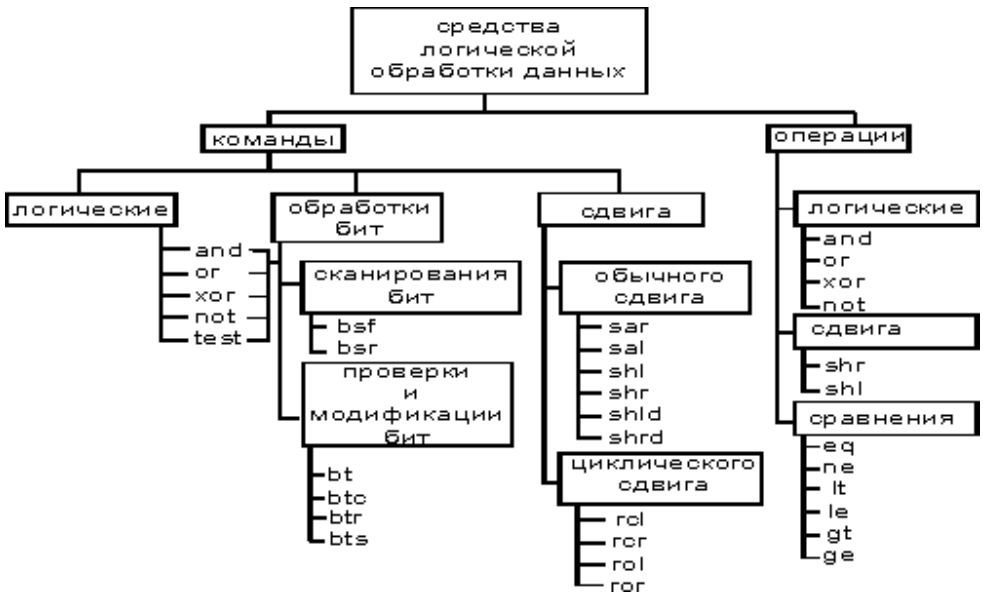
Лекция 8

2.3. Команды логической обработки данных

Команды разделяются на 4 группы:

1. логические команды;
2. команды сдвига;
3. команды циклического сдвига;
4. команды обработки бит данных.



2.3.1. Логические команды **(AND, OR, XOR, NOT, TEST)**

Эти команды реализуют поразрядные операции, то есть, і-тый разряд результата зависит от і-тых разрядов операндов. Операции выполняются параллельно над всеми разрядами. “Истина” - когда будет 1 хотя бы в одном разряде результата, и “Ложь” - когда во всех битах результата - нули.

Команды изменяют все флаги условий, но чаще обращаем внимание на бит **ZF**. Если результат - “истина”, то ZF = 0, а если “ Ложь”, то ZF = 1.

Операндами логических операций могут быть слова или байты, но **НЕ** одновременно.

Команда побитовое ”AND”

AND приемник, источник - операция логического умножения. Команда выполняет поразрядно логическую операцию “И” (конъюнкцию) над битами операндов **приемник** и **источник**. Результат записывается на место **приемника.**

Например:

```
AND AX, BX; AX*BX - результат записать в AX
```

В бите приемника устанавливается 1 тогда, когда в соответствующих битах источника и приемника были 1. Если же в бите источника был 0, то в соответствующем бите приемника установится 0, независимо от того, что там было. Поэтому, команда **AND** используется для

селективного установления 0 в тех битах приемника, которым отвечает 0 в источнике. Подбирая соответствующие биты источника, влияем на определенные биты приемника. Такие действия часто используются в операциях над битами для управления устройствами обмена. При этом оператор источник называется маской, а сама операция - *маскированием*.

Например:

```
AND  10010101 – приемник
     01011011 – источник (маска)
     00010001
     ↑   ↑
состояние изменилось
```

```
; AX=95h, BX=5Bh
AND AX, BX; AX = 11h
```

Операндами команды **AND** могут быть **байты** или **слова**. Могут использоваться два регистра, регистр со словом (байтом памяти), или непосредственное значение:

```
AND AL, M_BYTE
AND M_BYTE, AL
AND TABLE [BX], MASK
AND BL, 1101B
```

Следовательно, команда **AND** изменяет приемник. Однако, когда бит изменяется лишь самим устройством, то можно использовать команду **AND** для проверки состояния устройства.

Например, порт 200 соединенный с 16-битовым регистром внешнего устройства и 6-й бит показывает, включено (1) или НЕ включено (0) это устройство.

```
CHECK: IN AX, 200
       AND AX, 1 000 000B
       JZ CHECK ; НЕ включено
```

Программа может работать дальше, когда лишь устройство включено:

Если устройство НЕ включено, то в 6-ом бите - 0, и результат команды **AND** - 0, следовательно, **ZF** = 1 и выполняется команда JZ.

Как только в бите станет 1, **ZF** = 0, и команда JZ НЕ выполняется.

1	7	6	2	0
F	F	F	F	F

Команда побитовое ”OR ”

OR приемник, источник — операция логического сложения.

Команда выполняет поразрядно логическую операцию ИЛИ (дизъюнкцию) над битами операндов приемник и источник. Результат записывается на место приемник:

```
OR AX, BX ;AX+BX - результат записать в AX
```

OR **10**01**0**0**0**1 – приемник
0**1**011**0**1**1** – источник (маска)
1**1**011**0**1**1**
↑ ↑ ↑
состояние изменилось

Следовательно, команду **OR** употребляют для селективного установления 1 в приемнике.

```
; AX=91h, BX=5Bh
OR AX, BX; AX = DBh
```

Команда побитовое ”XOR ”

XOR приемник, источник — операция логического исключающего сложения.

Команда выполняет поразрядно логическую операцию исключающего ИЛИ над битами операндов приемник и источник. Результат записывается на место приемник.

Устанавливает 1 в те биты приемника, которые отличаются от битов источника. Пример:

```
XOR AX, TEST_PAR
JZ ALPHA
```

Если хотя бы в одном бите не совпадают коды, то результат команды **XOR**=1 и ZF=0, следовательно, команда **JZ** не будет выполнена. Она будет выполняться лишь тогда, когда коды полностью совпадают. Команда **XOR** изменяет приемник.

XOR 1**1**01**0**01**1** – приемник
0**1**00**1**00**1** – источник (маска)
1**0**011**1**01**0**
↑ ↑ ↑
состояние изменилось

```
; AX=93h, BX=49h
XOR AX, BX; AX = 9Ah
```

1	7	6	4	2	0
F	F	F	F	F	F

Команда побитовое ” NOT ”

NOT операнд — операция логического отрицания.

Результат записывается на место операнда. Команда NOT изменяет все биты на противоположные:

Пример:

```
flag    db      0ffh ; значение флага — истина
...
cycl:
...
        CMP     flag,0
        JE      m1
...
m1:     not     flag    ;установить флаг в истину
```

выполнение команды ***НЕ влияет на флаги***

```
; AX=10011010b
NOT AX; AX = 01100101b
```

Команда побитовое ” TEST ”

TEST приемник, источник — операция “проверить” (способом логического умножения).

Команда выполняет поразрядно логическую ***операцию И*** над битами операндов **приемник и источник**. Состояние операндов остается прежним, изменяются только флаги ***zf, sf,*** и ***pf***, что дает возможность анализировать состояние отдельных битов операнда без изменения их состояния.

Если хотя бы одна пара битов равняется 1, то результат команды равняется 1, следовательно, **ZF** = 0.

Например:

```
TEST    al,01h
JNZ     m1      ;переход, если нулевой бит al равен 1
```

TEST 0**11**011**0**1 – приемник
0**11**00**1**00 – источник (маска)
0**11**00**1**00
↑↑ ↑
проверены биты

1	7	6	2	0
F	F	F	F	F

2.3.2. Команды сдвига
(SHL, SHR, SAL, SAR, SHLD, SHRD, ROL, ROR, RCL, RCR)

К этой группе принадлежит 10 команд.

1. **6** сдвигают операнд (**SHL, SHR, SAL, SAR, SHLD, SHRD**)
2. **4** вращают или циклически сдвигают (**ROL, ROR, RCL, RCR**).

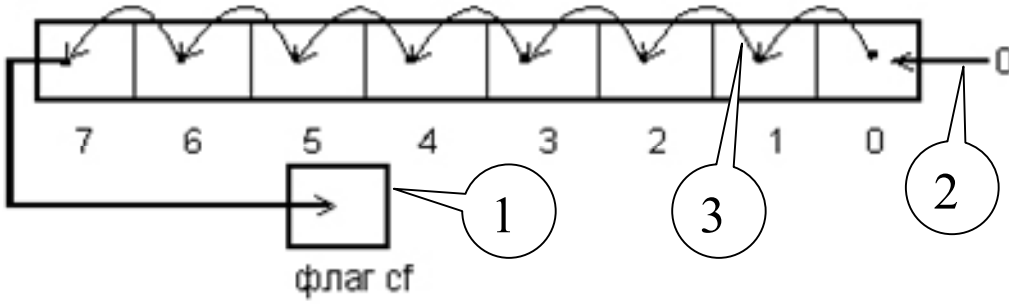
SHL, SHR, SAL, SAR, SHLD, SHRD

Алгоритм:

1. очередной "выдвигаемый" бит устанавливает флаг **CF;**
2. бит, вводимый в операнд с другого конца, имеет значение 0;
3. при сдвиге очередного бита он переходит во флаг **cf**, при этом значение предыдущего сдвинутого бита теряется!

SHL операнд, счетчик_сдвигов (Shift Logical Left) - логический сдвиг ***влево***.

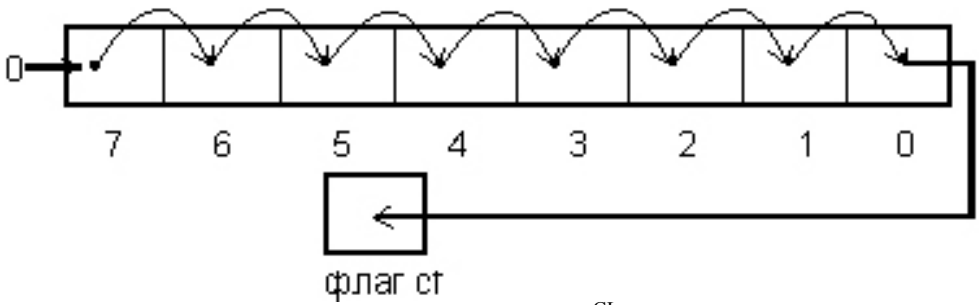
Содержимое операнда сдвигается влево на количество битов, определяемое значением **счетчик_сдвигов**. Справа (в позицию младшего бита) вписываются нули;



SHL AX, CL -- умножить AX без знака на 2^{CL}

SHR операнд, счетчик_сдвигов (Shift Logical Right) — логический сдвиг ***вправо***.

Содержимое операнда сдвигается вправо на количество битов, определяемое значением **счетчик_сдвигов**. Слева (в позицию старшего, знакового бита) вписываются нули. На рис. показан принцип работы этих команд.

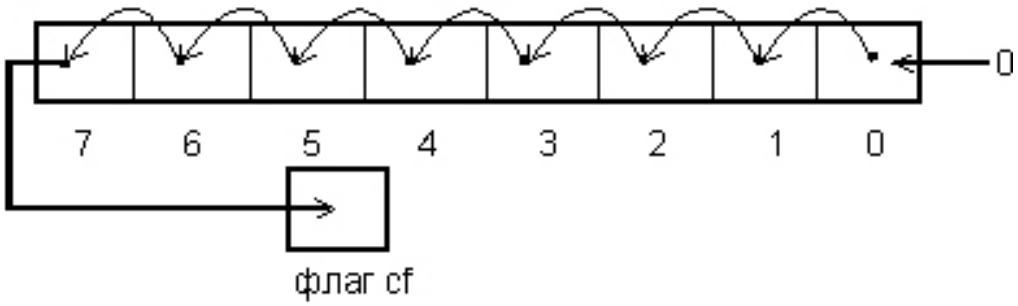


SHR AX, CL -- разделить AX без знака на 2^{CL}

SAL операнд, счетчик_сдвигов (Shift Arithmetic Left) — арифметический сдвиг ***влево***.

Содержимое операнда сдвигается влево на количество битов, определяемое значением **счетчик_сдвигов**. Справа (в позицию младшего бита) вписываются нули. Команда **SAL** не

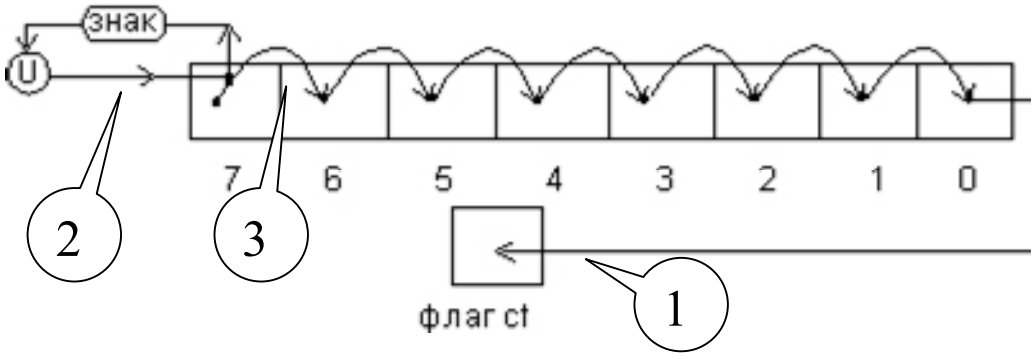
сохраняет знака, но устанавливает флаг ***cf*** в случае смены знака очередным выдвигаемым битом. В остальном команда SAL полностью аналогична команде **SHL**;



SAL AX, CL -- умножить AX со знаком на 2^{CL}

SAR операнд, счетчик_сдвигов (Shift Arithmetic Right) — арифметический сдвиг ***вправо***.

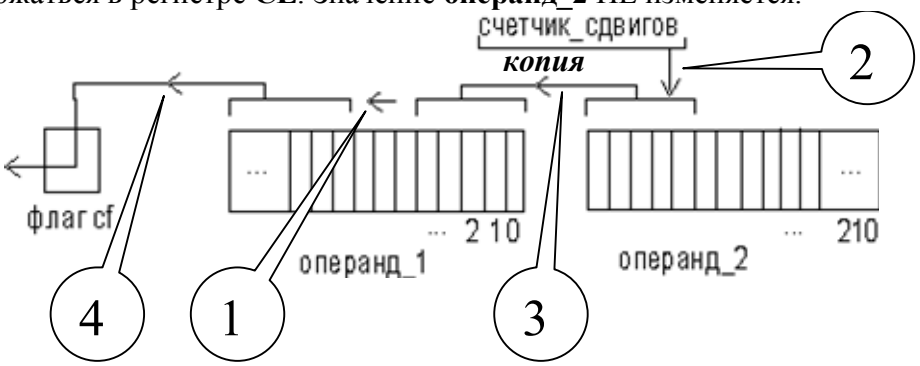
Содержимое операнда сдвигается вправо на количество битов, определяемое значением **счетчик_сдвигов**. Команда **SAR** сохраняет знак, восстанавливая его после сдвига каждого очередного бита.



SAR AX, CL - разделить AX со знаком на 2^{CL}.

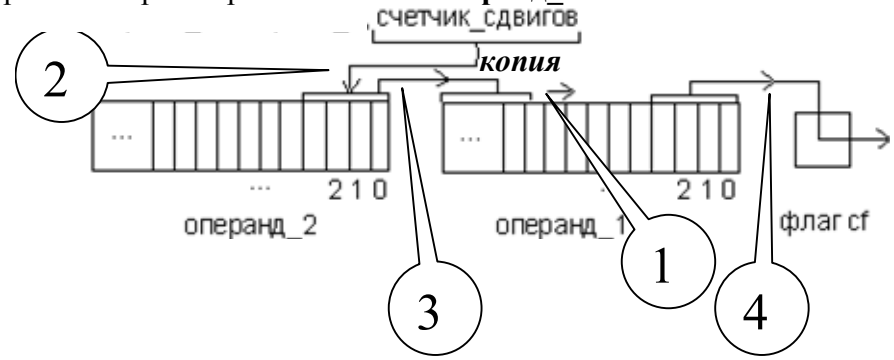
SHLD операнд_1,операнд_2, счетчик_сдвигов — сдвиг ***влево*** двойной точности.

Команда **SHLD** производит замену путем сдвига битов операнда **операнд_1** влево, заполняя его биты справа значениями битов, вытесняемых из **операнд_2** согласно схеме на рис. Количество сдвигаемых бит определяется значением **счетчик_сдвигов**, которое может лежать в диапазоне 0...31. Это значение может задаваться непосредственным операндом или содержаться в регистре **CL**. Значение **операнд_2** НЕ изменяется.



SHRD операнд_1,операнд_2,счетчик_сдвигов — сдвиг ***вправо*** двойной точности.

Команда производит замену путем сдвига битов операнда **операнд_1** вправо, заполняя его биты слева значениями битов, вытесняемых из **операнд_2** согласно схеме на рис. Количество сдвигаемых бит определяется значением счетчик_сдвигов, которое может лежать в диапазоне 0...31. Это значение может задаваться непосредственным операндом или содержаться в регистре **cl**. Значение **операнд_2** НЕ изменяется.



2.3.3. Команды циклического сдвига

1. Команды простого циклического сдвига

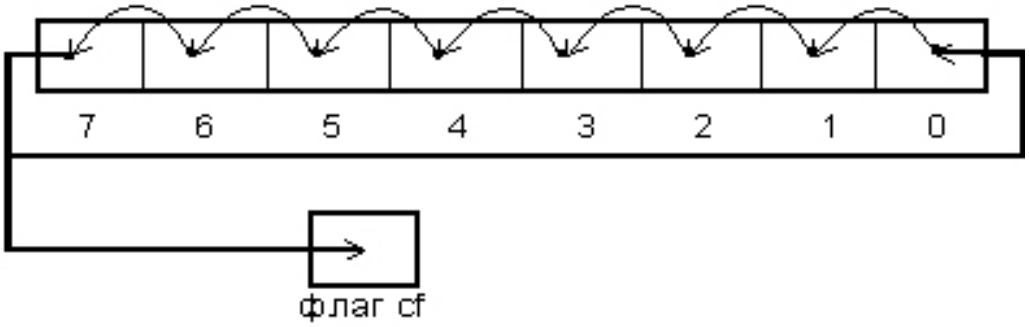
ROL, ROR

Алгоритм:

1. сдвиг всех битов операнда влево на один разряд, при этом старший бит операнда вдвигается в операнд справа и становится значением младшего бита операнда;
2. одновременно выдвигаемый бит становится значением флага переноса **cf**;
3. указанные выше два действия повторяются количество раз, равное значению второго операнда.

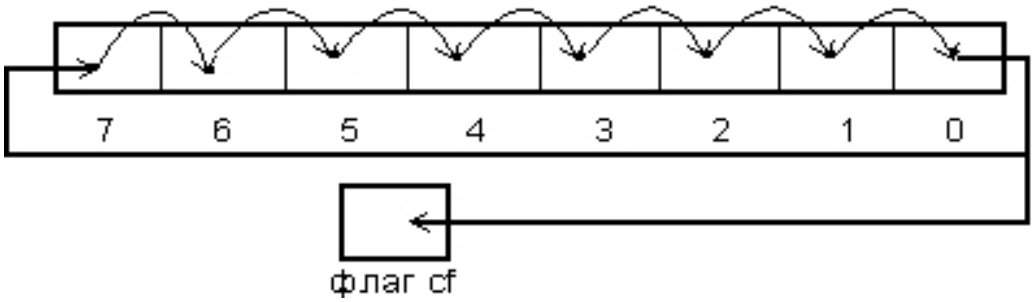
ROL операнд, счетчик_сдвигов (Rotate Left) — циклический сдвиг ***влево***.

Содержимое операнда сдвигается влево на количество бит, определяемое операндом **счетчик_сдвигов**. Сдвигаемые влево биты записываются в тот же операнд справа.



ROR операнд, счетчик_сдвигов (Rotate Right) — циклический сдвиг ***вправо***.

Содержимое операнда сдвигается вправо на количество бит, определяемое операндом **счетчик_сдвигов**. Сдвигаемые вправо биты записываются в тот же операнд слева.



2. Команды циклического сдвига через флаг переноса cf

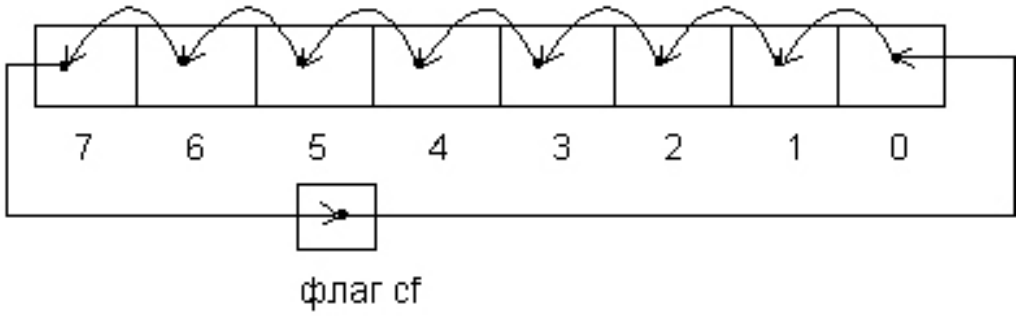
RCL, RCR

Алгоритм:

1. сдвиг всех битов операнда влево на один разряд, при этом старший бит операнда становится значением флага переноса **cf**;
2. одновременно старое значение флага переноса **cf** сдвигается в операнд справа и становится значением младшего бита операнда;
3. указанные выше два действия повторяются количество раз, равное значению второго операнда команды **rcl**.

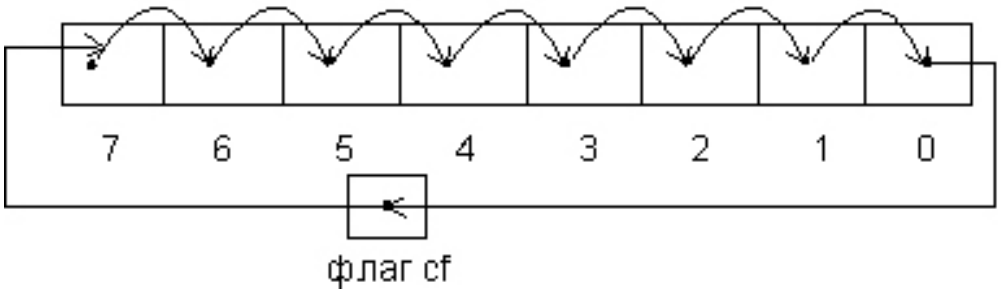
RCL операнд, счетчик_сдвигов (Rotate through Carry Left) — циклический сдвиг ***влево*** через перенос.

Содержимое операнда сдвигается влево на количество бит, определяемое операндом **счетчик_сдвигов**. Сдвигаемые биты поочередно становятся значением флага переноса **cf**.



RCR операнд, счетчик_сдвигов (Rotate through Carry Right) — циклический сдвиг ***вправо*** через перенос.

Содержимое операнда сдвигается вправо на количество бит, определяемое операндом **счетчик_сдвигов**. Сдвигаемые биты поочередно становятся значением флага переноса **cf**.



Эти команды выполняются намного быстрее, чем **MUL** и **DIV**