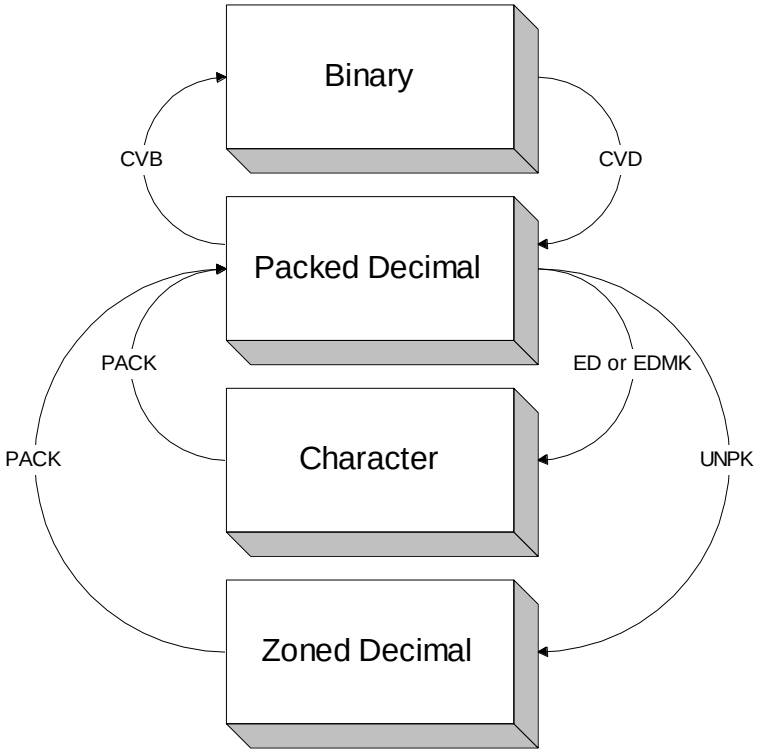
DATA CONVERSIONS

Programmers often find it necessary to change data from one data type to another. We will call this process “data conversion”. There are a number of instructions devoted to data conversion that we will examine; specifically, we will discuss conversion instructions involving four important data types: Zoned Decimal, Character, Packed Decimal, and Binary data types. The following diagram illustrates all possible conversions among the 4 data types, and the instructions necessary to perform the conversions.



Converting from Zoned and Character to Packed

Zoned decimal data and character data that contains digits have closely related data formats. (See **Zoned Decimal Data and Character Data**.) For example, consider the definitions below.

```
CFIELD    DC   C'12345'    CFIELD = X'F1F2F3F4F5'
ZFIELD    DC   Z'12345'    ZFIELD = X'F1F2F3F4C5'
```

In fact, the only difference above occurs in the zone portion of the rightmost byte. The zoned field contains a “C” while the character field contains an “F”. In a zoned format both “C” and “F” represent a positive sign (+) and so CFIELD AND ZFIELD are equivalent as zoned decimal fields. As a result, the **PACK** instruction can be used to convert both of these fields to a packed format:

```
          CPACKED   DS     PL3
          ZPACKED   DS     PL4
          DBLWD     DS     D
                    ...
                    PACK   CPACKED,CFIELD    CPACKED = X'12345F'
                    PACK   ZPACKED,ZFIELD    ZPACKED = X'0012345C'
                    PACK   DBLWD,CFIELD      DBLWD =
X'000000000012345F'
```

In converting the character field to packed decimal in the first example above, we decided to use a 3 byte field to hold the result. This decision is somewhat arbitrary. Since the character field contained 5 digits, it takes at least 3 bytes to hold the result in a packed format (2 packed digits per byte plus the sign). A smaller choice would have caused truncation of digits on the left. A larger field could have been chosen which would have resulted in the field being padded with zeros on the left. This in fact, happened when we packed the zoned field into a 4 byte field and again when we packed into a doubleword. There is a good reason we might want to pack a field into a doubleword and that is if our goal is to ultimately convert to binary format. This is discussed in the next topic.

Converting from Packed Decimal to Binary

In converting to Binary with the **CVB** instruction, there is a requirement that the packed field be a properly aligned doubleword. The range of integers which can be successfully converted is -2,147,483,648 to 2,147,483,647. Many programmers will use a work area for this purpose and **PACK** or **ZAP** the doubleword with the required field before converting to binary in a register:

```
                    ZAP   DBLWD,PFIELD  TRANSFER FIELD TO DBLWD
                    CVB   R6,DBLWD      BINARY VERSION TO R6
                    ...
          PFIELD    DC    P'2345'
          DBLWD     DS    D     DOUBLEWORD WORK AREA
```

In the example above "PFIELD" contains a packed decimal value and we wish to convert this value to 2's complement binary in a register. First the field is moved to a double word "staging" area called "DBLWD". Since PFIELD is packed, the field is transferred to DBLWD with a **ZAP** rather than a **MVC**. The **CVB** instruction performs the conversion from packed to binary.

Converting from Binary to Packed Decimal

The requirements for converting from binary to packed decimal using the **CVD** instruction are similar to those for using **CVB**. In this case, the field that is to receive the packed decimal field must be a doubleword. Since a doubleword can hold up to 15 packed digits, **any** value in a register can be converted to packed decimal. A small problem arises after the conversion if we wish to edit the data into a printable format. Since the doubleword will contain 15 digits, editing the result can be a bit cumbersome. A common practice is to **ZAP** the double word into a smaller packed field that could hold the result before proceeding to edit the data:

```
                    CVD    R6,DBLWD         CHANGE TO DECIMAL
                    ZAP    SMALLPK,DBLWD    DBLWD FIELD IS TOO
LARGE
                    ...
          DBLWD     DS     D
          SMALLPK   DS     PL4
```

The size of the smaller packed field is somewhat arbitrary. It must be big enough to hold the maximum integer you might process. Know your data!

Converting from Packed Decimal to Character

Preparing a packed field for printing involves the creation of an appropriate edit word and the use of the **ED** or **EDMK** instruction. This is an error-prone process for many beginning assembler programmers and many headaches could be avoided by paying attention to a few details:

1) Start with the packed field and count the number of packed digits it contains.

2) Create an edit word for the packed field. Count the number of X'20's and X'21's that appear in the edit word. The count in step 2 **must equal** the count in step 1. If you make a mistake with this, the results are highly unpredictable!

3) Count the number of **bytes** in the edit word. Count every byte including the fill character. Now create an output field that is exactly the size of the edit word. Again, a mistake here is fatal.

4) At execution time, move the edit word to the output field and edit the data.

For example, assume we want to edit the packed field below into a dollars and cents format.

```
PKFIELD   DC    PL4'543210'   PKFIELD = X'0543210C'
```

Examining PKFIELD we see that it contains 7 packed digits. This means that the edit word we create must contain 7 X'20's or X'21's. The following edit word will work.

```
EDWD      DC    X'4020206B2020214B2020'
```

Counting the bytes in EDWD we see it contains 10. Next we create an output field capable of containing the edit word.

```
POUT      DS    CL10
```

Now we are ready to edit the data.

```
MVC       PKOUT,EDWD
ED        POUT,PKFIELD
```

The hexadecimal contents of PKOUT after executing the code above is X'4040F5F46BF3F24BF1F0' and would appear in a printed report as ' 5,432.10'.

Converting from Packed Decimal to Zoned Decimal

The conversion to zoned from packed is a simple and straightforward use of the **UNPK** instruction. Be sure to provide a field big enough to hold the result. Consider the following conversion.

```
          UNPK  ZFIELD,PKFIELD
          ...
PKFIELD   DC    P'112233' PKFIELD = X'0112233C'
ZFIELD    DS    ZL7
```

After execution, ZFIELD contains X'F0F1F1F2F2F3C3'.

A COMPLETE EXAMPLE

The following code demonstrates how a single zoned decimal field (ZFIELD) can be converted to the other three data types.

```
            PACK      PKFIELD,ZFIELD     NOW WE HAVE A PACKED VERSION
            ZAP       DBLWD,PKFIELD      PREPARE TO CONVERT TO ...
            CVB       R8,DBLWD           ...A BINARY VERSION IN R8
            MVC       CHOUT,EDWD         GET READY TO EDIT...
            ED        CHOUT,PKFIELD      ... A CHARACTER VERSION IN
CHOUT
            UNPK      ZOUT,PKFIELD       BACK TO A ZONED VERSION IN ZOUT
            ...
  DBLWD     DS        D
  ZFIELD    DS        ZL5      5 ZONED DIGITS
  PKFIELD   DS        PL3      3 BYTES WILL HOLD ZFIELD'S DATA
  EDWD      DC        X'402020202120'  6 BYTE EDIT WORD
  CHOUT     DS        CL6
  ZOUT      DS        ZL5
```