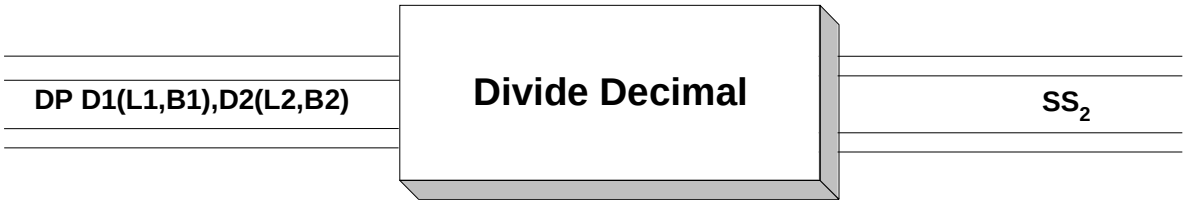


Op Code	L_1L_2	B_1D_1	D_1D_1	B_2D_2	D_2D_2

DP is a SS_2 instruction which is used to divide two packed decimal fields and produce a quotient and a remainder. Since packed decimal fields are integers, the division that occurs is an integer division. If you are interested in producing a quotient with decimals, you may need to use **SRP** to shift the quotient before the division, and **ED** to provide a decimal point in the output. (See the topic on "**Decimal Precision**".)

Operand 1 is a field containing a packed decimal number which is the dividend. Both the quotient and the remainder develop in this field. Operand 2 is a packed decimal field containing the divisor. The maximum length of the dividend is 16 bytes, and the maximum length of the divisor is 8 bytes. Keep in mind the following rules when thinking about the sizes of the generated quotient and remainder,

1) The remainder size is equal to the size of the divisor.

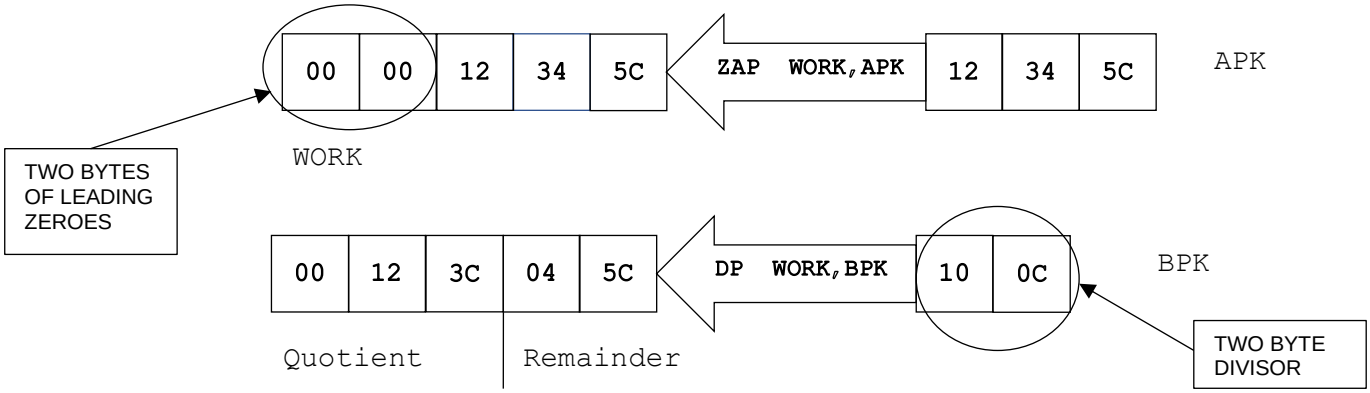
2) The quotient occupies the portion of operand 1 that is not occupied by the remainder.

The rules make practical sense when you think about the division process. Suppose you divide an m- byte divisor into an n-byte dividend. Without knowing the specific values, we can see that in an integer division, the remainder might be as large as m bytes, and the quotient might be as large as n-bytes. This observation leads to the following rule of thumb for creating work fields in which to perform the calculation,

If the dividend is an n-byte field, and the divisor is an m-byte field, make the work field for the computation at least n+m bytes.

Following this rule of thumb will ensure that the dividend contains enough bytes of leading zeroes before the computation. Specifically, the number of bytes of leading zeroes in the dividend must be at least large as the number of bytes in the

divisor. Here is a sample computation where we want to divide APK by BPK. Since APK is 3 bytes and BPK is 2 bytes, we create a work area called "WORK" which is 5 bytes. This will provide the required number of bytes of leading 0's for our computation.



Some unrelated DP's:

```
LPK      DC    PL4'100'     =X'0000100C'
MPK      DC    PL3'6'       =X'00006C'
NPK      DC    PL2'6        =X'006C'
OZONE    DC    Z'11'        =X'F1C1'

WORK1    DS   0CL7
QUOT1    DS    PL4
REM1     DS    PL3

WORK2    DS   0CL7
QUOT2    DS    PL5
REM2     DS    PL2


ZAP      WORK1,LPK           WORK1 = X'0000000000100C'
DP       WORK1,MPK           WORK1 = X'0000016C00004C' QUOT1 = X'0000016C'
                                                       REM1  = X'00004C'
ZAP      WORK2,LPK           WORK2 = X'0000000000100C'
DP       WORK2,NPK           WORK2 = X'000000016C004C' QUOT2 = X'000000016C'
                                                       REM2  = X'004C'


ZAP      WORK1,LPK           WORK1 = X'0000000000100C'
DP       WORK1,OZONE         ABEND, OZONE NOT PACKED

ZAP      WORK1,=P'12345678'  WORK1 = X'0000012345678C'
DP       WORK1,MPK           ABEND, NOT ENOUGH BYTES OF LEADING 0'S
```

Tips

1. Don't divide by zero. An attempt to divide by zero causes an "0CB" abend. Protect your divisions by testing the divisor before you divide.

```
ZAP      BPK,BPK          IS DIVISOR 0?
         BZ    ZERODIV    BRANCH IF ZERO
         ZAP   WORK,APK   OTHERWISE...
         DP    WORK,BPK   ... WE CAN DIVIDE
         ...
ZERODIV  DS   0H
(CODE TO HANDLE A ZERO DIVISOR)
```