## XOR—Logical Exclusive OR

| Opcode | Instruction | 64-Bit Mode | Compat/ Leg Mode | Description |
|---|---|---|---|---|
| 34 *ib* | XOR AL, *imm8* | Valid | Valid | AL XOR *imm8.* |
| 35 *iw* | XOR AX, *imm16* | Valid | Valid | AX XOR *imm16.* |
| 35 *id* | XOR EAX, *imm32* | Valid | Valid | EAX XOR *imm32.* |
| REX.W + 35 *id* | XOR RAX, *imm32* | Valid | N.E. | RAX XOR *imm32 (sign-extended).* |
| 80 /6 *ib* | XOR *r/m8, imm8* | Valid | Valid | *r/m8* XOR *imm8.* |
| REX + 80 /6 *ib* | XOR *r/m8*, imm8* | Valid | N.E. | *r/m8* XOR *imm8.* |
| 81 /6 *iw* | XOR *r/m16, imm16* | Valid | Valid | *r/m16* XOR *imm16.* |
| 81 /6 *id* | XOR *r/m32, imm32* | Valid | Valid | *r/m32* XOR *imm32.* |
| REX.W + 81 /6 *id* | XOR *r/m64, imm32* | Valid | N.E. | *r/m64* XOR *imm32 (sign-extended).* |
| 83 /6 *ib* | XOR *r/m16, imm8* | Valid | Valid | *r/m16* XOR *imm8 (sign-extended).* |
| 83 /6 *ib* | XOR *r/m32, imm8* | Valid | Valid | *r/m32* XOR *imm8 (sign-extended).* |
| REX.W + 83 /6 *ib* | XOR *r/m64, imm8* | Valid | N.E. | *r/m64* XOR *imm8 (sign-extended).* |
| 30 /*r* | XOR *r/m8, r8* | Valid | Valid | *r/m8* XOR *r8.* |
| REX + 30 /*r* | XOR *r/m8*, r8** | Valid | N.E. | *r/m8* XOR *r8.* |
| 31 /*r* | XOR *r/m16, r16* | Valid | Valid | *r/m16* XOR *r16.* |
| 31 /*r* | XOR *r/m32, r32* | Valid | Valid | *r/m32* XOR *r32.* |
| REX.W + 31 /*r* | XOR *r/m64, r64* | Valid | N.E. | *r/m64* XOR *r64.* |
| 32 /*r* | XOR *r8, r/m8* | Valid | Valid | *r8* XOR *r/m8.* |
| REX + 32 /*r* | XOR *r8*, r/m8** | Valid | N.E. | *r8* XOR *r/m8.* |
| 33 /*r* | XOR *r16, r/m16* | Valid | Valid | *r16* XOR *r/m16.* |
| 33 /*r* | XOR *r32, r/m32* | Valid | Valid | *r32* XOR *r/m32.* |
| REX.W + 33 /*r* | XOR *r64, r/m64* | Valid | N.E. | *r64* XOR *r/m64.* |

**NOTES:**

* 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.

## Description

Performs a bitwise exclusive OR (XOR) operation on the destination (first) and source (second) operands and stores the result in the destination operand location. The source operand can be an immediate, a register, or a memory location; the destination operand can be a register or a memory location. (However, two memory operands cannot be used in one instruction.) Each bit of the result is 1 if the corresponding bits of the operands are different; each bit is 0 if the corresponding bits are the same.

This instruction can be used with a LOCK prefix to allow the instruction to be executed atomically.

In 64-bit mode, 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. See the summary chart at the beginning of this section for encoding data and limits.

## Operation

```
DEST ← DEST XOR SRC;
```

## Flags Affected

The OF and CF flags are cleared; the SF, ZF, and PF flags are set according to the result. The state of the AF flag is undefined.

## Protected Mode Exceptions

| | |
|---|---|
| #GP(0) | If the destination operand points to a non-writable segment. |
| | If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. |
| | If the DS, ES, FS, or GS register contains a NULL segment selector. |
| #SS(0) | If a memory operand effective address is outside the SS segment limit. |
| #PF(fault-code) | If a page fault occurs. |
| #AC(0) | If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3. |
| #UD | If the LOCK prefix is used but the destination is not a memory operand. |

## Real-Address Mode Exceptions

| | |
|---|---|
| #GP | If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. |
| #SS | If a memory operand effective address is outside the SS segment limit. |

| | |
|---|---|
| #UD | If the LOCK prefix is used but the destination is not a memory operand. |

### Virtual-8086 Mode Exceptions

| | |
|---|---|
| #GP(0) | If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. |
| #SS(0) | If a memory operand effective address is outside the SS segment limit. |
| #PF(fault-code) | If a page fault occurs. |
| #AC(0) | If alignment checking is enabled and an unaligned memory reference is made. |
| #UD | If the LOCK prefix is used but the destination is not a memory operand. |

### Compatibility Mode Exceptions

Same exceptions as in protected mode.

### 64-Bit Mode Exceptions

| | |
|---|---|
| #SS(0) | If a memory address referencing the SS segment is in a non-canonical form. |
| #GP(0) | If the memory address is in a non-canonical form. |
| #PF(fault-code) | If a page fault occurs. |
| #AC(0) | If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3. |
| #UD | If the LOCK prefix is used but the destination is not a memory operand. |