**Арифметические и побитовые операции**

* Для сложения и вычитания – **add и sub**
  + Сложение и вычитание выполняется одинаково для знаковых и беззнаковых чисел
  + Отличается в работа со знаковыми и беззнаковыми числами в том, какие флаги анализируются
* Команды условного перехода для беззнаковых

ja ; > jna

jae ; >= jnae

jb ; < jnb

jbe ; <= jnbe

* Команды условного перехода для знаковых

jg ; > jzg

jge ; >= jnge

jl ; < jnl

jle ; <= jnle

* Для сложения и вычитание больших чисел предусмотрены
  + **ADC** – *Add With Carry*
    - dest <- dest + (src + CF)
  + **SBB** – *Subtract With Borrow*
    - dest <- dest – (src + CF)
  + Задача: сложить числа DX:AX и BX:CX, результат в DX:AX

add ax, cx

DX AX

adc dx, bx

Если был перенос, прибавляем еще его

AX + CX

BX CX

* + Inc dest
    - dest <- dest + 1
  + dec dest
    - dest <- dest – 1

**Команда NEG**

* Отрицание с изменением знака на противоположный
* Формат:
  + neg dest
* Флаги: OF, SF, ZF, AF, CF, DF
* Логика:
  + If dest = 0 then cf <- 0 else cf <- 1
  + dest <- - dest

**Умножение и деление**

* + Различается знаковое и беззнаковое

**Умножение:**

* + MUL – беззнаковое
  + IMUL – знаковое

**Деление:**

* DIV
* IDIV

**Поведение команд зависит от:**

* Разрядности
* Размера операндов

|  |  |
| --- | --- |
| Операнд | Принцип работы |
| mul r/m8 | AX <- AL\*r/m8 |
| mul r/m16 | DX:AX <- AX\*r/m16 |
| mul r/m32 | EDX:EAX <- EAX\*r/m32 |
| mul r/m64 | RDX:RAX <- RAX\*r/m64 |

|  |  |
| --- | --- |
| Операнд | Принцип работы |
| div r/m8 | AL <- AX/src (ост AH) |
| div r/m16 | AX < DX:AX/src (ост DX) |
| div r/m32 | EAX <- EDX:EAX (ост EDX) |
| div r/m64 | RAX <- RDX:RAX (ост RDX) |

**Если результат деления не помещается в предназначенные регистры, генерируется ошибка**

IMUL и IDIV – аналогично, но учитывая знак

Кроме того, у IMUL есть:

* 2-операндная форма
* 3-операндная форма
* Подробнее – в справке Intel

**Расширение чисел**

* Расширение (extension) – преобразование в эквивалентное число большей разрядности
* Может быть знаковым/беззнаковым
  + Беззнаковое – дополняется 0-ми в старших разрядах
  + Для знаковых – копируется знаковый бит исходного числа
* Инструкции:
  + **CBW –** Byte to Word
  + **CWD –** Word to Double Word
  + **CWDE –** DWord to DWord Ext
  + **CDQ –** DWord to QuadWord

**Преобразование типов г/г**

* Инструкции:
  + **movzx –** Move with Zero-Extend
  + **movsx –** Move with Sign-Extend
* Примеры:
  + movzx dx, bl
  + movsx si, ch
* Инструкции преобразования типов не изменяют Flags

**Команды изменения флагов**

* Инструкции:

STC CF <- 1

CLC CF <- 0

CMC CF <- not(CF)

CLD DF <- 0

STD DF <- 1

CLI IF <- 0

STI IF <- 1

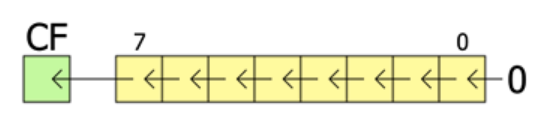
* Все инструкции – без операндов

**Побитовые сдвиги**

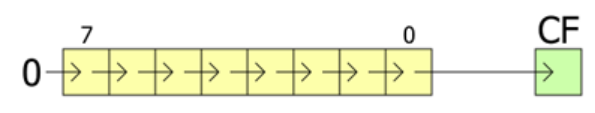
* Побитовые сдвиги:
  + Арифметические
  + Логические
  + Циклические
* Направления:
  + Влево – к старшему разряду
  + Вправо – в сторону младшего разряда

**Логические сдвиги:**

* Влево:
  + В освободившийся разряд: 0



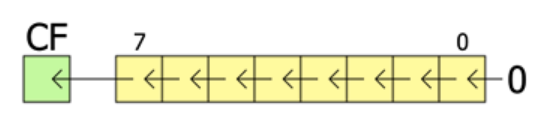
* Вправо
  + В освободившийся разряд – 0



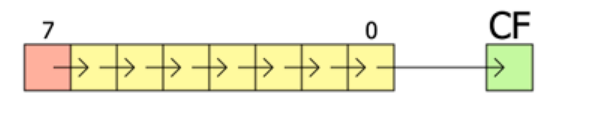
* Для умножения на 2N **(БЕЗЗНАКОВОЕ)**

**Арифметический сдвиг:**

* Влево
  + В освободившийся разряд – 0



* Вправо
  + В освободившийся разряд – старший разряд исходного значения



**Инструкции:**

* **SAL –** Shift Arifm Left
* **SAR –** Shift Arifm Right
* **SHL –** Shift Logical Left
* **SHR –** Shift Lofical Right

**Формат:**

* Инструкция dest, count
  + count – imm8 / регистр CL

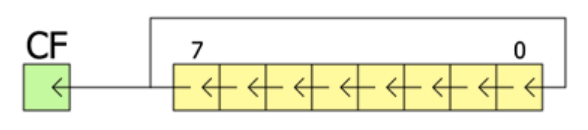
**Флаги:**

* Если count = 0, тогда flags не меняются
* CF – значение последнего выдвинутого байта
  + Для SHL/SHR, если count больше размера dest, значение CF не определено

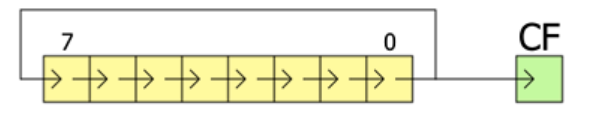
Начиная с 80286 на значение count накладывается маска IF, поэтому количество бит для этого сдвига отличается от одо31

**Циклический сдвиг через CF**

* Влево



* Вправо



**Инструкции:**

* **RCL –** Rotate through carry left
* **RCR –** Rotate through carry right
* **ROL –** Rotate Left
* **ROR –** Rotate Right

**Формат:**

* Инструкция dest, count
  + count – imm8 / регистр CL

**Флаги:**

* CF – значение оставшегося в нем бита
* SF, ZF, PF, AF – не изменяются
* **OF – изменяется только для count = 1 (См. доки интел)**

**Начиная с 80286 – маска**

**SHLD/SHRD**

Инструкция операнд1, операнд2, count

