

SAL/SAR/SHL/SHR—Shift

Opcode***	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
D0 /4	SAL <i>r/m8</i> , 1	Valid	Valid	Multiply <i>r/m8</i> by 2, once.
REX + D0 /4	SAL <i>r/m8**</i> , 1	Valid	N.E.	Multiply <i>r/m8</i> by 2, once.
D2 /4	SAL <i>r/m8</i> , CL	Valid	Valid	Multiply <i>r/m8</i> by 2, CL times.
REX + D2 /4	SAL <i>r/m8**</i> , CL	Valid	N.E.	Multiply <i>r/m8</i> by 2, CL times.
C0 /4 <i>ib</i>	SAL <i>r/m8</i> , <i>imm8</i>	Valid	Valid	Multiply <i>r/m8</i> by 2, <i>imm8</i> times.
REX + C0 /4 <i>ib</i>	SAL <i>r/m8**</i> , <i>imm8</i>	Valid	N.E.	Multiply <i>r/m8</i> by 2, <i>imm8</i> times.
D1 /4	SAL <i>r/m16</i> , 1	Valid	Valid	Multiply <i>r/m16</i> by 2, once.
D3 /4	SAL <i>r/m16</i> , CL	Valid	Valid	Multiply <i>r/m16</i> by 2, CL times.
C1 /4 <i>ib</i>	SAL <i>r/m16</i> , <i>imm8</i>	Valid	Valid	Multiply <i>r/m16</i> by 2, <i>imm8</i> times.
D1 /4	SAL <i>r/m32</i> , 1	Valid	Valid	Multiply <i>r/m32</i> by 2, once.
REX.W + D1 /4	SAL <i>r/m64</i> , 1	Valid	N.E.	Multiply <i>r/m64</i> by 2, once.
D3 /4	SAL <i>r/m32</i> , CL	Valid	Valid	Multiply <i>r/m32</i> by 2, CL times.
REX.W + D3 /4	SAL <i>r/m64</i> , CL	Valid	N.E.	Multiply <i>r/m64</i> by 2, CL times.
C1 /4 <i>ib</i>	SAL <i>r/m32</i> , <i>imm8</i>	Valid	Valid	Multiply <i>r/m32</i> by 2, <i>imm8</i> times.
REX.W + C1 /4 <i>ib</i>	SAL <i>r/m64</i> , <i>imm8</i>	Valid	N.E.	Multiply <i>r/m64</i> by 2, <i>imm8</i> times.
D0 /7	SAR <i>r/m8</i> , 1	Valid	Valid	Signed divide* <i>r/m8</i> by 2, once.
REX + D0 /7	SAR <i>r/m8**</i> , 1	Valid	N.E.	Signed divide* <i>r/m8</i> by 2, once.
D2 /7	SAR <i>r/m8</i> , CL	Valid	Valid	Signed divide* <i>r/m8</i> by 2, CL times.
REX + D2 /7	SAR <i>r/m8**</i> , CL	Valid	N.E.	Signed divide* <i>r/m8</i> by 2, CL times.
C0 /7 <i>ib</i>	SAR <i>r/m8</i> , <i>imm8</i>	Valid	Valid	Signed divide* <i>r/m8</i> by 2, <i>imm8</i> time.
REX + C0 /7 <i>ib</i>	SAR <i>r/m8**</i> , <i>imm8</i>	Valid	N.E.	Signed divide* <i>r/m8</i> by 2, <i>imm8</i> times.
D1 /7	SAR <i>r/m16</i> , 1	Valid	Valid	Signed divide* <i>r/m16</i> by 2, once.

Opcode	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
D3 /7	SAR <i>r/m16</i> , CL	Valid	Valid	Signed divide* <i>r/m16</i> by 2, CL times.
C1 /7 <i>ib</i>	SAR <i>r/m16</i> , <i>imm8</i>	Valid	Valid	Signed divide* <i>r/m16</i> by 2, <i>imm8</i> times.
D1 /7	SAR <i>r/m32</i> , 1	Valid	Valid	Signed divide* <i>r/m32</i> by 2, once.
REX.W + D1 /7	SAR <i>r/m64</i> , 1	Valid	N.E.	Signed divide* <i>r/m64</i> by 2, once.
D3 /7	SAR <i>r/m32</i> , CL	Valid	Valid	Signed divide* <i>r/m32</i> by 2, CL times.
REX.W + D3 /7	SAR <i>r/m64</i> , CL	Valid	N.E.	Signed divide* <i>r/m64</i> by 2, CL times.
C1 /7 <i>ib</i>	SAR <i>r/m32</i> , <i>imm8</i>	Valid	Valid	Signed divide* <i>r/m32</i> by 2, <i>imm8</i> times.
REX.W + C1 /7 <i>ib</i>	SAR <i>r/m64</i> , <i>imm8</i>	Valid	N.E.	Signed divide* <i>r/m64</i> by 2, <i>imm8</i> times
D0 /4	SHL <i>r/m8</i> , 1	Valid	Valid	Multiply <i>r/m8</i> by 2, once.
REX + D0 /4	SHL <i>r/m8**</i> , 1	Valid	N.E.	Multiply <i>r/m8</i> by 2, once.
D2 /4	SHL <i>r/m8</i> , CL	Valid	Valid	Multiply <i>r/m8</i> by 2, CL times.
REX + D2 /4	SHL <i>r/m8**</i> , CL	Valid	N.E.	Multiply <i>r/m8</i> by 2, CL times.
C0 /4 <i>ib</i>	SHL <i>r/m8</i> , <i>imm8</i>	Valid	Valid	Multiply <i>r/m8</i> by 2, <i>imm8</i> times.
REX + C0 /4 <i>ib</i>	SHL <i>r/m8**</i> , <i>imm8</i>	Valid	N.E.	Multiply <i>r/m8</i> by 2, <i>imm8</i> times.
D1 /4	SHL <i>r/m16</i> , 1	Valid	Valid	Multiply <i>r/m16</i> by 2, once.
D3 /4	SHL <i>r/m16</i> , CL	Valid	Valid	Multiply <i>r/m16</i> by 2, CL times.
C1 /4 <i>ib</i>	SHL <i>r/m16</i> , <i>imm8</i>	Valid	Valid	Multiply <i>r/m16</i> by 2, <i>imm8</i> times.
D1 /4	SHL <i>r/m32</i> , 1	Valid	Valid	Multiply <i>r/m32</i> by 2, once.
REX.W + D1 /4	SHL <i>r/m64</i> , 1	Valid	N.E.	Multiply <i>r/m64</i> by 2, once.
D3 /4	SHL <i>r/m32</i> , CL	Valid	Valid	Multiply <i>r/m32</i> by 2, CL times.
REX.W + D3 /4	SHL <i>r/m64</i> , CL	Valid	N.E.	Multiply <i>r/m64</i> by 2, CL times.
C1 /4 <i>ib</i>	SHL <i>r/m32</i> , <i>imm8</i>	Valid	Valid	Multiply <i>r/m32</i> by 2, <i>imm8</i> times.
REX.W + C1 /4 <i>ib</i>	SHL <i>r/m64</i> , <i>imm8</i>	Valid	N.E.	Multiply <i>r/m64</i> by 2, <i>imm8</i> times.

Opcode	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
D0 /5	SHR <i>r/m8</i> ,1	Valid	Valid	Unsigned divide <i>r/m8</i> by 2, once.
REX + D0 /5	SHR <i>r/m8</i> **, 1	Valid	N.E.	Unsigned divide <i>r/m8</i> by 2, once.
D2 /5	SHR <i>r/m8</i> , CL	Valid	Valid	Unsigned divide <i>r/m8</i> by 2, CL times.
REX + D2 /5	SHR <i>r/m8</i> **, CL	Valid	N.E.	Unsigned divide <i>r/m8</i> by 2, CL times.
C0 /5 <i>ib</i>	SHR <i>r/m8</i> , <i>imm8</i>	Valid	Valid	Unsigned divide <i>r/m8</i> by 2, <i>imm8</i> times.
REX + C0 /5 <i>ib</i>	SHR <i>r/m8</i> **, <i>imm8</i>	Valid	N.E.	Unsigned divide <i>r/m8</i> by 2, <i>imm8</i> times.
D1 /5	SHR <i>r/m16</i> , 1	Valid	Valid	Unsigned divide <i>r/m16</i> by 2, once.
D3 /5	SHR <i>r/m16</i> , CL	Valid	Valid	Unsigned divide <i>r/m16</i> by 2, CL times
C1 /5 <i>ib</i>	SHR <i>r/m16</i> , <i>imm8</i>	Valid	Valid	Unsigned divide <i>r/m16</i> by 2, <i>imm8</i> times.
D1 /5	SHR <i>r/m32</i> , 1	Valid	Valid	Unsigned divide <i>r/m32</i> by 2, once.
REX.W + D1 /5	SHR <i>r/m64</i> , 1	Valid	N.E.	Unsigned divide <i>r/m64</i> by 2, once.
D3 /5	SHR <i>r/m32</i> , CL	Valid	Valid	Unsigned divide <i>r/m32</i> by 2, CL times.
REX.W + D3 /5	SHR <i>r/m64</i> , CL	Valid	N.E.	Unsigned divide <i>r/m64</i> by 2, CL times.
C1 /5 <i>ib</i>	SHR <i>r/m32</i> , <i>imm8</i>	Valid	Valid	Unsigned divide <i>r/m32</i> by 2, <i>imm8</i> times.
REX.W + C1 /5 <i>ib</i>	SHR <i>r/m64</i> , <i>imm8</i>	Valid	N.E.	Unsigned divide <i>r/m64</i> by 2, <i>imm8</i> times.

NOTES:

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

** In 64-bit mode, *r/m8* can not be encoded to access the following byte registers if a REX prefix is used: AH, BH, CH, DH.

***See IA-32 Architecture Compatibility section below.

Description

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 the CL register. The count is masked to 5 bits (or 6 bits if in 64-bit mode and REX.W is used). The count range is limited to 0 to 31 (or 63 if 64-bit mode and REX.W is used). 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 *Intel® 64 and IA-32 Architectures 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 *Intel® 64 and IA-32 Architectures 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 *Intel® 64 and IA-32 Architectures 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 most-significant 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.

In 64-bit mode, the instruction's default operation size is 32 bits and the mask width for CL is 5 bits. Using a REX prefix in the form of REX.R permits access to additional registers (R8-R15). Using a REX prefix in the form of REX.W promotes operation to 64-bits and sets the mask width for CL to 6 bits. See the summary chart at the beginning of this section for encoding data and limits.

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.

Operation

IF 64-Bit Mode and using REX.W

THEN

countMASK \leftarrow 3FH;

ELSE

countMASK \leftarrow 1FH;

FI

tempCOUNT \leftarrow (COUNT AND countMASK);

tempDEST \leftarrow DEST;

WHILE (tempCOUNT \neq 0)

DO

IF instruction is SAL or SHL

THEN

CF \leftarrow MSB(DEST);

ELSE (* Instruction is SAR or SHR *)

CF \leftarrow LSB(DEST);

FI;

IF instruction is SAL or SHL

THEN

DEST \leftarrow DEST * 2;

ELSE

IF instruction is SAR

THEN

DEST \leftarrow DEST / 2; (* Signed divide, rounding toward negative infinity *)

ELSE (* Instruction is SHR *)

DEST \leftarrow DEST / 2; (* Unsigned divide *)

FI;

FI;

tempCOUNT \leftarrow tempCOUNT - 1;

OD;