Packed BCD and TBYTE

PACKED BCD AND TBYTE

Packed BCD data is a way of representing decimal numbers in binary form. It is more compact than representing decimal numbers as binary integers, because it stores two decimal digits in each byte.

The **TBYTE directive** in MASM is used to declare packed BCD variables. To declare a packed BCD variable, you use the following syntax:

```
TBYTE variable_name
my_bcd_variable TBYTE
```

Constant initializers for packed BCD variables must be in hexadecimal, because the assembler does not automatically translate decimal initializers to BCD.

For example, the following code initializes the my_bcd_variable variable to the decimal value 1234:

```
my_bcd_variable TBYTE 1234h
```

The following example is invalid, because the assembler will encode the constant as a binary integer rather than a packed BCD integer:

```
my_bcd_variable TBYTE -1234
```

Here is a table showing the hexadecimal storage bytes for positive and negative decimal 1234:

Decimal Value	Storage Bytes
+1234	34 12 00 00 00 00 00 00 00 00
−1234	34 12 00 00 00 00 00 00 00 80

Repeat:

Packed BCD: Packed BCD is a way of representing decimal numbers in binary form, where each byte (except the highest one) contains two decimal digits. The lower 9 bytes store the decimal digits, and the highest byte indicates the number's sign.

Packed Binary Coded Decimal is a way of representing decimal numbers in a binary form. In this format, each byte contains two decimal digits. It's called "packed" because it packs two decimal digits into each byte. For example, the number 1234 is represented as 34 12 in packed BCD.

• Sign Encoding: In the highest byte, the highest bit is used to indicate the number's sign. If the highest byte equals 80h (hexadecimal), the number is negative, and if it equals 00h, the number is positive.

• Integer Range: Packed BCD can represent a wide range of decimal integers, from -999,999,999,999,999,999 to +999,999,999,999,999,999. Example: For example, let's consider the decimal number 1234:

In packed BCD, this is represented as:

```
34 12 00 00 00 00 00 00 00 00  ;for the positive value.

34 12 00 00 00 00 00 00 00 80  ;for the negative value.
```

TBYTE Directive:

TBYTE is a directive in assembly language, but it doesn't specifically refer to 10 bytes(80 bits). Instead, it's used to declare variables that can hold different types of data, including packed BCD.

In MASM (Microsoft Macro Assembler), you use the TBYTE directive to declare packed BCD variables. When initializing such variables with constants, the values must be in hexadecimal form because MASM doesn't automatically translate decimal initializers to packed BCD. For example, to represent the decimal -1234 as packed BCD in MASM, you should use hexadecimal notation like this:

```
intVal TBYTE 80000000000001234h ; valid
```

In essence, packed BCD is a way to store decimal numbers efficiently in binary form. The TBYTE directive in assembly allows you to work with variables that can hold such packed BCD data.

So, when you see TBYTE used in assembly code, it's not necessarily referring to a fixed size of 80 bytes; instead, it's indicating that the variable can hold data in different formats, including packed BCD.

Repeat 2:

Packed BCD stands for **Packed Binary Coded Decimal**. It is a way of representing decimal numbers in binary form, where each byte stores two decimal digits. This makes it more compact than representing decimal numbers as binary integers, because it uses half the number of bytes.

The **TBYTE** directive in MASM is used to declare packed BCD variables. It is a 10-byte type, with the first byte being the sign byte and the remaining 9 bytes being used to store the decimal digits.

The hexadecimal storage bytes that you provided are correct for the positive and negative values of 1234 in packed BCD format. The first byte of the positive value is 00h, which indicates that the number is positive. The next four bytes contain the decimal digits 3, 4, 1, and 2, in that order.

The first byte of the negative value is 80h, which indicates that the number is negative. The next four bytes contain the decimal digits 3, 4, 1, and 2, in that order.

Why are we doing this?

Packed BCD data is often used in financial and scientific applications, where it is important to represent decimal numbers accurately and efficiently. It is also used in some embedded systems, where memory is limited.

In the context of your question, I am not sure why you are doing this. It is possible that you are working on a financial or scientific application, or on an embedded system. However, without more information, I cannot say for sure.

DECLARING PACKED BCD

Let's clarify the notes on declaring packed BCD variables in MASM (Microsoft Macro Assembler) and converting real numbers to packed BCD:

TBYTE Directive for Packed BCD:

In MASM, you can use the TBYTE directive to declare packed BCD variables. Packed BCD represents decimal numbers efficiently in binary form.

When initializing packed BCD variables with constant values, those constants must be in hexadecimal format.

MASM doesn't automatically translate decimal initializers to packed BCD. Here are examples:

```
intVal TBYTE 8000000000000001234h ; Valid - Represents packed BCD for decimal -1234
intVal TBYTE -1234 ; Invalid - MASM interprets this as a binary integer, not packed BCD
```

The second example is invalid because MASM encodes the constant as a binary integer rather than a packed BCD integer.

Converting Real Numbers to Packed BCD: If you want to convert a real number to packed BCD, you can use the FPU (Floating-Point Unit) instructions. Here's an example:

```
.data
posVal REAL8 1.5     ;Define a real number
bcdVal TBYTE ?       ;Declare a packed BCD variable

.code
fld posVal       ;Load the real number onto the floating point stack
fbstp bcdVal     ;Convert it to packed BCD and store it in bcdVal

;If posVal were equal to 1.5, the resulting BCD value would be 2.
```

In this code, we first define a real number posVal. Then, we declare a packed BCD variable bcdVal. We use the FLD instruction to load posVal onto the floating-point stack and the FBSTP instruction to convert it to packed BCD, rounding to the nearest integer. If posVal were equal to 1.5, the

resulting BCD value in bcdVal would be 2.

Packed BCD (Binary Coded Decimal) is like a special way of writing down numbers in a computer. Imagine you have a calculator, and you want to write numbers in a way that's easy for the computer to understand when doing math. Packed BCD helps with that.

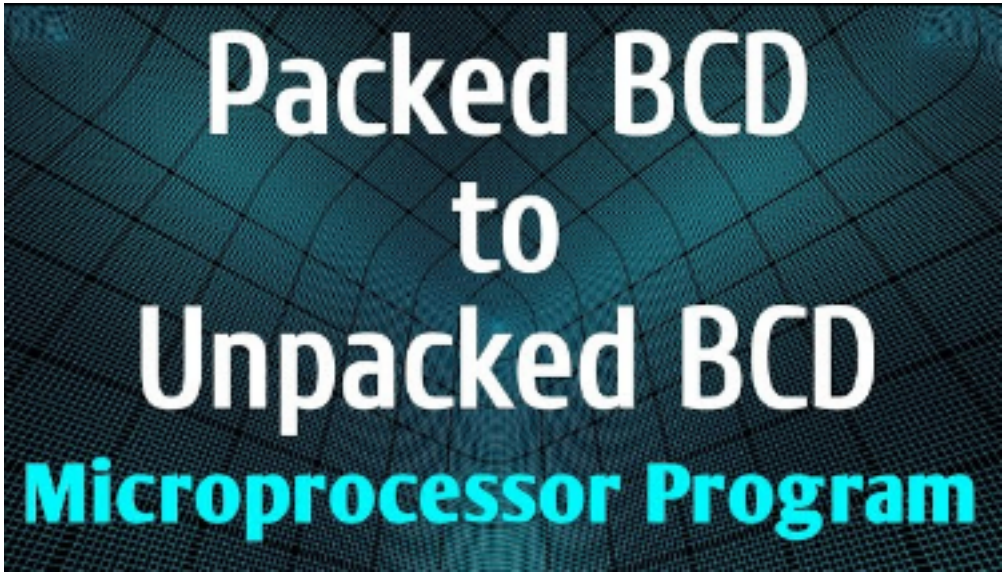
Here's how it works:

1. Digits in Pairs: In regular numbers, we have 0 to 9. In Packed BCD, we group these numbers in pairs. So, for example, 0-0, 1-1, 2-2, and so on up to 9-9.

2. Sign Indicator: We also have a special way to show if a number is positive or negative. If it starts with '80' in Packed BCD, it's negative, and if it starts with '00', it's positive.

3. Efficient Storage: Packed BCD helps computers store numbers more efficiently. Each pair of digits is stored in one byte (8 bits). So, even if it's 2 digits, it takes just one byte.

4. Decimal Math: Packed BCD makes it easier for computers to do decimal math, like adding, subtracting, multiplying, and dividing, because the numbers are organized in pairs.



So, think of Packed BCD as a way for computers to handle numbers neatly, especially when dealing with money or other decimal-based calculations. It's like having numbers with a specific structure that the computer can easily work with.