AND—Logical AND

Opcode	Instruction	64-Bit Mode	Comp/Leg Mode	Description
24 *ib*	AND AL, *imm8*	Valid	Valid	AL AND *imm8.*
25 *iw*	AND AX, *imm16*	Valid	Valid	AX AND i*mm16.*
25 *id*	AND EAX, *imm32*	Valid	Valid	EAX AND *imm32.*
REX.W + 25 *id*	AND RAX, *imm32*	Valid	N.E.	RAX AND *imm32 sign-extended to 64-bits.*
80 /4 *ib*	AND *r/m8, imm8*	Valid	Valid	*r/m8* AND *imm8.*
REX + 80 /4 *ib*	AND *r/m8**, *imm8*	Valid	N.E.	*r/m64* AND *imm8 (sign-extended).*
81 /4 *iw*	AND *r/m16, imm16*	Valid	Valid	*r/m16* AND *imm16.*
81 /4 *id*	AND *r/m32, imm32*	Valid	Valid	*r/m32* AND *imm32.*
REX.W + 81 /4 *id*	AND *r/m64, imm32*	Valid	N.E.	*r/m64* AND *imm32 sign extended to 64-bits.*
83 /4 *ib*	AND *r/m16, imm8*	Valid	Valid	*r/m16* AND *imm8 (sign-extended).*
83 /4 *ib*	AND *r/m32, imm8*	Valid	Valid	*r/m32* AND *imm8 (sign-extended).*
REX.W + 83 /4 *ib*	AND *r/m64, imm8*	Valid	N.E.	*r/m64* AND *imm8 (sign-extended).*
20 /*r*	AND *r/m8, r8*	Valid	Valid	*r/m8* AND *r8.*
REX + 20 /*r*	AND *r/m8**, *r8**	Valid	N.E.	*r/m64* AND *r8 (sign-extended).*
21 /*r*	AND *r/m16, r16*	Valid	Valid	*r/m16* AND *r16.*
21 /*r*	AND *r/m32, r32*	Valid	Valid	*r/m32* AND *r32.*
REX.W + 21 /*r*	AND *r/m64, r64*	Valid	N.E.	*r/m64* AND *r32.*
22 /*r*	AND *r8, r/m8*	Valid	Valid	*r8* AND *r/m8.*
REX + 22 /*r*	AND *r8**, *r/m8**	Valid	N.E.	*r/m64* AND *r8 (sign-extended).*
23 /*r*	AND *r16, r/m16*	Valid	Valid	*r16* AND *r/m16.*
23 /*r*	AND *r32, r/m32*	Valid	Valid	*r32* AND *r/m32.*
REX.W + 23 /*r*	AND *r64, r/m64*	Valid	N.E.	*r64* AND *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 AND 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 set to 1 if both corresponding bits of the first and second operands are 1; otherwise, it is set to 0.

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

In 64-bit mode, the instruction's default operation size is 32 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. See the summary chart at the beginning of this section for encoding data and limits.

Operation

```
DEST ← DEST AND 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.