Intrinsic Datatypes

Summary of intrinsic data types in assembly language:

• Intrinsic data types describe the size, signed or unsigned nature, and whether they represent integers or real numbers.
• The assembler cares about operand size, but does not enforce the distinction between signed and unsigned integers.
• Programmers often use SDWORD to indicate signedness, but this is not required.
• Intrinsic data types help describe how data is stored and used in assembly, with a focus on size and, sometimes, signedness.

Key takeaways:

• Intrinsic data types are important for understanding how data is stored and used in assembly language.
• The assembler cares about operand size, so it is important to choose the correct data type for your variables and expressions.
• Programmers often use SDWORD to indicate signedness, but this is not required.

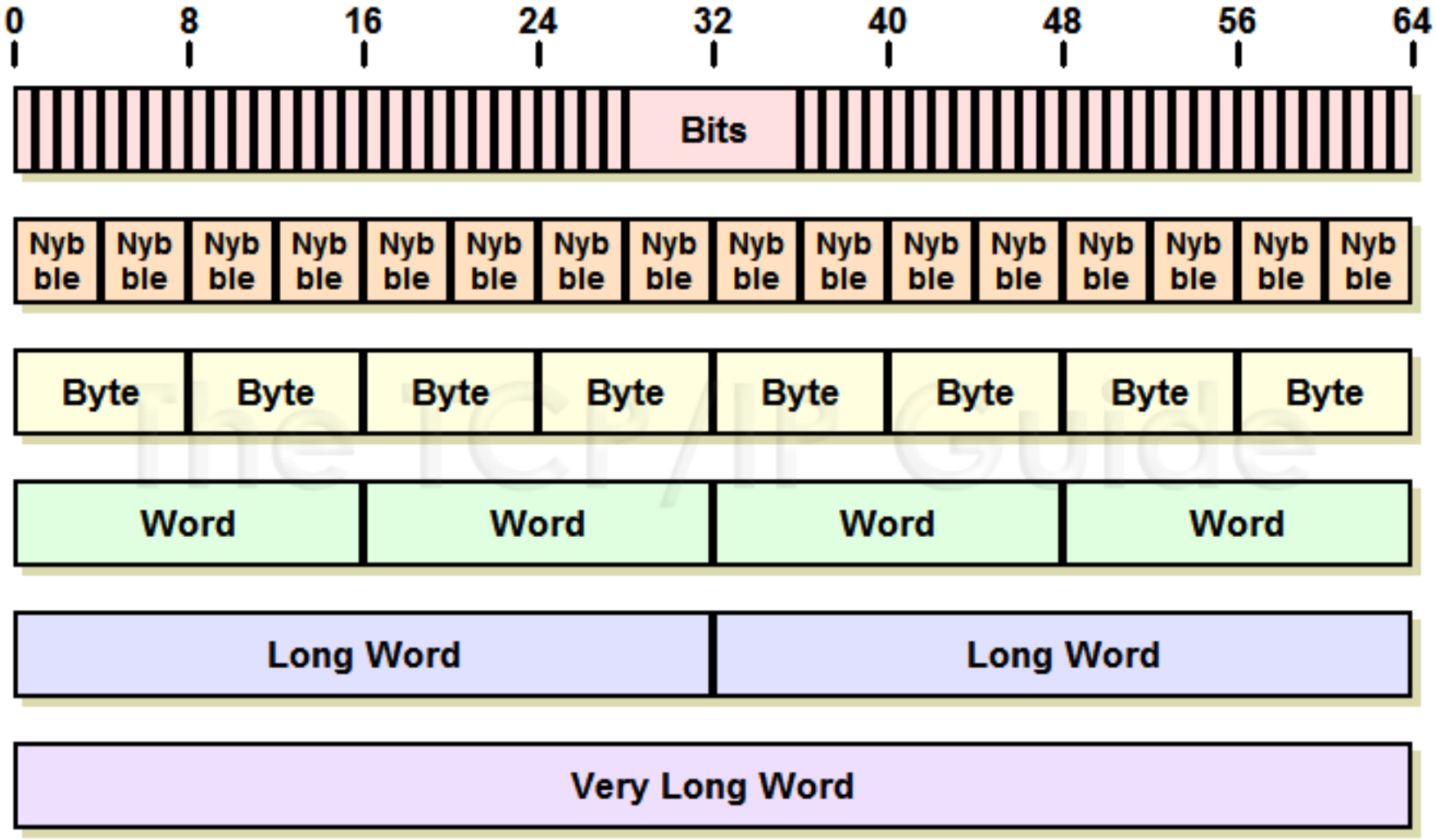

Figure 8: Binary Information Representations and Terms

This diagram shows the relative sizes of the most commonly-sized "collections" of binary information.

Some types, like **DWORD (32-bit unsigned integer)** and **SDWORD (32-bit signed integer)**, overlap in

functionality.

Programmers often use SDWORD to indicate that a value has a sign, but the assembler doesn't enforce this distinction. It mainly considers the size of operands.

So, when i say intrinsic datatypes i mean those listed in that image??

Yes, when you say intrinsic datatypes, you mean those listed in the image you sent. These are the most common data types used in computing, and they are represented in binary.

- **Bit:** The smallest unit of information, represented by a 0 or a 1.
- **Nibble:** A group of four bits, which can be used to represent a single hexadecimal digit.
- **Byte:** A group of eight bits, which can be used to represent a single character or a small number.
- **Word:** A group of 16 bits, which can be used to represent a larger number or a pair of bytes.
- **Double Word:** A group of 32 bits, which can be used to represent a very large number or a pair of words.
- **Quad Word:** A group of 64 bits, which can be used to represent an even larger number or a pair of double words.

These datatypes are used to store and represent all sorts of data in computers, from images and videos to text and numbers.

Here is a summary of the intrinsic data types in assembly language, without a table and with explanations, written each as its own point:

- **BYTE:** An 8-bit unsigned integer. This means that it can store a value from 0 to 255.

- **SBYTE:** An 8-bit signed integer. This means that it can store a value from -128 to 127.

- **WORD:** A 16-bit unsigned integer. This means that it can store a value from 0 to 65535.
- **SWORD:** A 16-bit signed integer. This means that it can store a value from -32768 to 32767.

- **DWORD:** A 32-bit unsigned integer. This means that it can store a value from 0 to 4,294,967,295.
- **SDWORD:** A 32-bit signed integer. This means that it can store a value from -2,147,483,648 to 2,147,483,647.

- **FWORD:** A 48-bit integer. This is used for far pointers in protected mode.
- **QWORD:** A 64-bit integer. This is the largest integer data type in assembly language.

- **TBYTE:** An 80-bit integer. This is rarely used, but it is available on some processors.
- **REAL4:** A 32-bit **floating-point** number. This is the most common floating-point data type in assembly language.

- **REAL8:** A 64-bit **floating-point** number. This is used for high-precision calculations.
- **REAL10:** An 80-bit **floating-point** number. This is rarely used, but it is available on some processors.

It is important to note that the assembler does not enforce the distinction between signed and unsigned integers. However, programmers often use SDWORD to indicate that a value has a sign. This is because it can help to avoid errors and make code more readable.