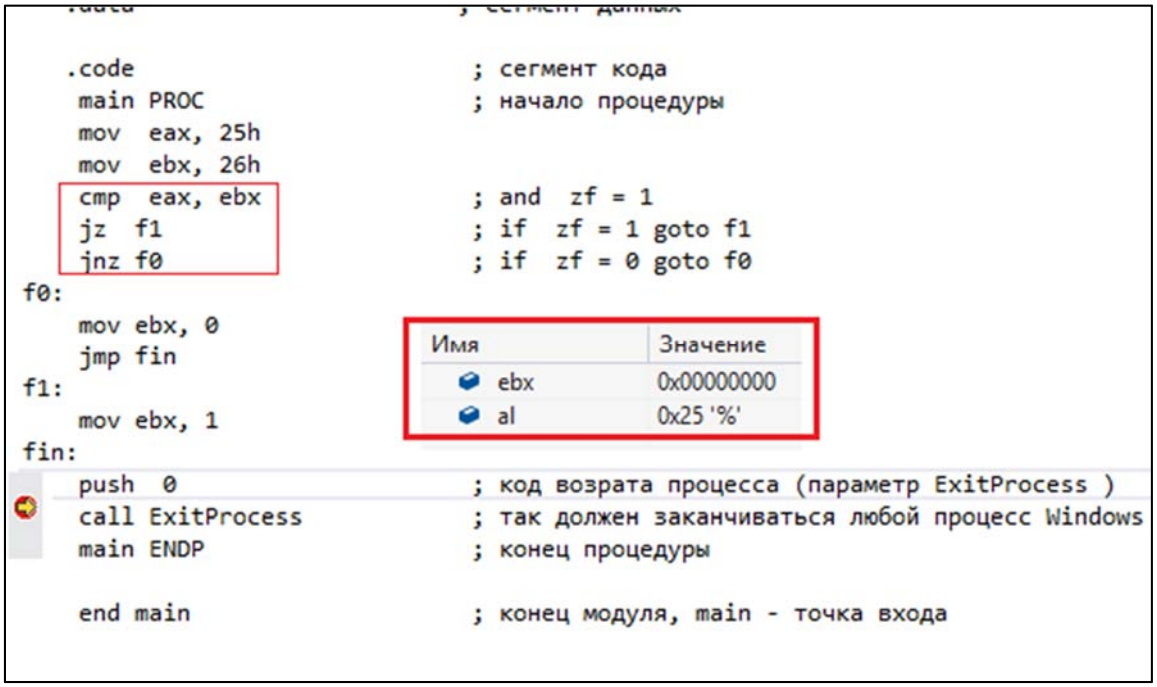
## Комада CMP

Команда вычитает исходный операнд из операнда получателя и устанавливает флаги – ***флаг переноса*** (CF), ***флаг нуля*** (ZF), ***флаг знака*** (SF), ***флаг переполнения*** (OF), ***флаг четности*** (PF), ***флаг служебного переноса*** (AF). Значение операнда-получателя не изменяется.

**Флаг нуля:**

ZF (ZR) (Zero Flag) Устанавливается в 1, если результат нулевой. Сбрасывается в 0 в противном случае.




.code ; сегмент кода
main PROC ; начало процедуры
mov eax, 25h
mov ebx, 26h
cmp eax, ebx ; and zf = 1
jz f1 ; if zf = 1 goto f1
jnz f0 ; if zf = 0 goto f0
f0:
mov ebx, 0
jmp fin
f1:
mov ebx, 1
fin:
push 0 ; код возрата процесса (параметр ExitProcess )
call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP ; конец процедуры
end main ; конец модуля, main - точка входа
Имя
Значение
ebx
0x00000000
al
0x25 '%'

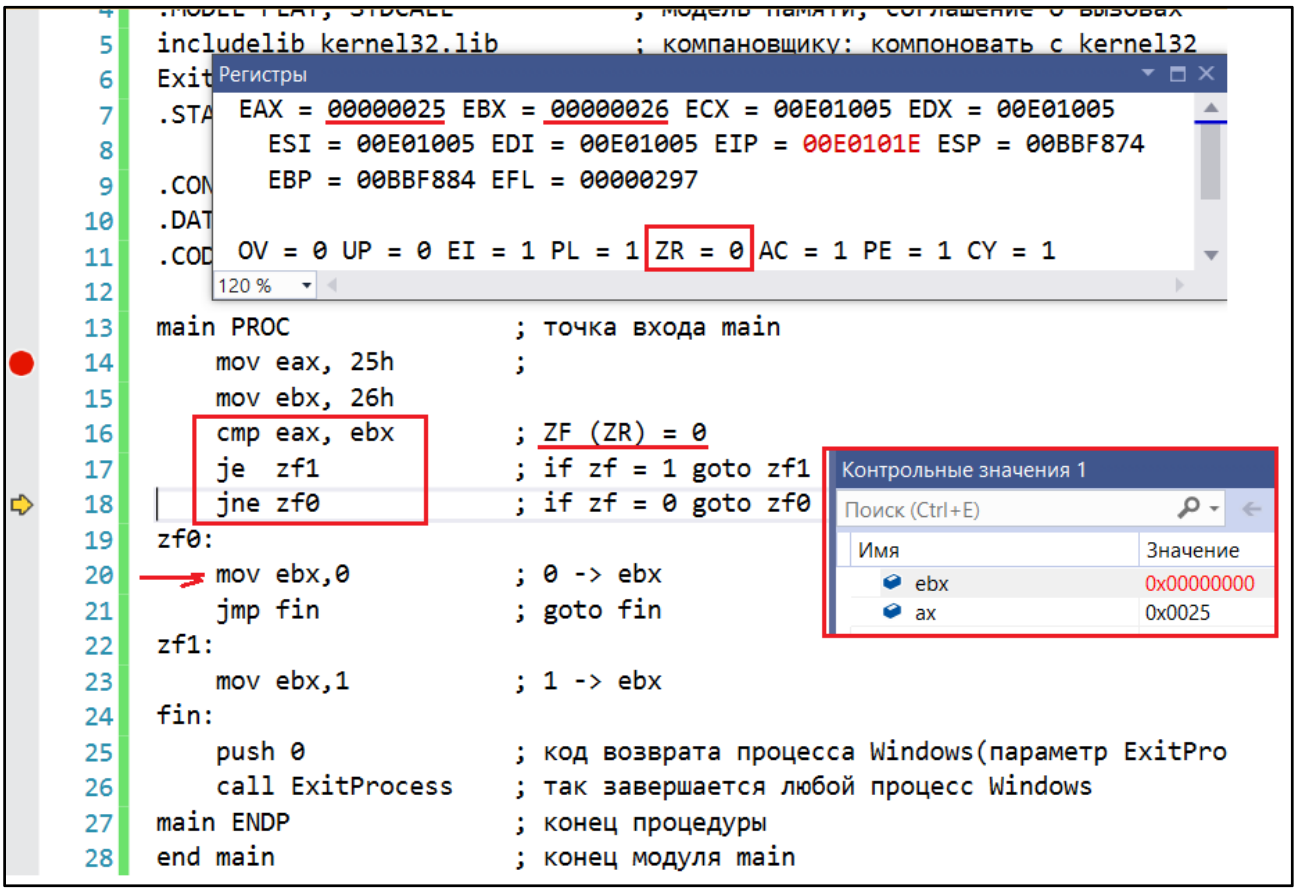
## 4. Команды переходов при беззнаковом CMP-сравнении чисел CMP

### *4.1 Команды перехода в зависимости от равенства операндов или равенства нулю регистра ECX (CX)*

**Флаг нуля:**

ZF (ZR) (Zero Flag) Устанавливается в 1, если результат нулевой. Сбрасывается в 0 в противном случае.

| Команда | Описание | Состояние |
|---|---|---|
| JE | переход, если равны | ZF=1 |
| JNE | переход, если не равны | ZF=0 |

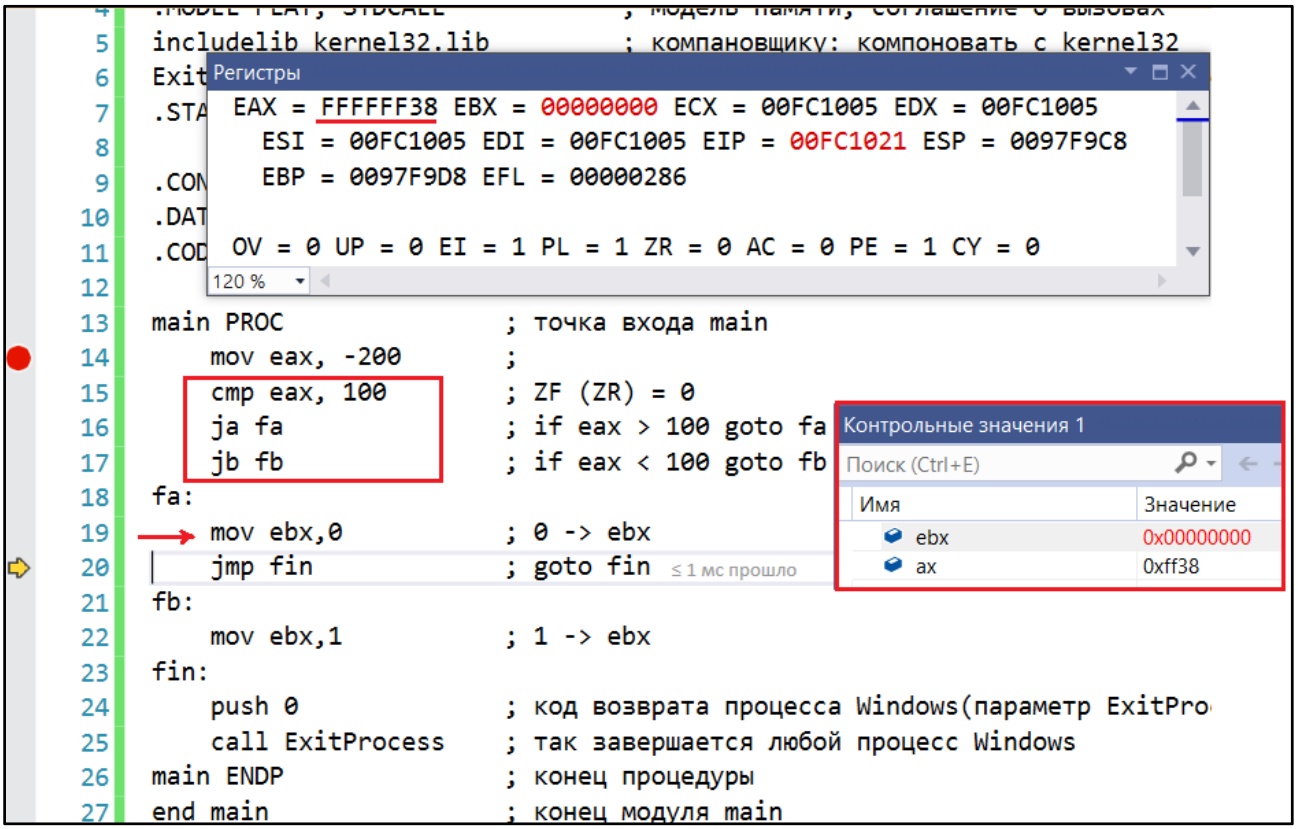


### *4.2 Команды перехода в зависимости от равенства беззнаковых операндов*

| Команда | Описание | Состояние |
|---|---|---|
| JA | переход, если выше,<br>т.е. левый операнд > правого операнда | ZF=1 |
| JB | переход, если ниже,<br>т.е. левый операнд < правого операнда | ZF=0 |

**Флаг нуля:**

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.
Сбрасывается в 0 в противном случае.

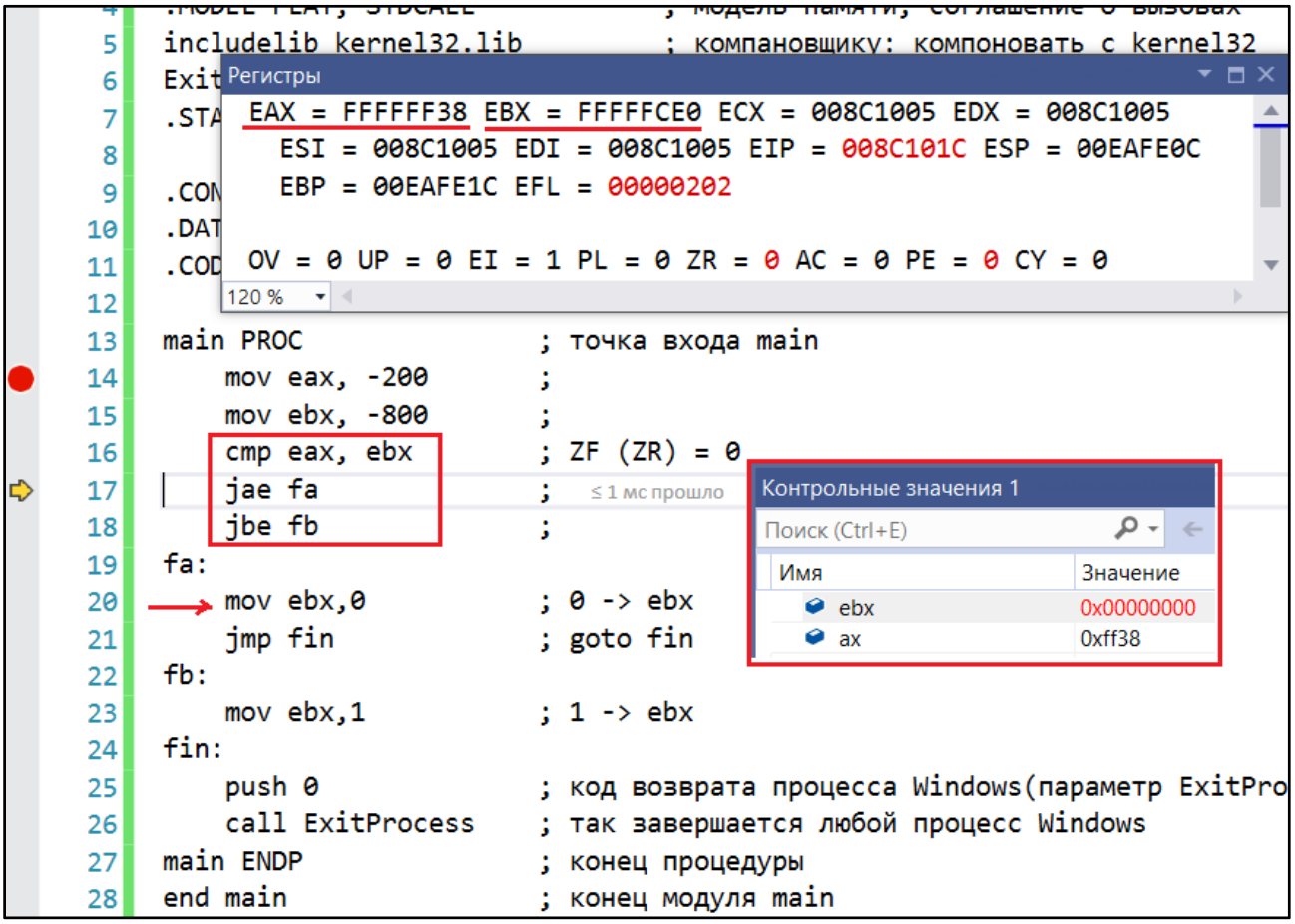


### *4.3 Команды перехода в зависимости от равенства беззнаковых операндов*

| Команда | Описание | Состояние |
|---|---|---|
| JAE | переход, если выше,<br>т.е. левый операнд >= правого операнда | ZF=0 |
| JBE | переход, если ниже,<br>т.е. левый операнд < =правого операнда | ZF=1 |

**Флаг нуля:**

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.
Сбрасывается в 0 в противном случае.



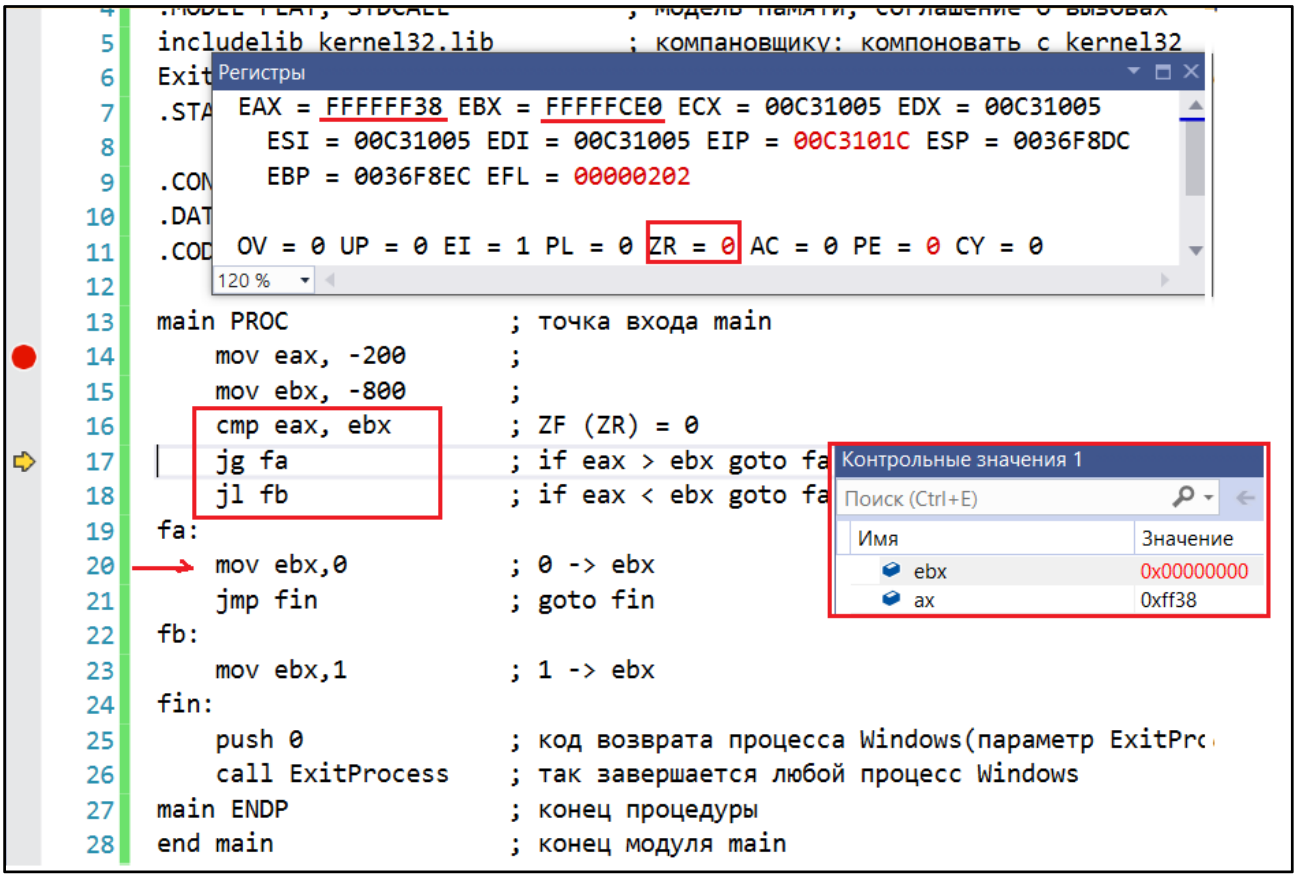
## 5. Команды переходов при CMP-сравнении чисел со знаком

### *5.1 Команды перехода после выполнения команд сравнения операндов со знаком*

| Команда | Описание | Состояние |
|---|---|---|
| JG | переход, если больше,<br>т.е. левый операнд > правого операнда | ZF=0 |
| JL | переход, если меньше,<br>т.е. левый операнд < правого операнда | ZF=0 |

**Флаг нуля:**

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.
Сбрасывается в 0 в противном случае.

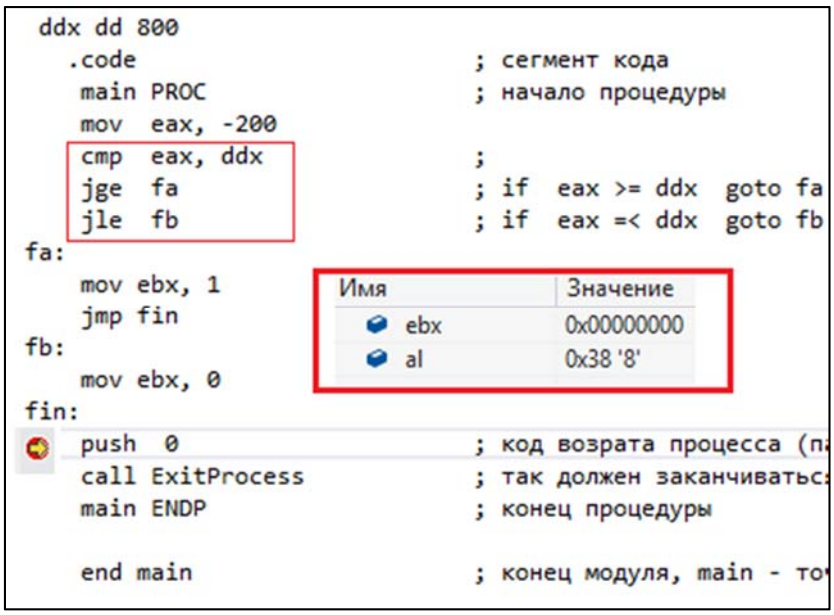


5.2Команды перехода после выполнения команд сравнения операндов со знаком

| Команда | Описание | Состояние |
|---|---|---|
| JGE | переход, если больше или равно,<br>т.е. левый операнд >= правого операнда | ZF=1 |
| JLE | переход, если меньше или равно,<br>т.е. левый операнд <= правого операнда | ZF=1 |

**Флаг нуля:**

ZF (Zero Flag) Устанавливается в 1, если результат нулевой. Сбрасывается в 0 в противном случае.



## 6. Пример программы сравнения двух строк

```
    .stack 4096                  ; сегмент стека объемом  4096
    .const                       ; сегмент констант
    .data                        ; сегмент данных
 hw  byte "Hello, World!!!"
 pm  byte "Привет,  Мир!!!"
    .code                        ; сегмент кода
     main PROC                   ; начало процедуры
     mov  ecx, sizeof hw
     cmp  ecx, sizeof pm
     je   mje                     ; if sizeof hw  ==  sizeof pm
     ja   mhw                     ; if sizeof hw >    sizeof pm
mpm:
     mov ebx, -1                  ;  hw <  pm
     jmp fin
mje:                              ;sizeof hw  ==  sizeof pm
     mov esi, 0
loopmje:
     mov  al, hw[esi]
     cmp  al, pm[esi]
     ja   mhw
     jb   mpm
     add  esi, 1
     loop loopmje
     mov ebx, 0                   ; hw = pm
     jmp fin
mhw:
     mov ebx, 1                   ;  hw >  pm
fin:
     push  0                      ; код возрата процесса (парамет
     call ExitProcess             ; так должен заканчиваться любо
     main ENDP                    ; конец процедуры

     end main                     ; конец модуля, main - точка вх
```

## 7. Команды  проверки и установки отдельных битов

Команды BT, BTR, BTC и BTC используются для работы с отдельными битами. Команды используют для организации семафоров.

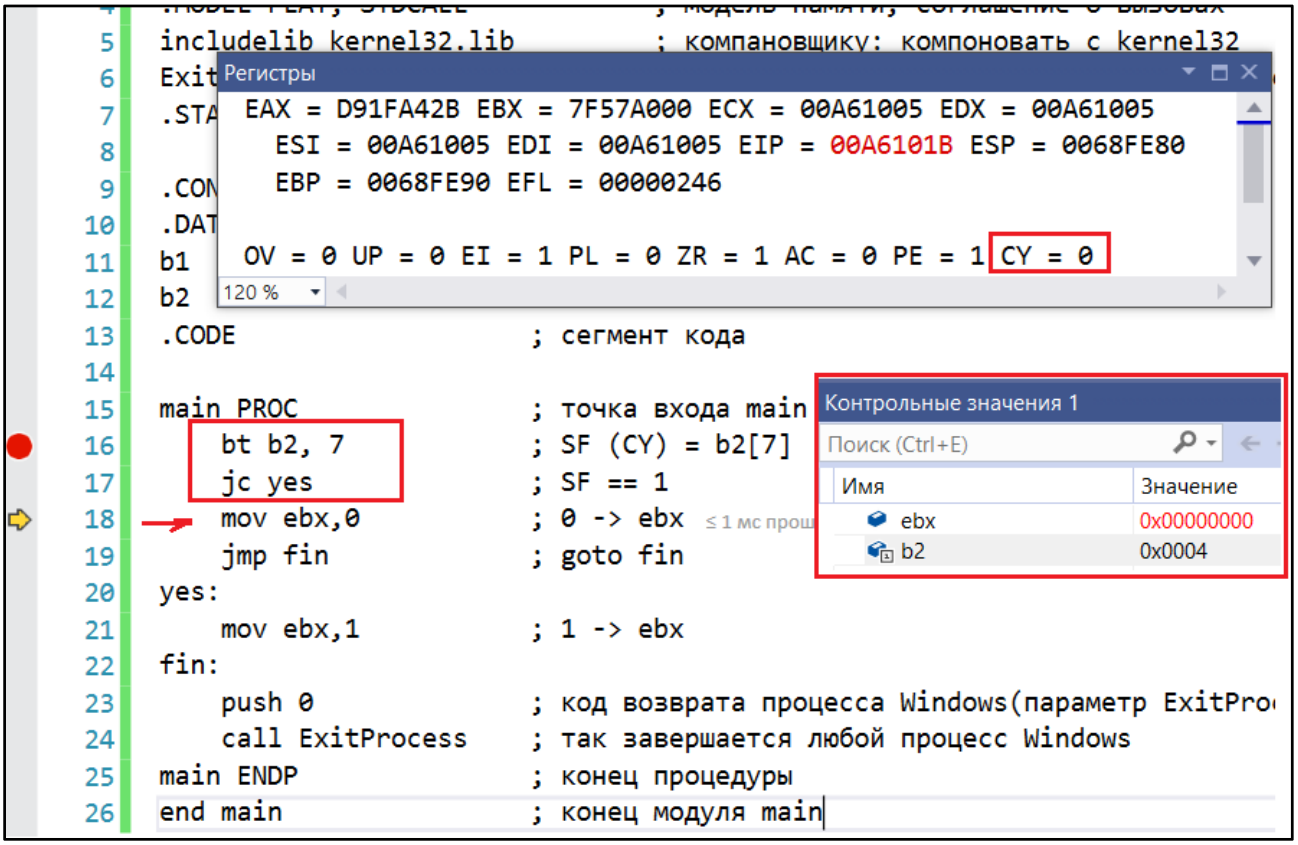
Синтаксис команды тестирование бита:

| **BT** | **строка_битов, n** |
|---|---|

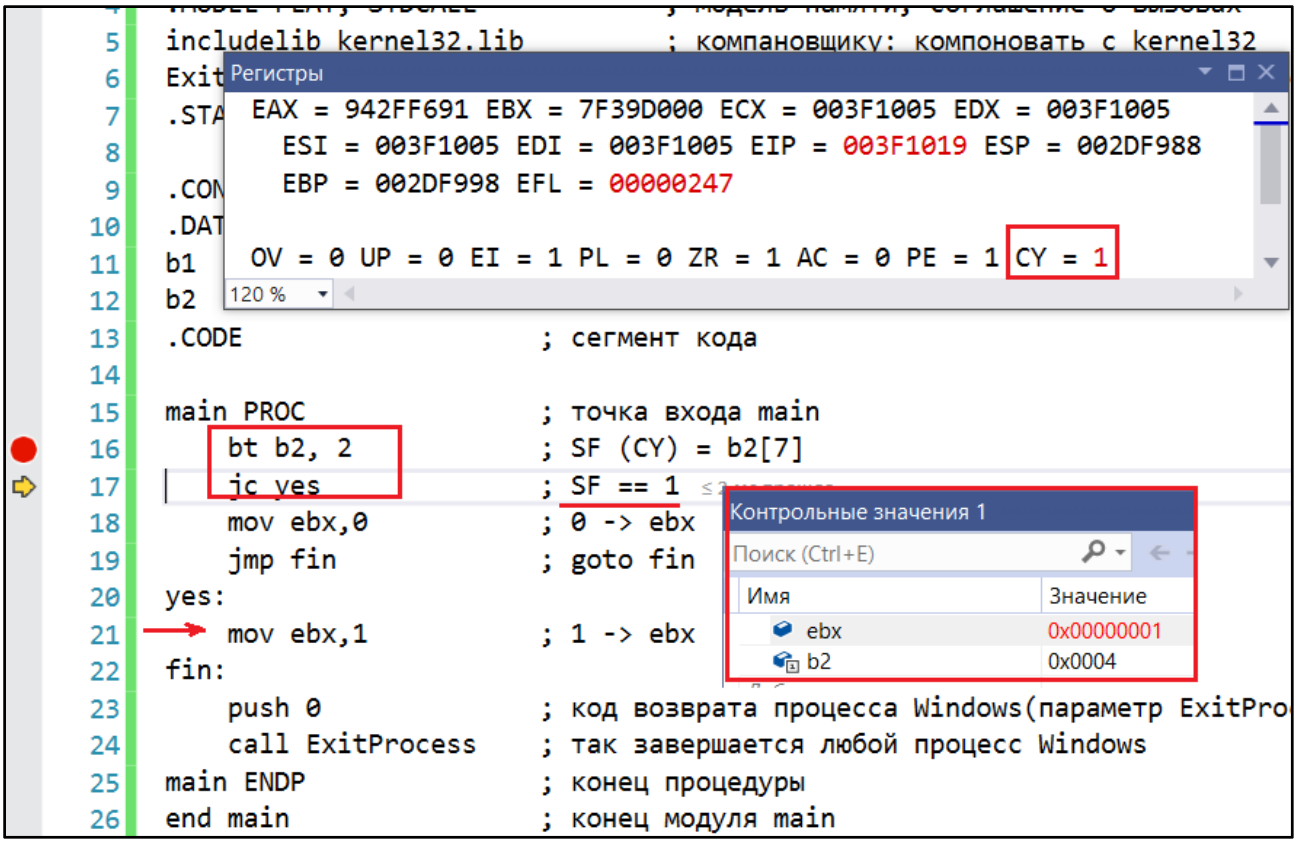
**Флаг переноса:**

CF (Carry Flag) (CY) Устанавливается в 1, когда арифметическая операция генерирует перенос или выход за разрядную сетку результата. Сбрасывается в 0 в противном случае.

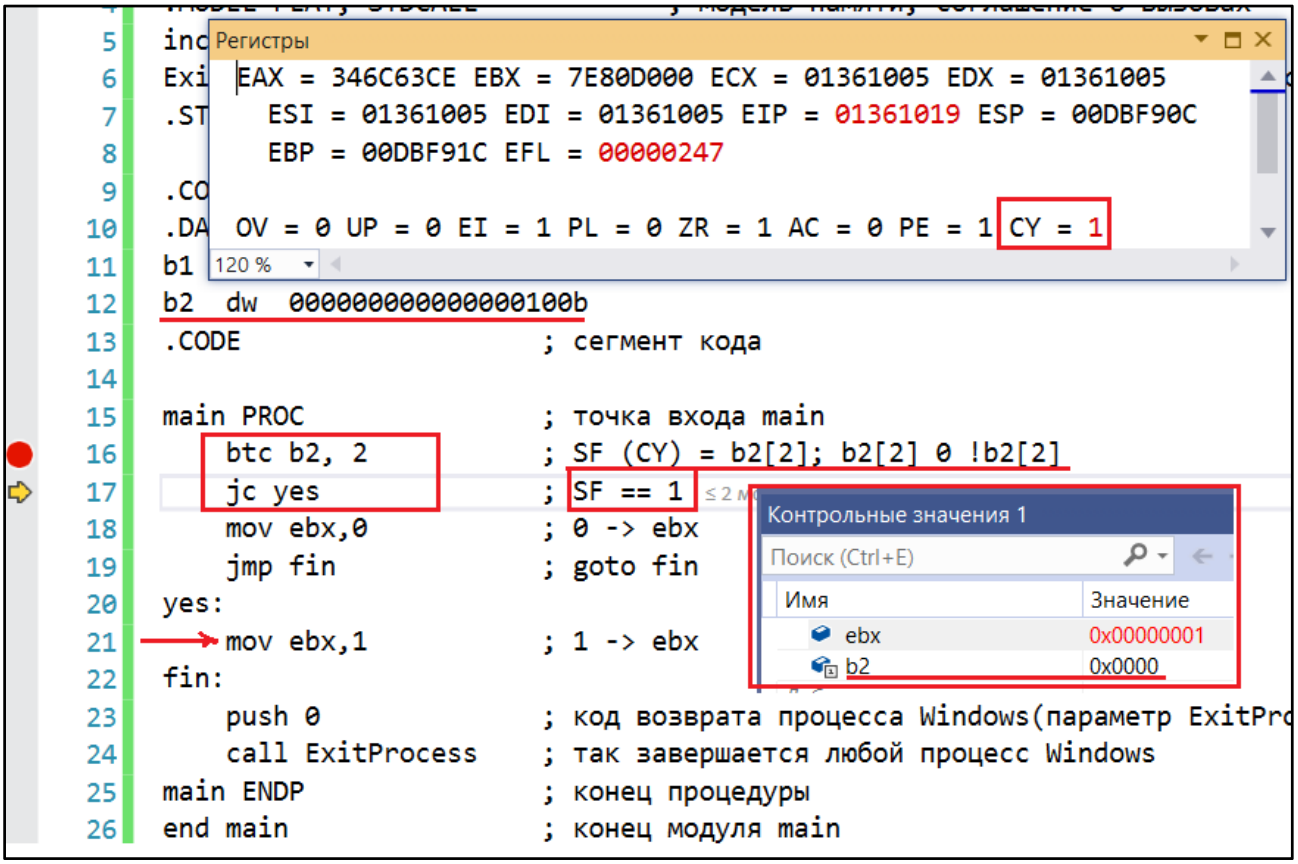
| Команда | Описание | Состояние |
|---|---|---|
| JC | переход, если перенос | CF=1 |
| JNC | переход, если нет переноса | CF=0 |



**Команда тестирование бита:**




**Команда тестирование бита с инверсией:**



**Команда тестирование бита с инверсией:**

```
   .data                        ; сегмент данных
;            876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
  .code                         ; сегмент кода
   main PROC                    ; начало процедуры
   btc b1, 2                    ; cf = bt1[2] bt1[2] = !bt1[2]
   jc  yes                      ; cf == 1
   mov ebx, 0
   jmp fin
yes:
   mov ebx, 1
fin:
   push  0                      ; код возрата процесса (параметр ExitProcess )
   call ExitProcess             ; так должен заканчиваться любой процесс Windows
   main ENDP                    ; конец процедуры

   end main                     ; конец модуля, main - точка входа
```

| Имя | Значение |
|---|---|
| ebx | 0x00000000 |
| b1 | 0x0004 |

**Команда тестирование бита с установкой:**

```
   .stack 4096                  ; сегмент стека объемом  4096
   .const                       ; сегмент констант
   .data                        ; сегмент данных
;            876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
  .code                         ; сегмент кода
   main PROC                    ; начало процедуры
   bts b1, 2                    ; cf = bt1[2] bt1[2] = 1
   jc  yes                      ; cf == 1
   mov ebx, 0
   jmp fin
yes:
   mov ebx, 1
fin:
   push  0                      ; код возрата процесса (парам
   call ExitProcess             ; так должен заканчиваться лю
   main ENDP                    ; конец процедуры
```

| | |
|---|---|
| ebx | 0x00000000 |
| b1 | 0x0004 |

**Команда тестирование бита со сбросом:**

```
;            876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
  .code                         ; сегмент кода
   main PROC                    ; начало процедуры
   btr b2, 2                    ; cf = bt2[2] bt2[2] = 0
   jc  yes                      ; cf == 1
   mov ebx, 0
   jmp fin
yes:
   mov ebx, 1
fin:
   push  0                      ; код возрата процесса (параметр Exit
   call ExitProcess             ; так должен заканчиваться любой проц
   main ENDP                    ; конец процедуры

   end main                     ; конец модуля, main - точка входа
```

| | |
|---|---|
| ebx | 0x00000001 |
| b2 | 0x0000 |