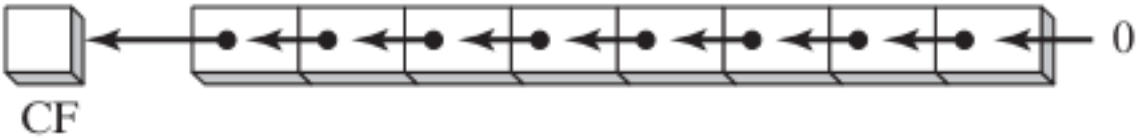
Arithmetic Shift Left/Arithmetic Shift Right

==========

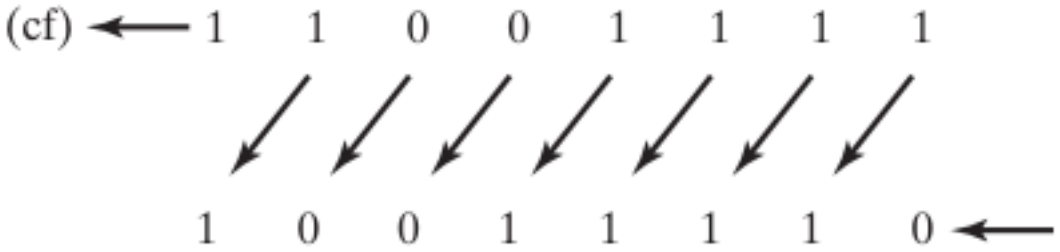
SAL (Shift Arithmetic Left) Instruction:

==========

The **SAL instruction**, also known as **Shift Arithmetic Left**, is similar to the SHL (Shift Left) instruction in its behavior. It's used for moving the bits of the destination operand to the left by a specified number of bits.



Shift Left Operation: SAL shifts each bit in the destination operand to the next highest bit position. In other words, it moves the bits leftward. If you shift binary 11001111 to the left by one bit, it becomes 10011110:



Carry Flag: As the bits shift to the left, the highest bit in the destination operand is moved into the Carry flag. The Carry flag stores the value that was in the most significant bit (MSB) position.

Lowest Bit: The lowest bit in the destination operand is assigned the value 0. This means that the bit that was in the least significant bit (LSB) position is set to 0.

Discarding the Carry Bit: The bit that was in the Carry flag is effectively discarded in this operation. It is no longer a part of

the destination operand.

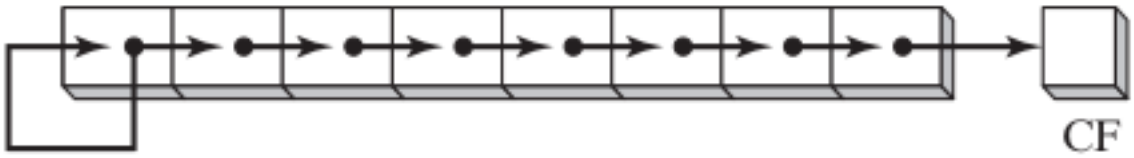
The SAL instruction is typically used to perform logical left shifts and, like SHL, is often used for operations such as multiplying a value by powers of 2. It's a valuable tool for bit manipulation in assembly language programming.

==========

SAR (Shift Arithmetic Right) Instruction:

==========

The SAR instruction, which stands for Shift Arithmetic Right, is used for shifting the bits of the destination operand to the right by a specified number of bits. Unlike logical right shifts (as in SHR), SAR is used for signed numbers to preserve the sign bit.



Shift Right Operation: SAR performs a right shift on the bits of the destination operand, moving them to the right.

```
SAR destination,count
```

Sign Bit Preservation: The key feature of SAR is that it preserves the sign bit. This means that when a signed number is shifted right, the most significant bit (MSB), which indicates the sign (0 for positive, 1 for negative), is preserved during the shift.

Carry Flag: The lowest bit of the destination operand is moved into the Carry flag, similar to other shift instructions. The Carry flag now stores the value that was in the least significant bit (LSB) position.

Bit Insertion: The highest bit position, which was shifted out, is determined by the original sign bit. If the original sign bit was 0 (indicating a positive number), the highest bit is set to 0. If the original sign bit was 1 (indicating a negative number), the highest bit is set to 1. This preserves the signedness of the number.

SAR is used for operations like signed division by powers of 2 and other signed arithmetic operations. It's a crucial instruction in working with signed numbers in assembly language programming, ensuring that the sign bit is maintained correctly during shifts.

==========

Examples:

==========

The following example shows how **SAR duplicates the sign bit**. AL is negative before and after it is shifted to the right:

```
mov  al,0F0h                ; AL = 11110000b (-16)
sar  al,1                   ; AL = 11111000b (-8), CF = 0
```

Signed Division

You can divide a signed operand by a power of 2, using the SAR instruction. In the following example, –128 is divided by 23. The quotient is –16:

```
mov  dl,-128                ; DL = 10000000b
sar  dl,3                   ; DL = 11110000b
```

Sign-Extend AX into EAX

Suppose AX contains a signed integer and you want to extend its sign into EAX. First shift EAX 16 bits to the left, then shift it arithmetically 16 bits to the right:

```
08 mov ax, -128     ;EAX = ????FF80h
09 shl eax, 16      ;EAX = FF800000h
10 sar eax, 16      ;EAX = FFFFFF80h
```