

DATA TYPES

Table 4-1. Signed Integer Encodings

Class		Two's Complement Encoding	
		Sign	
Positive	Largest	0	11..11
		.	.
	Smallest	0	00..01
Zero		0	00..00
Negative	Smallest	1	11..11
		.	.
	Largest	1	00..00
Integer indefinite		1	00..00
		Signed Byte Integer:	← 7 bits →
		Signed Word Integer:	← 15 bits →
		Signed Doubleword Integer:	← 31 bits →
		Signed Quadword Integer:	← 63 bits →

The sign bit is set for negative integers and cleared for positive integers and zero. Integer values range from -128 to $+127$ for a byte integer, from $-32,768$ to $+32,767$ for a word integer, from -2^{31} to $+2^{31} - 1$ for a doubleword integer, and from -2^{63} to $+2^{63} - 1$ for a quadword integer.

When storing integer values in memory, word integers are stored in 2 consecutive bytes; doubleword integers are stored in 4 consecutive bytes; and quadword integers are stored in 8 consecutive bytes.

The integer indefinite is a special value that is sometimes returned by the x87 FPU when operating on integer values. For more information, see Section 8.2.1, "Indefinites."

4.2.2 Floating-Point Data Types

The IA-32 architecture defines and operates on three floating-point data types: single-precision floating-point, double-precision floating-point, and double-extended precision floating-point (see Figure 4-3). The data formats for these data types correspond directly to formats specified in the IEEE Standard 754 for Binary Floating-Point Arithmetic.

Table 4-2 gives the length, precision, and approximate normalized range that can be represented by each of these data types. Denormal values are also supported in each of these types.

Table 4-2. Length, Precision, and Range of Floating-Point Data Types

Data Type	Length	Precision (Bits)	Approximate Normalized Range	
			Binary	Decimal
Single Precision	32	24	2^{-126} to 2^{127}	1.18×10^{-38} to 3.40×10^{38}
Double Precision	64	53	2^{-1022} to 2^{1023}	2.23×10^{-308} to 1.79×10^{308}
Double Extended Precision	80	64	2^{-16382} to 2^{16383}	3.37×10^{-4932} to 1.18×10^{4932}

NOTE

Section 4.8, "Real Numbers and Floating-Point Formats," gives an overview of the IEEE Standard 754 floating-point formats and defines the terms integer bit, QNaN, SNaN, and denormal value.

Table 4-3 shows the floating-point encodings for zeros, denormalized finite numbers, normalized finite numbers, infinities, and NaNs for each of the three floating-point data types. It also gives the format for the QNaN floating-point indefinite value. (See Section 4.8.3.7, "QNaN Floating-Point Indefinite," for a discussion of the use of the QNaN floating-point indefinite value.)

For the single-precision and double-precision formats, only the fraction part of the significand is encoded. The integer is assumed to be 1 for all numbers except 0 and denormalized finite numbers. For the double extended-precision format, the integer is contained in bit 63, and the most-significant fraction bit is bit 62. Here, the integer is explicitly set to 1 for normalized numbers, infinities, and NaNs, and to 0 for zero and denormalized numbers.

DATA TYPES

Table 4-3. Floating-Point Number and NaN Encodings

Class		Sign	Biased Exponent	Significand	
				Integer ¹	Fraction
Positive	+∞	0	11..11	1	00..00
	+Normals	0	11..10	1	11..11
		.	.	.	.
		0	00..01	1	00..00
	+Denormals	0	00..00	0	11..11
		.	.	.	.
		0	00..00	0	00..01
	+Zero	0	00..00	0	00..00
Negative	−Zero	1	00..00	0	00..00
	−Denormals	1	00..00	0	00..01
		.	.	.	.
		1	00..00	0	11..11
	−Normals	1	00..01	1	00..00
		.	.	.	.
		1	11..10	1	11..11
	−∞	1	11..11	1	00..00
NaNs	SNaN	X	11..11	1	0X..XX ²
	QNaN	X	11..11	1	1X..XX
	QNaN Floating-Point Indefinite	1	11..11	1	10..00
Single-Precision:		← 8 Bits →		← 23 Bits →	
Double-Precision:		← 11 Bits →		← 52 Bits →	
Double Extended-Precision:		← 15 Bits →		← 63 Bits →	

NOTES:

1. Integer bit is implied and not stored for single-precision and double-precision formats.
2. The fraction for SNaN encodings must be non-zero with the most-significant bit 0.

The exponent of each floating-point data type is encoded in biased format; see Section 4.8.2.2, “Biased Exponent.” The biasing constant is 127 for the single-precision format, 1023 for the double-precision format, and 16,383 for the double extended-precision format.

When storing floating-point values in memory, single-precision values are stored in 4 consecutive bytes in memory; double-precision values are stored in 8 consecutive bytes; and double extended-precision values are stored in 10 consecutive bytes.

The single-precision and double-precision floating-point data types are operated on by x87 FPU, and SSE/SSE2/SSE3 instructions. The double-extended-precision floating-point format is only operated on by the x87 FPU. See Section 11.6.8, "Compatibility of SIMD and x87 FPU Floating-Point Data Types," for a discussion of the compatibility of single-precision and double-precision floating-point data types between the x87 FPU and SSE/SSE2/SSE3 extensions.

4.3 POINTER DATA TYPES

Pointers are addresses of locations in memory.

In non-64-bit modes, the architecture defines two types of pointers: a **near pointer** and a **far pointer**. A near pointer is a 32-bit (or 16-bit) offset (also called an **effective address**) within a segment. Near pointers are used for all memory references in a flat memory model or for references in a segmented model where the identity of the segment being accessed is implied.

A far pointer is a logical address, consisting of a 16-bit segment selector and a 32-bit (or 16-bit) offset. Far pointers are used for memory references in a segmented memory model where the identity of a segment being accessed must be specified explicitly. Near and far pointers with 32-bit offsets are shown in Figure 4-4.

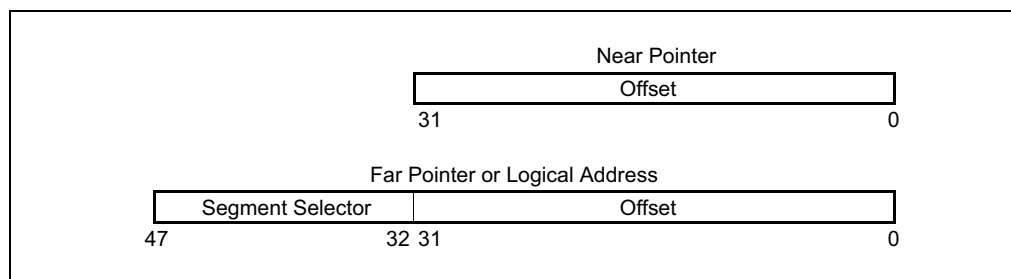


Figure 4-4. Pointer Data Types

4.3.1 Pointer Data Types in 64-Bit Mode

In 64-bit mode (a sub-mode of IA-32e mode), a **near pointer** is 64 bits. This equates to an effective address. **Far pointers** in 64-bit mode can be one of three forms:

- 16-bit segment selector, 16-bit offset if the operand size is 32 bits
- 16-bit segment selector, 32-bit offset if the operand size is 32 bits
- 16-bit segment selector, 64-bit offset if the operand size is 64 bits

See Figure 4-5.