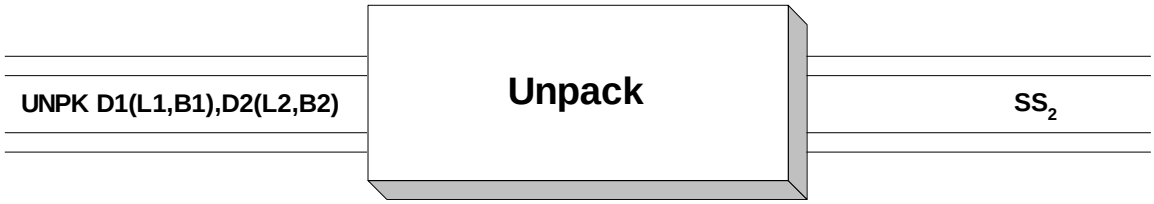
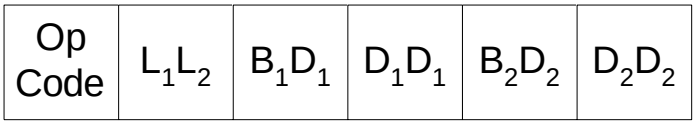




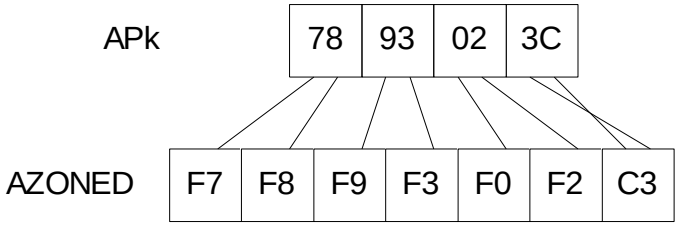
UNPK is a SS_2 instruction which is designed to convert data from a packed format to a zoned decimal format. The operation proceeds by transferring the contents of operand 2 to operand 1. Bytes in Operand 1 and 2 are referenced from right to left within the fields. The rightmost byte of operand 2 is referenced first, and the zone and numeric parts of the byte are reversed and placed in the rightmost byte of operand 1. The next byte (moving right to left) in Operand 1 is processed by taking it numeric part, prefixing it with bits 1111, and storing it in the next byte (moving right to left) of Operand 1. Similarly the zone portion of the byte is prefixed with bits 1111 and stored in the "next" byte of Operand 1. This process of splitting the zone and numeric portions of a byte and prefixing them with bits 1111 continues with all subsequent bytes in Operand 2. The process is terminated (with possible truncation) if Operand 1 becomes filled. Keep in mind that the lengths of both operands are provided in the instruction since it has an SS_2 format. If all the bytes in Operand 2 are processed and there are bytes remaining in Operand 1, the bytes are filled with X'F0's. If Operand 1 is filled before processing all the bytes in Operand 2, high-order truncation of the value in Operand 1 will occur.

The example below illustrates this idea. The first field, APK, contains packed decimal data, while the second field, AZONED, shows the results of executing the code below.

```
        UNPK    AZONED,APK
```

We assume the following definitions,

```
APK         DC     PL4'7893023'
AZONED      DS     ZL7
```

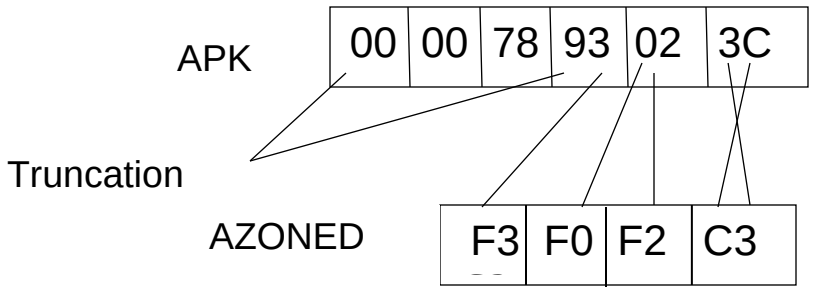


Since **UNPK** is SS_2, each operand contributes a 4-bit length which is stored in the second byte of the object code as pictured at the top of this page. The maximum length in the object code is B'1111' = 15. Since the assembler always decrements lengths by 1, packed fields are limited to a maximum of 16 bytes. It is also important to note that the lengths of both fields are used to execute this instruction. For instance, if AZONED were defined as ZL3, the high-order digits in the packed field (those on the left) are truncated in the result. Consider execution of the the following instruction.

```
UNPK    AZONED,APK
```

We assume the following definitions,

```
APK       DC    PL6'7893023'
AZONED    DS    ZL4
```



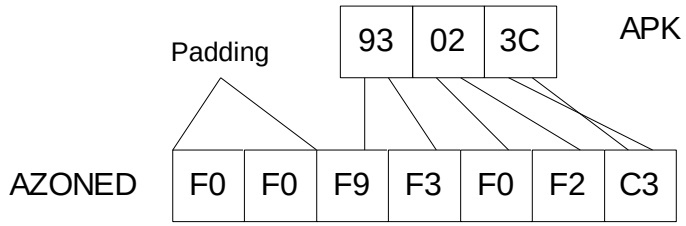
The rightmost bytes of APK are processed and placed in AZONED. When this field is full, the remaining bytes in APK are truncated.

On the other hand, if there are too few bytes in operand 2, zeros will be padded on the left in operand 1 as illustrated below. Assume the following declarations,

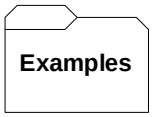
```
APK      DC   P' '
AZONED   DS   ZL6
```

Suppose we execute the following instruction.

```
UNPK    AZONED,APK
```



All the bytes of APK are processed and fill up five of the bytes in AZONED. Since AZONED was defined as a seven byte field, the leftmost 2 bytes are padded with X'F0's.



Some Unrelated UNPK's:

```
A           DC      P'12345'      = X'12345C'
B           DC      PL3'-12345'   = X'12345D'
N           DS      ZL7
P           DS      ZL5
Q           DS      ZL3
R           DS      ZL2

            UNPK    N,A         P = X'F0F0F1F2F3F4C5'  ZERO PADDING
            UNPK    P,A         P = X'F1F2F3F4C5'
            UNPK    Q,A         Q = X'F3F4C5'  LEFT TRUNCATION
            UNPK    R,B         R = X'F4D5'    LEFT TRUNCATION
```

Tips

1. **UNPK** will operate on any kind of data. That does not mean that you will like the results. It simply means that the program will not abend because of the data contents in a field you unpacking.

2. Use **ED** or **EDMK** to convert data to a printable format. **UNPK** leaves packed data in a format that is not acceptable for printing. For example, consider unpacking XPK into XZONE.

```
XPK   DC   P'15'    XPK = X'015C'
XZONE DS   ZL3
```

After executing "UNPK XZONE, XPK", XZONE contains X'F0F1C5'. When printed the field would appear as 01E, since X'C5' is equivalent to a character E.