x86 Instruction Set Reference

SAL/SAR/SHL/SHR

Shift

Opcode	Mnemonic	Description
D0 /4	SAL r/m8	Multiply r/m8 by 2, 1 time.
D2 /4	SAL r/m8,CL	Multiply r/m8 by 2, CL times.
C0 /4 ib	SAL r/m8,imm8	Multiply r/m8 by 2, imm8 times.
D1 /4	SAL r/m16	Multiply r/m16 by 2, 1 time.
D3 /4	SAL r/m16,CL	Multiply r/m16 by 2, CL times.
C1 /4 ib	SAL r/m16,imm8	Multiply r/m16 by 2, imm8 times.
D1 /4	SAL r/m32	Multiply r/m32 by 2, 1 time.
D3 /4	SAL r/m32,CL	Multiply r/m32 by 2, CL times.
C1 /4 ib	SAL r/m32,imm8	Multiply r/m32 by 2, imm8 times.
D0 /7	SAR r/m8	Signed divide* r/m8 by 2, 1 times.
D2 /7	SAR r/m8,CL	Signed divide* r/m8 by 2, CL times.
C0 /7 ib	SAR r/m8,imm8	Signed divide* r/m8 by 2, imm8 times.
D1 /7	SAR r/m16	Signed divide* r/m16 by 2, 1 time.
D3 /7	SAR r/m16,CL	Signed divide* r/m16 by 2, CL times.
C1 /7 ib	SAR r/m16,imm8	Signed divide* r/m16 by 2, imm8 times.
D1 /7	SAR r/m32	Signed divide* r/m32 by 2, 1 time.
D3 /7	SAR r/m32,CL	Signed divide* r/m32 by 2, CL times.
C1 /7 ib	SAR r/m32,imm8	Signed divide* r/m32 by 2, imm8 times.
D0 /4	SHL r/m8	Multiply r/m8 by 2, 1 time.
D2 /4	SHL r/m8,CL	Multiply r/m8 by 2, CL times.
C0 /4 ib	SHL r/m8,imm8	Multiply r/m8 by 2, imm8 times.
D1 /4	SHL r/m16	Multiply r/m16 by 2, 1 time.
D3 /4	SHL r/m16,CL	Multiply r/m16 by 2, CL times.
C1 /4 ib	SHL r/m16,imm8	Multiply r/m16 by 2, imm8 times.
D1 /4	SHL r/m32	Multiply r/m32 by 2, 1 time.
D3 /4	SHL r/m32,CL	Multiply r/m32 by 2, CL times.
C1 /4 ib	SHL r/m32,imm8	Multiply r/m32 by 2, imm8 times.
D0 /5	SHR r/m8	Unsigned divide r/m8 by 2, 1 time.
D2 /5	SHR r/m8,CL	Unsigned divide r/m8 by 2, CL times.
C0 /5 ib	SHR r/m8,imm8	Unsigned divide r/m8 by 2, imm8 times.
D1 /5	SHR r/m16	Unsigned divide r/m16 by 2, 1 time.
D3 /5	SHR r/m16,CL	Unsigned divide r/m16 by 2, CL times.
C1 /5 ib	SHR r/m16,imm8	Unsigned divide r/m16 by 2, imm8 times.
D1 /5	SHR r/m32	Unsigned divide r/m32 by 2, 1 time.
D3 /5	SHR r/m32,CL	Unsigned divide r/m32 by 2, CL times.
C1 /5 ib	SHR r/m32,imm8	Unsigned divide r/m32 by 2, imm8 times.

Description

NOTE: * Not the same form of division as IDIV; rounding is toward negative infinity.

Shifts the bits in the first operand (destination operand) to the left or right by the number of bits specified in the second operand (count operand). Bits shifted beyond the destination operand boundary are first shifted into the CF flag, then discarded. At the end of the shift operation, the CF flag contains the last bit shifted out of the destination operand.

The destination operand can be a register or a memory location. The count operand can be an immediate value or register CL. The count is masked to 5 bits, which limits the count range to 0 to 31. A special opcode encoding is provided for a count of 1.

The shift arithmetic left (SAL) and shift logical left (SHL) instructions perform the same operation; they shift the bits in the destination operand to the left (toward more significant bit locations).

For each shift count, the most significant bit of the destination operand is shifted into the CF flag, and the least significant bit is cleared (see Figure 7-7 in the IA-32 Intel Architecture Software Developer's Manual, Volume 1).

The shift arithmetic right (SAR) and shift logical right (SHR) instructions shift the bits of the destination operand to the right (toward less significant bit locations). For each shift count, the least significant bit of the destination operand is shifted into the CF flag, and the most significant bit is either set or cleared depending on the instruction type. The SHR instruction clears the most significant bit (see Figure 7-8 in the IA-32 Intel Architecture Software Developer's Manual, Volume 1); the SAR instruction sets or clears the most significant bit to correspond to the sign (most significant bit) of the original value in the destination operand. In effect, the SAR instruction fills the empty bit position's shifted value with the sign of the unshifted value (see Figure 7-9 in the IA-32 Intel Architecture Software Developer's Manual, Volume 1).

The SAR and SHR instructions can be used to perform signed or unsigned division, respectively, of the destination operand by powers of 2. For example, using the SAR instruction to shift a signed integer 1 bit to the right divides the value by 2.

Using the SAR instruction to perform a division operation does not produce the same result as the IDIV instruction. The quotient from the IDIV instruction is rounded toward zero, whereas the "quotient" of the SAR instruction is rounded toward negative infinity. This difference is apparent only for negative numbers. For example, when the IDIV instruction is used to divide -9 by 4, the result is -2 with a remainder of -1. If the SAR instruction is used to shift -9 right by two bits, the result is -3 and the "remainder" is +3; however, the SAR instruction stores only the most significant bit of the remainder (in the CF flag).

The OF flag is affected only on 1-bit shifts. For left shifts, the OF flag is set to 0 if the mostsignificant bit of the result is the same as the CF flag (that is, the top two bits of the original operand were the same); otherwise, it is set to 1. For the SAR instruction, the OF flag is cleared for all 1-bit shifts. For the SHR instruction, the OF flag is set to the most-significant bit of the original operand.

Operation

```
TemporaryCount = Count & 0x1F;
TemporaryDestination = Destination;
while(TemporaryCount != 0) {
        if(Instruction == SAL || Instruction == SHL) {
                CF = MSB(Destination);
                Destination = Destination << 1;
```

```
        }
        //instruction is SAR or SHR
        else {
                CF = LSB(Destination);
                if(Instruction == SAR) Destination = Destination / 2; //Signed divide, rounding toward negative infinity
                //Instruction is SHR
                else Destination = Destination / 2; //Unsigned divide
        }
        TemporaryCount = TemporaryCount - 1;
}
//Determine overflow
if(Count & 0x1F == 1) {
        if(Instruction == SAL || Instruction == SHL) OF = MSB(Destination) ^ CF;
        else if(Instruction == SAR) OF = 0;
        //Instruction == SHR
        else OF = MSB(TemporaryDestination);
}
else OF = Undefined;
```

Flags affected

The CF flag contains the value of the last bit shifted out of the destination operand; it is undefined for SHL and SHR instructions where the count is greater than or equal to the size (in bits) of the destination operand. The OF flag is affected only for 1-bit shifts (see "Description" above); otherwise, it is undefined. The SF, ZF, and PF flags are set according to the result. If the count is 0, the flags are not affected. For a non-zero count, the AF flag is undefined.

IA-32 Architecture Compatibility

The 8086 does not mask the shift count. However, all other IA-32 processors (starting with the Intel 286 processor) do mask the shift count to 5 bits, resulting in a maximum count of 31. This masking is done in all operating modes (including the virtual-8086 mode) to reduce the maximum execution time of the instructions.

Instruction	Latency	Throughput	Execution Unit
CPUID	0F3n/0F2n/069n	0F3n/0F2n	0F2n
SAL/SAR/SHL/SHR	1/4/1	0.5/1	-