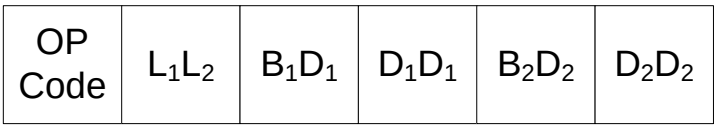


ZAP is an SS_2 instruction which is used to copy packed decimal fields between storage locations in memory. The copying preserves the arithmetic quality of the sending field by first zeroing the target field, and then adding the sending field to it. The initial contents of Operand 1 don't affect the operation. Operand 2 must contain a packed field before the operation occurs, otherwise an S0C7 abend will occur. The operands are limited to a maximum length of 16 bytes and may have different sizes since this is a SS_2 instruction.

A decimal overflow (condition code = 3) can occur if a significant digit is lost from the source. A decimal overflow may or may not cause a program interruption (abend). This depends on the setting of a bit in the PSW (See SPM). Otherwise, the condition code is set to indicate whether the result was zero (condition code = 0), negative (condition code = 1), or positive (condition code = 2). You can test these conditions with BZ or BNZ, BM or BNM, and BP or BNP.

Consider the following AP example.

```
          ZAP  APK,BPK
          BP   APOSITIV     BRANCH IF APK IS POSITIVE
         …
APOSITIV EQU  *
```

Assume the variables are initially in the following states,

```
APK       DC  PL4'34'   = X'0000034C'
BPK       DC  PL3'22'   = X'00022C'
```

After the ZAP instruction has executed, the variables have the following values.

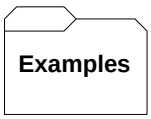
APK = X'0000022C'
BPK = X'00022C'

The contents of APK were zeroed in a packed decimal format, and the contents of BPK were added and the result stored in APK. The contents of BPK were unaffected by the ZAP operation. The branch would occur and execution would resume at "APOSITIV" since the condition code was set to positive.

On the other hand, consider the following example,

```
         ZAP  APK,APK
         BP   THERE
         …
APK      DC PL2'-999'  = X'999D'
```

After the ZAP instruction has executed, the condition code is set to 1, indicating low or negative. APK is unchanged. The branch to THERE is not taken.



Some unrelated ZAP's:

```
A          DC  P'12345'  = X'12345C'   3 BYTES
B          DC  P'-32'    = X'032D'     2 BYTES
C          DC  Z'11'     = X'F1C1'     2 BYTES

     Results:
          ZAP  A,=P'20'       A = X'00020C'    C.C. = HIGH/POSITIVE
          ZAP  B,=P'-20'      B = X'020D'      C.C. = LOW/MINUS
          ZAP  B,=P'12345'    B = X'345C'      C.C. = OVERFLOW
          ZAP  A,A            A = X'12345C'    C.C. = HIGH/POSITIVE
          ZAP  B,B            B = X'032D'      C.C. = LOW/MINUS
          ZAP  B,=P'0'        B = X'000C'      C.C. = ZERO
          ZAP  A,C            DATA EXCEPTION - C NOT PACKED
          ZAP  C,A            C = X'345C'      C.C. = OVERFLOW
                                                Initial contents of C
                                                doesn't matter
```

Tips

1. Use ZAP to copy packed data. Avoid using MVC to transfer packed decimal fields between storage locations.
2. ZAP a field to itself if you want to set the condition code based on the contents of the field.