Chapter 13
More Packed Decimal Arithmetic

Objectives

Upon completion of this chapter you will be able to:
- Use the MP instruction to multiply one number by another,
- Use the DP instruction to divide one number by another,
- Show how to avoid division by zero,
- Show how to track the location of the implied decimal point when using MP and DP,
- Use the SRP instruction to multiply or divide a number by a power of ten, and
- Use the SRP instruction to round a number, especially a quotient following division.

Introduction

In chapter seven we introduced the packed decimal format as well as selected instructions including AP, SP, CP, ZAP, PACK, and UNPK. In this chapter we look at the multiply (MP), divide (DP), and shift-and-round (SRP) instructions.

In this chapter we will continue our work with the inventory file for Cogsworth Industries. Recall the record layout for that file, COGS.DAT, is as follows:

Field Nbr	Field Name	Description	Begins	Ends	Len	Format
1	DESC	Product desc	1	10	10	CH
2	CALIF	Calif sales	11	13	3	ZD
3	ILL	Illinois sales	14	16	3	ZD
4	UTAH	Utah sales	17	19	3	ZD
5	WISC	Wisconsin sales	20	22	3	ZD
6	BEGIN	Beginning inv.	23	25	3	ZD
7	PURCH	Purchases	26	28	3	ZD
8	QOH	Quantity on hand	29	31	3	ZD
9	COST	Cost (each)	32	35	4	99V99
10	SELL	Sell for (each)	36	39	4	99V99
11	CRLF	PC/370 Only	40	41	2	CR/LF

...and the data is as follows:

```
         1         2         3
12345678901234567890123456789012345678901
GIZMOS    020030020020017099023122529990
WIDGETS   015010010002022034019001100250
JUNQUE    025015015018030052010015503390
```

We will look at two programs in this chapter. The first reads COGS.DAT and calculates and displays company-wide dollar sales. The output is done through WTOs only: there is no output file. The output appears as follows:

```
A:\MIN>cogs13a
COGS13A ... Begin execution
COGS13A ... Nationwide dollar sales are  2,955.82
COGS13A ... Normal end of program
```

Let's look at the calculation of nationwide dollar sales:

Product	Calif	Ill	Utah	Wisc	Total	Sell for	Product
GIZMOS	20	30	20	20	90	29.99	2699.10
WIDGETS	15	10	10	2	37	.25	9.25
JUNQUE	25	15	15	18	73	3.39	247.47
						TOTAL	2955.82

As you can see, we will need multiplication in order to determine the nationwide dollar sales. We now discuss the multiply packed instruction in general, after which we will return to this problem.

The Multiply Packed Instruction

The general format of the Multiply Packed (`MP`) instruction is: `MP FLDA,FLDB`

This instruction will multiply `FLDA` by `FLDB` storing the product (result) in `FLDA`. Of course, both fields must be valid packed fields. There are also two other very important rules:

- The length of the second operand must be less than or equal to 8, and
- The first operand must have as many bytes of high-order zeroes as there are bytes (all bytes) in the second operand. (Note it follows, therefore, that the second operand must be smaller in length than the first operand.)

This second rule is somewhat confusing and is best explained through some examples. Given the following definitions:

```
FLDA   DC   PL3'5'
FLDB   DC   PL3'20'
FLDC   DC   PL3'1000'
FLDD   DC   PL5'5'
FLDE   DC   PL5'20'
FLDF   DC   PL5'1000'
```

Consider the following multiplication instructions:

1. `MP  FLDA,FLDB`

   Recall that the length of the first operand (`FLDA` here) must be greater than the length of the second operand (`FLDB` here). Since these fields are of equal length (both are `PL3`), this `MP` is invalid. Note that this is in spite of the fact that the product (`5*20=100`) *will* fit in a `PL3` field!

   Note: This instruction *will* assemble correctly. *The error will not be detected until run time!* This can be seen with the following (very short) program:

```
MP       BEGIN
         MP    FLDA,FLDB
         RETURN
FLDA     DC    PL3'5'
FLDB     DC    PL3'20'
FLDC     DC    PL3'1000'
FLDD     DC    PL5'5'
FLDE     DC    PL5'20'
FLDF     DC    PL5'1000'
         END
```

(You should key this program and run it. By changing the `MP` line only, you can test each of the `MP` instructions we will be looking at and verify the conclusions presented in throughout chapter.)

From the following screen images you can see that, as mentioned above, *the error is not detected until run time:*

```
A:\MIN>m370

A:\MIN>a370 mp/lx

        (Copyright message appears here)

PC/370 CROSS ASSEMBLER  OPTIONS=LXACE
STATS SYM=00009  MAXSTD=00003  LIT=00000  MAXLTD=00000  BTMEM=52284
NO ERRORS FOUND

A:\MIN>l370 mp/lx

        (Copyright message appears here)

PC/370 LINKAGE EDITOR  OPTIONS ON = LXEFIP
STATS SYM=00001  MAXSTD=00001  BTMEM=57788
NO ERRORS FOUND

A:\MIN>mp
TRACE EP A=08E4 ID=BUG   370 A=000270 OP=58DD0004
 ***********************************************
 *    PC/370 System Release 4.2    01/07/88    *
 *    Copyright (C) 1988 Donald S. Higgins     *
 *                                             *
 * You are encouraged to copy and share this   *
 * package with other users on the condition   *
 * the package is not distributed in modified  *
 * form, and that no fee is charged.  If you   *
 * find PC/370 useful, send 45 dollars to the  *
 * address below to become registered user and *
 * support continued shareware development.    *
 * Registered users will receive notices of    *
 * future PC/370 releases.                     *
 *                                             *
 *      Don Higgins                            *
 *      6365 - 32 Avenue, North                *
 *      St. Petersburg, Florida 33710          *
 ***********************************************
TYPE H FOR HELP
+
```

(Press Esc to return to the DOS prompt.)

As with our earlier discussion of PC/370's test facility, `ID=BUG` tells us that the program has ended, and `A=000270` tells us where. We subtract `X'200'` from the address giving us `000070`, the address of the next instruction to be executed had the program not ended. We see from the `.PRN` listing that this is the location of the `RETURN` macro, so the instruction which caused the error is the previous instruction: the `MP`.

```
 MP                                                      PAGE    1
PC/370 CROSS ASSEMBLER  OPTIONS=LXACE
   LOC                ADR1   ADR2 LINE LABEL    OP       OPERANDS
000000                              1 *+        BEGIN
000000                              2 MP        CSECT
000000                              3           USING    *,15
000000 47F0F058                0058    4           B        KZHQX001
000004 0B                              5           DC       AL1(11)
000005 D4D7404040404040                6           DC       CL11'MP      '
000010 0000000000000000                7 HZQKX001  DC       18F'0'
000058 90ECD00C                000C    8 KZHQX001  STM      14,12,12(13)
00005C 50D0F014                0014    9           ST       13,HZQKX001+4
000060 18ED                           10           LR       14,13
000062 41D0F010                0010   11           LA       13,HZQKX001
000066 50D0E008                0008   12           ST       13,8(0,14)
00006A                                13           DROP     15
00006A                                14           USING    HZQKX001,13
00006A FC22D06AD06D   007A    007D   15           MP       FLDA,FLDB
000070                                16 *+++++++ RETURN
000070 58DD0004                0004   17           L        13,4(13)
000074 98ECD00C                000C   18           LM       14,12,12(13)
000078 07FE                           19           BR       14
00007A 00005C                         20 FLDA     DC       PL3'5'
00007D 00020C                         21 FLDB     DC       PL3'20'
000080 01000C                         22 FLDC     DC       PL3'1000'
000083 000000005C                     23 FLDD     DC       PL5'5'
000088 000000020C                     24 FLDE     DC       PL5'20'
00008D 000001000C                     25 FLDF     DC       PL5'1000'
000098                                26           END
```

2. `MP  FLDD,FLDC`

`FLDD` is `PL5'5'` and `FLDC` is `PL3'1000'`. The length of the second operand *is* less than the length of the first operand. This is a necessary but not sufficient condition for `MP`. Let's look at the hex representation of the first operand. `FLDD` is `X'000000005C'`. Since `FLDC` is `PL3`, `FLDD` must have at least three bytes of high-order zeroes; that is, six hex digits of zero as in `X'000000'`. We can see that `FLDD` does in fact have four bytes of high-order zeroes. Therefore, this `MP` is valid.

Again, we can check this by using PC/370's test facility to stop the execution of `MP.MLC` immediately after the `MP`, and then look at `FLDD`. We should see `5*1000=5000`. Recall that an upper case `T` is required to use the test facility. The complete test is shown here with user responses in bold:

```
A:\MIN>mp T
TRACE EP A=07AB ID=370   370 A=000200 OP=47F0F058
```

(Copyright message appears here)

```
TYPE H FOR HELP
+a
ADDR STOP ON
A=270
    000270  58DD0004 98ECD00C 07FE0000 5C00020C  ....q.......*...
T(A-ADDR, E-DATA =, OR N-DATA <>)= a
+t
TRACE SET
TRACE EP A=1433 ID=BC    370 A=000200 OP=47F0F058
TRACE EP A=1F9B ID=STM   370 A=000258 OP=90ECD00C
TRACE EP A=17D1 ID=ST    370 A=00025C OP=50D0F014
TRACE EP A=0CAD ID=LR    370 A=000260 OP=18ED
TRACE EP A=1649 ID=LA    370 A=000262 OP=41D0F010
TRACE EP A=17D1 ID=ST    370 A=000266 OP=50D0E008
TRACE EP A=2376 ID=MP    370 A=00026A OP=FC42D073D070
ADDR STOP
    000270  58DD0004 98ECD00C 07FE0000 5C00020C  ....q.......*...
TRACE EP A=162D ID=L     370 A=000270 OP=58DD0004
+d
A=280
    000280  01000C00 0005000C 00000002 0C000001  ................
    000290  000C0000 00000000 00000000 0007C6E8  ..............FY
+
```

(Press Esc to return to the DOS prompt.)

3. `MP  FLDF,FLDA`

   `FLDF` is `PL5'1000'` and `FLDA is PL3'5'`. This is similar to the previous example where we multiply `5` by `1000`, but here we multiply `1000` by `5`. Intuitively, if the previous example is valid, then so should this one be. But this is *not* the case! Again, let's look at the hex representation of the first operand. `FLDF` is `X'000001000C'`. Since `FLDA` is `PL3`, `FLDF` must have at least three bytes of high-order zeroes; that is, six hex digits of zero as in `X'000000'`. We can see that `FLDF` has only five high-order zeroes. Therefore, this `MP` is invalid.

   Verify this using the test facility.

You Try It...

1. Show why `MP FLDB,FLDE` is invalid.
2. Show why `MP FLDE,FLDB` is valid.
3. Show why `MP FLDE,FLDC` is valid.
4. Show why `MP FLDB,=PL1'5'` is valid.
5. Show why `MP FLDB,=PL2'10'` is invalid.
6. Show why `MP FLDE,=PL2'10'` is valid.
7. Show why `MP FLDE,=CL2'10'` is invalid.

Sample Program: Cogsworth's Nationwide Dollar Sales

We return now to our original programming problem: we want to determine nationwide dollar sales. Our output will appear as follows:

```
A:\MIN>cogs13a
COGS13A ... Begin execution
COGS13A ... Nationwide dollar sales are  2,955.82
COGS13A ... Normal end of program
```

...which we will display with a `WTO`, so we define the output as:

```
OMSG     DS    0CL49
         DC    CL39'COGS13A ... Nationwide dollar sales are'
ODOLLARS DC    XL10'4020206B2021204B2020'   BZZ,ZZ9.99
```

Output dollars will be up to `99,999.99`, for a total of 7 digits, so we will use a `PL4` field to accumulate the sales. We therefore define this total as:

```
WDOLLARS DC    PL4'0'             Nationwide dollar sales
```

We will determine total sales within the `PROCESS` section. For each record (which contains units sold by product), each state's sales is added, giving the total units sold for this product. The total units sold is multiplied by the product's selling price giving total dollar sales for this product. This figure is then added to the accumulator. Read the next record and repeat.

```
PROCESS  EQU   *
         ST    R10,SVPROC
         PACK  WCALIF,ICALIF      Each product's sales must
         PACK  WILL,IILL           be packed so they can be
         PACK  WUTAH,IUTAH          added to total for this
         PACK  WWISC,IWISC           product...
         ZAP   WTOTAL,=P'0'       Initialize the total to zero
         AP    WTOTAL,WCALIF       and start adding...
         AP    WTOTAL,WILL
         AP    WTOTAL,WUTAH
         AP    WTOTAL,WWISC
         PACK  WSELL,ISELL        Unit sell price
         ZAP   PK5,WTOTAL         Length of WTOTAL is PL2
         MP    PK5,WSELL           and length of WSELL is PL3
         AP    WDOLLARS,PK5         so need PL5 for product.
         BAL   R10,READ
         L     R10,SVPROC
         BR    R10
```

The complete program, `COGS13A.MLC`, follows.

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  COGS13A.MLC                                 *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  Determine nationwide dollar sales for       *
*                   COGSWORTH INDUSTRIES.                       *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'COGS13A ... Begin execution'
         BAL   R10,SETUP
MAIN     EQU   *
         CLI   EOFSW,C'Y'
         BE    EOJ
         BAL   R10,PROCESS
         B     MAIN
EOJ      EQU   *
         BAL   R10,WRAPUP
         WTO   'COGS13A ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                before any records are processed.              *
*****************************************************************
SETUP    EQU   *
         ST    R10,SVSETUP
         OI    INVENTRY+10,X'08'  PC/370 ONLY - Convert all
*                                 input from ASCII to EBCDIC
         OPEN  INVENTRY
         BAL   R10,READ
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
READ     EQU   *
         ST    R10,SVREAD
         GET   INVENTRY,IREC      Read a single product record
         B     READX
ATEND    EQU   *
         MVI   EOFSW,C'Y'
READX    EQU   *
         L     R10,SVREAD
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS  EQU   *
         ST    R10,SVPROC
         PACK  WCALIF,ICALIF      Each product's sales must
         PACK  WILL,IILL           be packed so they can be
         PACK  WUTAH,IUTAH          added to total for this
         PACK  WWISC,IWISC           product...
         ZAP   WTOTAL,=P'0'       Initialize the total to zero
         AP    WTOTAL,WCALIF       and start adding...
         AP    WTOTAL,WILL
         AP    WTOTAL,WUTAH
         AP    WTOTAL,WWISC
```

(continued)

```
         PACK  WSELL,ISELL        Unit sell price
         ZAP   PK5,WTOTAL         Length of WTOTAL is PL2
         MP    PK5,WSELL           and length of WSELL is PL3
         AP    WDOLLARS,PK5         so need PL5 for product.
         BAL   R10,READ
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         ED    ODOLLARS,WDOLLARS
         WTO   OMSG
         CLOSE INVENTRY
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.DAT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP  DC    F'0'               SETUP
SVPROC   DC    F'0'               PROCESS
SVREAD   DC    F'0'               READ
SVWRAP   DC    F'0'               WRAPUP
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
EOFSW    DC    CL1'N'             End of file? (Y/N)
WCALIF   DC    PL2'0'             Units sold in Calif
WILL     DC    PL2'0'             Units sold in Illinois
WUTAH    DC    PL2'0'             Units sold in Utah
WWISC    DC    PL2'0'             Units sold in Wisconsin
WTOTAL   DC    PL2'0'             Units sold in all states
WSELL    DC    PL3'0'             Sell for (each) 999V99
WDOLLARS DC    PL4'0'             Nationwide dollar sales
PK5      DC    PL5'0'
*****************************************************************
*        Input record definition                                *
*****************************************************************
IREC     DS    0CL41        1-41  Inventory record
IDESC    DS    CL10         1-10  Product description
ICALIF   DS    CL3         11-13  Units sold in Calif
IILL     DS    CL3         14-16  Units sold in Illinois
IUTAH    DS    CL3         17-19  Units sold in Utah
IWISC    DS    CL3         20-22  Units sold in Wisconsin
IBEGIN   DS    CL3         23-25  Beginning inventory
IPURCH   DS    CL3         26-28  Purchases throughout year
IQOH     DS    CL3         29-31  Actual quantity on hand
ICOST    DS    CL4         32-35  Cost (each) 99V99
ISELL    DS    CL4         36-39  Sell for (each) 99V99
ICRLF    DS    CL2         40-41  PC/370 only - CR/LF
```

(continued)

```
*************************************************************
*        Output message definition                          *
*************************************************************
OMSG     DS    0CL49
         DC    CL39'COGS13A ... Nationwide dollar sales are'
ODOLLARS DC    XL10'4020206B2021204B2020'   BZZ,ZZ9.99
         END   BEGIN
```

Rounding with SRP

Consider the following programming problem: A card image file contains a price in positions 16-20 (`999V99`) and a percent off value in positions 21-22 (`99V`). We want to determine the sale price (`BZZ9.99`). We will use the following field definitions:

```
PRICE    DS    CL5
PCTOFF   DS    CL2
SALE     DC    X'402021204B2020'
```

Assume `PRICE` contains `'00425'`, representing $4.25, and `PCTOFF` contains `'15'`, representing 15%. We first determine the discount as `PRICE` x `PCTOFF`: $4.25 x .15 = $.6375. But this discount, .6375, should be rounded to .64. We then subtract the (rounded) discount from the regular price giving the sale price: $4.25 - $.64 = $3.61. We now discuss rounding, after which we will return to this problem.

* * * * * * * * * * * * * * * * * * * *

Rounding in BAL is done through the `SRP` (shift-and-round packed) instruction. This instruction can be used for more than just rounding. How it works is quite interesting. Consider the following:

```
FLDA  DC  PL4'1278'    same as: X'0001278C'
```

The `SRP` instruction causes the data in the first operand to be shifted left or right by the number of digits specified in the second operand. For example,

```
SRP   FLDA,1,0      would give: X'0012780C'
           : :
           : :..... This zero says don't round.
           :        (The rounding option does not
           :        apply to multiplication.)
           :
           :....... This one says to shift one
                    digit to the left.
```

Note that the result, `X'0012780C'` is equal to `PL4'12780'`, or 12,780, which is the same as if we had multiplied `FLDA` by ten! Another example (using the original data):

```
SRP   FLDA,3,0      would give: X'1278000C'
           :
           :....... This three says to shift three
                    digits to the left.
```

Note that the result, `X'1278000C'` is equal to `PL4'1278000'`, or 1,278,000, which is the same as if we had multiplied `FLDA` by one thousand! Of course, $1000 = 10^3$ hence the 3 as the second operand, and in general, `SRP` with a second operand of `n` is equivalent to multiplying the field by 10^n. Another example:

```
SRP   FLDA,62,0     would give: X'0000012C'
           :  :
           :  :.... This zero says don't round.
           :
           :....... This 62, which is 64-2, says to
                    shift two digits to the right.
```

Note that the result, `X'0000012C'` is equal to `PL4'12'`, or 12, which is the same as if we had divided `FLDA` by one hundred! Continuing with the original data:

```
SRP   FLDA,64-2,5   would give: X'0000013C'
           :    :
           :    :.. This five says do round.
           :
           :....... This is often used in place
                    of 62 in the above example.
```

Note that the result, `X'0000013C'` is equal to `PL4'13'`, or 13, which is the same as if we had divided `FLDA` by ten (giving 127), added five (giving 132), and divided by ten more (giving 13).

In general, `SRP` with a second operand of the form `64-n` is equivalent to multiplying the field by 10^{-n} ,or dividing the field by 10^n. If the third operand is zero, the result is not rounded. If the third operand is five, the result is rounded.

* *

We return to our programming problem. Given:

```
PRICE    DS    CL5         contains 00425 representing $4.25
PCTOFF   DS    CL2         contains 15 representing 15%
SALE     DC    X'402021204B2020'
```

Both fields must first be packed. We can pack `PCTOFF` into a two byte field. `PRICE` will fit into a three byte field, but if we are going to multiply it by the percent off (a two byte field), then it should be defined as a five byte field:

```
PACK  PK5,PRICE
PACK  PK2,PCTOFF
MP    PK5,PK2        Multiply PRICE by PCTOFF
SRP   PK5,64-2,5      giving DISCOUNT (rounded)
PACK  PK3,PRICE      PRICE minus
SP    PK3,PK5         DISCOUNT equals
ED    SALE,PK3          SALE PRICE
```

You Try It...

Given:
```
X   DC   PL3'456'
Y   DC   PL4'2345'
```

8. Show that `SRP X,2,0` gives `X'45600C'`
9. Show that `SRP Y,3,0` gives `X'2345000C'`
10. Show `X` after `SRP X,63,0`
11. Show `Y` after `SRP Y,64-2,0`
12. Show `X` after `SRP X,62,0`
13. Show `X` after `SRP X,62,5`
14. Show `Y` after `SRP Y,64-3,0`
15. Show `Y` after `SRP Y,64-3,5`

Sample Program: California's Contribution to Sales

We will make use of the `SRP` instruction in our next program, which is to produce a report showing the percent of sales by product for California; that is, California's contribution to total sales. The report will appear as follows:

```
          1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890
                  COGSWORTH INDUSTRIES
          California's Contribution to Sales

                Nationwide   California   Percent of
 Product          Sales        Sales      National
----------      ----------   ----------   ----------
XXXXXXXXXX         BZZ9         BZZ9         BZZ9%
XXXXXXXXXX         BZZ9         BZZ9         BZZ9%
XXXXXXXXXX         BZZ9         BZZ9         BZZ9%
----------      ----------   ----------   ----------
TOTALS             BZZ9         BZZ9         BZZ9%
```

In order to determine the percent of sales by product for California, we divide California sales by total (nationwide) sales. We now discuss the divide packed instruction in general, after which we will return to this problem.

The DP Instruction

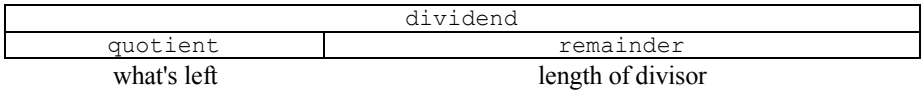
In performing any arithmetic in BAL, we must keep in mind that there is no allowance for a decimal point; that is, all BAL arithmetic is presumed to be integer arithmetic. It is up to you, as the programmer, to keep track of where the decimal point will go, and to put it there when printing the number. This is even more complicated in the case of division. All BAL division is modulus division; that is, the result of a division is always an integer quotient and an integer remainder, never a fractional number.

A review of division terminology is appropriate. If I divide 25 by 4, I get 6 and a remainder of 1. The dividend is 25, the divisor is 4, and the quotient is 6.

```
Dividend -->  25            6    <-- Quotient
Divisor  -->   4      4 | 25
                        24
                         1    <-- Remainder
```

That's what it's like in BAL: As a result of a division, you get a quotient and a remainder. The format of the divide packed instruction is: `DP dividend,divisor`

Both fields must be valid packed numbers or else the program will abend. Following the divide (this part is really weird...) the area containing the `dividend` is split into two parts: the remainder is placed in the right side of `dividend`. Its length is equal to that of the `divisor`. The quotient is then placed on the left side of `dividend` in those bytes not already occupied by the remainder. For example:



For example, given the following field definitions:

```
A        DC    PL5'25'     dividend
B        DC    PL2'4'      divisor
```

...then `DP A,B` results in the following:

A, before DP	00	00	00	02	5C
A, after DP	00	00	6C	00	1C

The quotient, then, is at `A(3)` and the remainder is at `A+3(2)`. Note that `A`, in its entirety, is no longer a valid packed number.

You Try It...

16. Given `A DC PL3'47'` and `B DC PL1'9'`, show that `DP A,B` results in `A = '005C2C'`. `ZAP` the quotient into `Q` and `ZAP` the remainder into `R`.
17. Given `C DC PL5'1276'` and `D DC PL2'100'`, show `C` and `D` after `DP C,D`. `ZAP` the quotient into `Q` and `ZAP` the remainder into `R`.
18. Given `E DC PL4'10'` and `F DC PL2'25'`, show `E` and `F` after `DP E,F`. `ZAP` the quotient into `Q` and `ZAP` the remainder into `R`.

The results of the previous exercise can be demonstrated through a short program and PC/370's test facility. Consider the following program, `DP.MLC`:

```
DP       BEGIN
         DP    A,B
         DP    C,D
         DP    E,F
         RETURN
A        DC    PL3'47'
B        DC    PL1'9'
C        DC    PL5'1276'
D        DC    PL2'100'
E        DC    PL4'10'
F        DC    PL2'25'
         END
```

The resulting `.PRN` file is as follows:

```
 DP                                                    PAGE    1
PC/370 CROSS ASSEMBLER  OPTIONS=LXACE
   LOC                ADR1   ADR2 LINE LABEL    OP       OPERANDS
000000                                1 *+       BEGIN
000000                                2 DP       CSECT
000000                                3          USING    *,15
000000 47F0F058              0058     4          B        KZHQX001
000004 0B                             5          DC       AL1(11)
000005 C4D7404040404040               6          DC       CL11'DP      '
000010 0000000000000000               7 HZQKX001 DC       18F'0'
000058 90ECD00C              000C     8 KZHQX001 STM      14,12,12(13)
00005C 50D0F014              0014     9          ST       13,HZQKX001+4
000060 18ED                          10          LR       14,13
000062 41D0F010              0010    11          LA       13,HZQKX001
000066 50D0E008              0008    12          ST       13,8(0,14)
00006A                               13          DROP     15
00006A                               14          USING    HZQKX001,13
00006A FD20D076D079   0086   0089    15          DP       A,B
000070 FD41D07AD07F   008A   008F    16          DP       C,D
000076 FD31D081D085   0091   0095    17          DP       E,F
00007C                               18 *+++++++ RETURN
00007C 58DD0004              0004    19          L        13,4(13)
000080 98ECD00C              000C    20          LM       14,12,12(13)
000084 07FE                          21          BR       14
000086 00047C                        22 A        DC       PL3'47'
000089 9C                            23 B        DC       PL1'9'
00008A 000001276C                    24 C        DC       PL5'1276'
00008F 100C                          25 D        DC       PL2'100'
000091 0000010C                      26 E        DC       PL4'10'
000095 025C                          27 F        DC       PL2'25'
000098                               28          END
```

We will use the text facility to stop the program before and after the `DP`s and view the contents of `A`, `B`, `C`, `D`, `E`, and `F`. The results are as follows:

```
A:\MIN>dp T
TRACE EP A=07AB ID=370   370 A=000200 OP=47F0F058

        (Copyright message appears here)

TYPE H FOR HELP
+a
ADDR STOP ON
A=26a
    00026A  FD20D076 D079FD41 D07AD07F FD31D081  .....`...:."...a
T(A-ADDR, E-DATA =, OR N-DATA <>)= a
+t
TRACE SET
TRACE EP A=1433 ID=BC    370 A=000200 OP=47F0F058
TRACE EP A=1F9B ID=STM   370 A=000258 OP=90ECD00C
TRACE EP A=17D1 ID=ST    370 A=00025C OP=50D0F014
TRACE EP A=0CAD ID=LR    370 A=000260 OP=18ED
TRACE EP A=1649 ID=LA    370 A=000262 OP=41D0F010
TRACE EP A=17D1 ID=ST    370 A=000266 OP=50D0E008
ADDR STOP
    00026A  FD20D076 D079FD41 D07AD07F FD31D081  .....`...:."...a
TRACE EP A=2110 ID=DP    370 A=00026A OP=FD20D076D079
+d
A=286       |  A |B|     C     | D |   E    | F
    000286  00047C9C 00000127 6C100C00 00010C02  ..@.....%.......
    000296  5C000000 00000007 C6E80000 00000000  *.......FY......
+a
ADDR STOP OFF
+a
ADDR STOP ON
A=27c
    00027C  58DD0004 98ECD00C 07FE0004 7C9C0000  ....q.......@...
T(A-ADDR, E-DATA =, OR N-DATA <>)= a
+t
TRACE SET
TRACE EP A=2110 ID=DP    370 A=000270 OP=FD41D07AD07F
TRACE EP A=2110 ID=DP    370 A=000276 OP=FD31D081D085
ADDR STOP
    00027C  58DD0004 98ECD00C 07FE005C 2C9C0001  ....q......*....
TRACE EP A=162D ID=L     370 A=00027C OP=58DD0004
+d
A=286       |  A |B|     C     | D |   E    | F
    000286  005C2C9C 00012C07 6C100C00 0C010C02  .*......%.......
    000296  5C000000 00000007 C6E80000 00000000  *.......FY......
+

(Press Esc to return to the DOS prompt.)
```

* * * * * * * * * * * * * * * * * * * *

The DP instruction has rules similar to those of the MP:

- The length of the second operand must be less than or equal to 8, and
- The length of the second operand must be less than the length of the first operand.

Furthermore, according to IBM's Principles of Operations:

> "A decimal-divide exception occurs if the dividend does not have at least one leading zero...A quotient larger than the number of digits allowed is recognized as a decimal-divide exception. The operation is suppressed, and a program interruption occurs."

For example, the following program will abend:

```
DP2      BEGIN
         DP    X,Y
         RETURN
X        DC    PL5'5006'
Y        DC    PL3'5'
         END
```

```
Dividend -->  5006          1001  <-- Quotient
Divisor  -->     5       5 | 5006
                             5005
                                1  <-- Remainder
```

X, before DP	00	00	05	00	6C
X, after DP	00	1C	00	00	1C

|__ *not enough room for quotient (1001)*

Admittedly, keeping these rules straight, and remembering where is the quotient and where is the remainder can be confusing. So I recommend the following "fool proof" method for doing division....

I always define the following fields for division:

```
DIVIDEND DS    0PL16
QUOTIENT DS    PL8
REMAINDR DS    PL8
DIVISOR  DS    PL8
```

Prior to dividing, I will move the dividend to DIVIDEND and the divisor to DIVISOR. I then check to make sure the DIVISOR is not zero, as division by zero is prohibited. I then perform the DP, which leaves the quotient in QUOTIENT and the remainder in REMAINDR. I can shift and round the QUOTIENT if appropriate, then ZAP it into the target field. Consider the following example. Given

```
COUNT    DS    PL3
TOTAL    DS    PL3
PERCENT  DS    CL4    BZZ9
```

Assume COUNT = 200 and TOTAL = 300. Then COUNT divided by TOTAL is .66666, or 67% after rounding. I would code this as follows:

```
ZAP   DIVIDEND,COUNT
ZAP   DIVISOR,TOTAL
CP    DIVISOR,=P'0'
BE    error
SRP   DIVIDEND,3,0          Dividend becomes 200000
DP    DIVIDEND,DIVISOR      Quotient becomes 666
SRP   QUOTIENT,64-1,5       Quotient becomes 67
MVC   PERCENT,=X'40202120'  Three digits only so use
ED    PERCENT,QUOTIENT+6    last two bytes of quotient
```

Comments

1. The condition code is set as a result of ZAP. The condition code will indicate if the result is zero, less than zero, greater than zero, or overflow. Consequently, it is not necessary to do the CP after the ZAP. Instead, I could code the following:

```
ZAP   DIVISOR,TOTAL
BZ    error
```

2. I don't like explicit displacement, so I would prefer to use a work field for the ED:

```
MVC   PERCENT,=X'40202120'  Three digits only so use
ZAP   PK2,QUOTIENT          last two bytes of quotient
ED    PERCENT,PK2
```

Of course, it would be nice to have an easy way to include these division fields in a program without keying them or copying from another program. Well, there is! We can make use of PC/370's COPY feature. We simply create a file containing the instructions we wish to copy. This file should have a .CPY extension. Here is my DIVISION.CPY:

```
*************************************************************
*        This is DIVISION.CPY -  BQ's Easy Divides          *
*        Usage: COPY  DIVISION  (with COPY in column 10)    *
*************************************************************
DIVIDEND DS    0PL16
QUOTIENT DS    PL8
REMAINDR DS    PL8
DIVISOR  DS    PL8
```

Then in my program, I type COPY DIVISION (with COPY in column 10). At assembly time, the file with that name (and .CPY extension) will be merged with the .MLC file which is input to the M370 program, creating the .ALC file which is input to the A370 program.

Note: COPY members cannot contain macros or other copy members: the M370 program does not check for "nested" COPY statements, or macros within COPY statements.

COPY members are useful not only for common field definitions such as this, but for record layouts as well! For example, I can create COGS.CPY as follows:

```
*****************************************************************
*        This is COGS.CPY - Cogsworth's Inventory Data          *
*        Usage: COPY  COGS      (with COPY in column 10)        *
*****************************************************************
IREC     DS    0CL41         1-41  Inventory record
IDESC    DS    CL10          1-10  Product description
ICALIF   DS    CL3          11-13  Units sold in Calif
IILL     DS    CL3          14-16  Units sold in Illinois
IUTAH    DS    CL3          17-19  Units sold in Utah
IWISC    DS    CL3          20-22  Units sold in Wisconsin
IBEGIN   DS    CL3          23-25  Beginning inventory
IPURCH   DS    CL3          26-28  Purchases throughout year
IQOH     DS    CL3          29-31  Actual quantity on hand
ICOST    DS    CL4          32-35  Cost (each) 99V99
ISELL    DS    CL4          36-39  Sell for (each) 99V99
ICRLF    DS    CL2          40-41  PC/370 only - CR/LF

          * * * * * * * * * * * * * * * * * * * * *
```

We return now to our second programming problem: to determine the percent of sales by product for California:

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890
                 COGSWORTH INDUSTRIES
          California's Contribution to Sales

                Nationwide   California   Percent of
 Product           Sales        Sales      National
----------      ----------   ----------   ----------
XXXXXXXXXX         BZZ9         BZZ9         BZZ9%
XXXXXXXXXX         BZZ9         BZZ9         BZZ9%
XXXXXXXXXX         BZZ9         BZZ9         BZZ9%
----------      ----------   ----------   ----------
TOTALS             BZZ9         BZZ9         BZZ9%
```

Note that the second heading line contains an apostrophe. To include an apostrophe in a literal, code a double apostrophe (not a quote). For example:

```
HD2      DS    0CL62
         DC    CL60'       California''s Contribution to Sales'
         DC    XL2'0D25'
```

In order to determine the percent of sales by product for California, we divide California sales by total (nationwide) sales. After finding total sales for this product (using the same method shown in the previous program), we proceed as follows:

```
         ZAP   DIVIDEND,WCALIF
         ZAP   DIVISOR,WTOTAL
         SRP   DIVIDEND,3,0
         DP    DIVIDEND,DIVISOR
         SRP   QUOTIENT,64-1,5
         ZAP   PK2,QUOTIENT
         MVC   OPCT,=X'40202120'
         ED    OPCT,PK2
         MVI   OPCT+L'OPCT,PERCENT
```

(For the sake of brevity I have omitted the code to check for division by zero.)

Note the use of the `MVI` instruction and the use of the length operator as a displacement to move a percent sign to immediately after the percent value.

Similar logic is needed at end of job* (in `WRAPUP`) to compute percent of total. The final output is:

```
                COGSWORTH INDUSTRIES
          California's Contribution to Sales

                Nationwide   California   Percent of
 Product           Sales        Sales      National
----------      ----------   ----------   ----------
GIZMOS               90           20          22%
WIDGETS              37           15          41%
JUNQUE               73           25          34%
----------      ----------   ----------   ----------
TOTALS              200           60          30%
```

The complete program, `COGS13B.MLC`, follows.

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  COGS13B.MLC                                 *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  Produce report for COGSWORTH INDUSTRIES     *
*                   California's contribution to sales.         *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'COGS13B ... Begin execution'
         BAL   R10,SETUP
MAIN     EQU   *
         CLI   EOFSW,C'Y'
         BE    EOJ
         BAL   R10,PROCESS
         B     MAIN
EOJ      EQU   *
         BAL   R10,WRAPUP
         WTO   'COGS13B ... Normal end of program'
         RETURN
```

(continued)

* *Remember, it is never appropriate to use the individual percents (22, 41, and 34) to determine a total percent. Many beginning programmers will attempt to arrive at the final figure by averaging the individual percents. But (22+41+34)/3 = 32%, not 30%. Percent of total is always a separate calculation based on the totals of the original data, not the individual percents.*

```
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                before any records are processed.              *
*****************************************************************
SETUP    EQU   *
         ST    R10,SVSETUP
         OI    INVENTRY+10,X'08'  PC/370 ONLY - Convert all
*                                 input from ASCII to EBCDIC
         OI    REPORT+10,X'08'    PC/370 ONLY - Convert all
*                                 output from EBCDIC to ASCII
         OPEN  INVENTRY
         OPEN  REPORT
         BAL   R10,HDGS
         BAL   R10,READ
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU   *
         ST    R10,SVHDGS
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         PUT   REPORT,HD5
         PUT   REPORT,HD6
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS  EQU   *
         ST    R10,SVPROC
         BAL   R10,FORMAT
         BAL   R10,WRITE
         BAL   R10,READ
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        FORMAT - Format a single detail line.                  *
*****************************************************************
FORMAT   EQU   *
         ST    R10,SVFORM
         MVC   OREC,BLANKS
         MVC   ODESC,IDESC
         PACK  WCALIF,ICALIF      Each product's sales must
         PACK  WILL,IILL           be packed so they can be
         PACK  WUTAH,IUTAH          added to total for this
         PACK  WWISC,IWISC           product...
         ZAP   WTOTAL,=P'0'       Initialize the total to zero
         AP    WTOTAL,WCALIF       and start adding...
         AP    WTOTAL,WILL
         AP    WTOTAL,WUTAH
         AP    WTOTAL,WWISC
         AP    TTOTAL,WTOTAL      Grand total nationwide
         AP    TCALIF,WCALIF      Grand total for Calif
         MVC   OTOTAL,=X'40202120'
         ED    OTOTAL,WTOTAL
         MVC   OCALIF,=X'40202120'
         ED    OCALIF,WCALIF
```

(continued)

```
         ZAP   DIVIDEND,WCALIF
         ZAP   DIVISOR,WTOTAL
         SRP   DIVIDEND,3,0
         DP    DIVIDEND,DIVISOR
         SRP   QUOTIENT,64-1,5
         ZAP   PK2,QUOTIENT
         MVC   OPCT,=X'40202120'
         ED    OPCT,PK2
         MVI   OPCT+L'OPCT,PERCENT
         MVC   OCRLF,WCRLF        PC/370 only.
         L     R10,SVFORM
         BR    R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
READ     EQU   *
         ST    R10,SVREAD
         GET   INVENTRY,IREC      Read a single product record
         B     READX
ATEND    EQU   *
         MVI   EOFSW,C'Y'
READX    EQU   *
         L     R10,SVREAD
         BR    R10
*****************************************************************
*        WRITE - Write a single detail line.                    *
*****************************************************************
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   REPORT,OREC        Write report line
         L     R10,SVWRITE
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         PUT   REPORT,HD6
         MVC   OREC,BLANKS
         MVC   ODESC(6),=C'TOTALS'
         MVC   OTOTAL,=X'40202120'
         ED    OTOTAL,TTOTAL
         MVC   OCALIF,=X'40202120'
         ED    OCALIF,TCALIF
         ZAP   DIVIDEND,TCALIF
         ZAP   DIVISOR,TTOTAL
         SRP   DIVIDEND,3,0
         DP    DIVIDEND,DIVISOR
         SRP   QUOTIENT,64-1,5
         ZAP   PK2,QUOTIENT
         MVC   OPCT,=X'40202120'
         ED    OPCT,PK2
         MVI   OPCT+L'OPCT,PERCENT
         MVC   OCRLF,WCRLF        PC/370 only.
         BAL   R10,WRITE
         CLOSE INVENTRY
         CLOSE REPORT
         WTO   'COGS13B ... Sales report on REPORT.TXT'
```

(continued)

```
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.DAT'
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP  DC    F'0'               SETUP
SVHDGS   DC    F'0'               HDGS
SVPROC   DC    F'0'               PROCESS
SVREAD   DC    F'0'               READ
SVFORM   DC    F'0'               FORMAT
SVWRITE  DC    F'0'               WRITE
SVWRAP   DC    F'0'               WRAPUP
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'            PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'             End of file? (Y/N)
BLANKS   DC    CL62' '
WCALIF   DC    PL2'0'             Units sold in Calif
WILL     DC    PL2'0'             Units sold in Illinois
WUTAH    DC    PL2'0'             Units sold in Utah
WWISC    DC    PL2'0'             Units sold in Wisconsin
WTOTAL   DC    PL2'0'             Units sold in all states
TCALIF   DC    PL2'0'             Grand total for Calif
TTOTAL   DC    PL2'0'             Grand total nationwide
PK2      DC    PL2'0'
PERCENT  EQU   C'%'
         COPY  DIVISION
         COPY  COGS
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62               1-62
ODESC    DS    CL10                1-10  Product description
         DS    CL7                11-17
OTOTAL   DS    CL4                18-21  Units sold Nationwide
         DS    CL9                22-30
OCALIF   DS    CL4                31-34  Units sold in Calif
         DS    CL8                35-42
OPCT     DS    CL4                43-46  Percent sales from Calif
         DS    CL14               47-60
OCRLF    DS    CL2                61-62  PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
HD1      DS    0CL62
         DC    CL60'              COGSWORTH INDUSTRIES      '
         DC    XL2'0D25'
```

(continued)

```
HD2      DS    0CL62
         DC    CL60'       California''s Contribution to Sales'
         DC    XL2'0D25'
HD3      DS    0CL62
         DC    CL60' '
         DC    XL2'0D25'
HD4      DS    0CL62
         DC    CL40'              Nationwide   California   '
         DC    CL20'Percent of'
         DC    XL2'0D25'
HD5      DS    0CL62
         DC    CL40' Product         Sales        Sales     '
         DC    CL20' National '
         DC    XL2'0D25'
HD6      DS    0CL62
         DC    CL40'----------    ----------   ----------   '
         DC    CL20'----------'
         DC    XL2'0D25'
         END   BEGIN
```

A Comprehensive Example

The following example is more complex and illustrates the thought process required to do multiplication and division problems as they appear in the real world....

Assume the current social security tax rate (FICA) is 6.29% and applies to the first $57,600 of income only. Total FICA withholdings cannot exceed $3,623.04. Payroll data comes from a card-image file and includes the following fields:

Input Fields

```
IPPGR    DS    CL6    Current pay period Gross (9999V99)
IYTDGR   DS    CL8    Year-to-date Gross (999999V99)
IYTDFICA DS    CL7    Year-to-date FICA W/H (99999V99)
```

Work Fields

```
WFICA%   DC    PL3'0629'    FICA rate (V9999)
WFICA$   DC    PL4'362304'  FICA dollar cap (99999V99)
```

Write the BAL code necessary to produce the following output fields. Define any additional work fields as necessary.

Output Fields

```
OPPFICA  DS    CL5    Current pay period FICA W/H (999V99)
OYTDGR   DS    CL8    Year-to-date Gross (999999V99)
OYTDFICA DS    CL7    Year-to-date FICA (99999V99)
```

Solution

Since each of the input and output fields are zoned decimal (unpacked, as indicated by definition as `CL`), and since each will be the input to, or result of, arithmetic, let's define a corresponding packed field of appropriate size for each:

```
WPPGR    DS    PL4    Current pay period Gross (9999V99)   - Input
WYTDGR   DS    PL5    Year-to-date Gross (999999V99)       - I/O
WYTDFICA DS    PL4    Year-to-date FICA W/H (99999V99)     - I/O
WPPFICA  DS    PL3    Current pay period FICA W/H (999V99) - Output
```

Let's pack each of the input fields:

```
          PACK  WPPGR,IPPGR
          PACK  WYTDGR,IYTDGR
          PACK  WYTDFICA,IYTDFICA
```

OYTDGR is the easiest field to determine: simply add WPPGR to WYTDGR. The output field is zoned decimal, so the sum must be unpacked and the sign removed:

```
          AP    WYTDGR,WPPGR   Add current pay period to year-to-date
          UNPK  OYTDGR,WYTDGR  Move new YTD gross to output
          MVZ   OYTDGR+L'OYTDGR-1(1),=X'F0'   Remove sign
```

The FICA amount for the current pay period will be the lesser of (1) the current pay period gross times the FICA rate, or (2) the difference between the FICA dollar cap and the Year-to-date FICA withholdings.

To calculate the first amount, we must multiply a number with two decimal places (WPPGR) by another number with four decimal places (WFICA%), then round the result back to two decimal places. These fields are defined as PL4 and PL3 (above) so we will use a seven byte work field for the product so as to avoid any size errors with the multiply command:

```
          ZAP   PK7,WPPGR      PK7 is V99
          MP    PK7,WFICA%     Now PK7 is V999999
          SRP   PK7,64-4,5     Now PK7 is back to V99, rounded
          ZAP   WPPFICA,PK7    This is pay period FICA amount
*                              unless it puts us over the cap
```

Let's call the second amount (the difference between the FICA dollar cap and the Year-to-date FICA withholdings) WMAXFICA, defined as:

```
WMAXFICA DS    PL4    Cap for current pay period FICA W/H, V99
```

This field was defined as PL4 so it could hold WFICA$, which is required for the calculation. The FICA cap and YTD amounts are both V99, so we simply subtract:

```
          ZAP   WMAXFICA,WFICA$    Max this period is yearly max ...
          SP    WMAXFICA,WYTDFICA  less what has already been withheld
```

We now determine the lesser of these two amounts, WPPFICA and WMAXFICA, as follows:

```
            CP    WPPFICA,WMAXFICA   Over the cap?
            BL    SKIP               No  - use percent of gross
            ZAP   WPPFICA,WMAXFICA   Yes - use what remains of the cap
SKIP        EQU   *
            UNPK  OPPFICA,WPPFICA    Move pay period FICA to output
            MVZ   OPPFICA+L'OPPFICA-1(1),=X'F0'   Remove sign
```

Finally, we update the year-to-date FICA withholdings:

```
            AP    WYTDFICA,WPPFICA   Add pay period FICA to YTD FICA
            UNPK  OYTDFICA,WYTDFICA  Move new YTD FICA amount to output
            MVZ   OYTDFICA+L'OYTDFICA-1(1),=X'F0'    Remove sign
```

And we are finished.

Exercises

1. True or false. Given `A DC PL5'2000'`, `B DC PL2'15'`, and `C DC PL3'6'` ...

   T F a. `MP A,B` is valid, but `MP B,A` is not.
   T F b. `MP A,C` is valid, but `MP C,A` is not.
   T F c. `MP C,B` and `SRP C,1,0` give the same result in `C`.
   T F d. If `A` represents 20.00 and `B` represents 1.5 then `MP A,B` gives `A` equal `X'000000030C'` representing 30.000.
   T F e. `SRP A,62,0` and `SRP A,64-2,5` give the same result in `A`.
   T F f. `SRP B,63,0` and `SRP B,64-1,5` give the same result in `B`.
   T F g. `DP A,B` gives `A` equal `X'00133C003C'`
   T F h. `DP A,C` gives `A` equal `X'333C00002C'`
   T F i. `DP C,=PL1'2'` gives `C` equal `X'003C0C'`
   T F j. `DP C,=PL2'2'` is invalid.
   T F k. After `DP A,=PL1'5'`, the remainder is `A+4(1)`.
   T F l. After `DP A,=PL2'5'`, the quotient is `A(3)`.
   T F m. After `DP A,=PL3'5'`, the quotient is `A(2)`.

2. You are given the following fields definitions:

```
IREG     DS   CL3    Input Regional total, 999V
INAT     DS   CL4    Input National total, 9999V
OPCT     DS   0CL8   Output Percent region to nation, BZZ9.99%
ONBR     DS   CL7    Output Mask gets moved here - BZZ9.99
OSIGN    DS   CL1    Output Percent sign gets moved here
WREG     DS   PL2    Packed work field, Regional total
WNAT     DS   PL3    Packed work field, National total
DIVIDEND DS   0PL16
QUOTIENT DS   PL8
REMAINDR DS   PL8
DIVISOR  DS   PL8
```

   We will divide the regional total by the national total giving the percent of total in the form `BZZ9.99%`. Using the given fields only, write the instructions to:

   a. Move the input regional total (zoned decimal) to its packed work field,
   b. Move the input national total (zoned decimal) to its packed work field,
   c. Move the packed regional total to the dividend,
   d. Move the packed national total to the divisor,
   e. If the divisor is equal to zero, move zero to the output field and skip the divide,
   f. Otherwise use `SRP` to multiply the dividend by 100000,
   g. Divide the dividend by the divisor,
   h. Round the quotient,
   i. Move the proper edit mask to the output field,
   j. Edit the quotient into the output field, and
   k. Move the percent sign to the output field.
   l. Write a program to do all of the above. Test using PC/370's test facility.

Exercises

3. (Refer to the Small Town Hardware Store database in More Datasets.) Write a program which will display the total cost of inventory on hand for the Small Town Hardware Store. This is defined as the sum of the (cost * quantity on hand) for all tools and wrappers. Your output should be by `WTO` only: there is no output file. Your message should appear as follows:

```
Total cost of inventory on hand is BZZ,ZZ9.99.
```

(Note: it would seem appropriate to include a dollar sign in the output, but PC/370's implementation of the `WTO` command will not allow a dollar sign. You can see this for yourself by examining the `.PRN` file of any program using a `WTO` and checking the expansion of the `WTO` macro.)

4. Modify the cross tab program in Packed Decimal Arithmetic, Exercise 10, to show percents as well as counts. Your output should appear as follows:

```
         1         2         3         4
1234567890123456789012345678901234567890
         STUDENT STATISTICS

Status      Male    Female   Total
--------   ------   ------   -----
Single     BZZ9     BZZ9     BZZ9
           BZZ9%    BZZ9%    BZZ9%

Married    BZZ9     BZZ9     BZZ9
           BZZ9%    BZZ9%    BZZ9%
--------   ------   ------   -----
Total      BZZ9     BZZ9     BZZ9
           BZZ9%    BZZ9%    BZZ9%
```

5. A dean at the Small Town Community College would like to compare the average grade awarded by course. Use the `GRADE` table to produce such a report. An A is worth 4 points, a B is worth 3 points, a C is worth 2 points, a D is worth 1 point, and an F is worth 0 points. The file will need to be sorted by course ID. Your report should appear as follows:

```
         1         2         3         4
1234567890123456789012345678901234567890
      SMALL TOWN COMMUNITY COLLEGE
        Average Grade by Course

       Course ID    Count     GPA
       ---------    -----    -----
         XXXXX      BZZ9    BZ.99
         XXXXX      BZZ9    BZ.99
```

For example, `MA107` was taken 3 times, with grades B, A, and D. The GPA is calculated as follows:

```
B   A   D
3 + 4 + 1 = 8 / 3 = 2.666 → round to → 2.67
```

Exercises

6. Modify exercise 5 to include an average grade for all courses. *Note: this figure is not an average of the averages.*

7. Modify exercise 5 to include the course description as well as the course number. This will require the use of the `COURSE` file and matching logic.

8. (Refer to the Small Town Hardware Store database in More Datasets.) Produce a report showing markup per item. Markup is defined as (sell-cost)/cost. Round where appropriate. Do not show wrappers, which are indicated by a sell price of zero. Your report should appear as follows:

```
         1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
                    SMALL TOWN HARDWARE STORE
                          Markup Report

TID        Description           Cost      Sell     Markup
---   ----------------------    ------    ------   ------
XXX   XXXXXXXXXXXXXXXXXXXXXX   BZZ9.99   BZZ9.99    ZZ9%
XXX   XXXXXXXXXXXXXXXXXXXXXX   BZZ9.99   BZZ9.99    ZZ9%
XXX   XXXXXXXXXXXXXXXXXXXXXX   BZZ9.99   BZZ9.99    ZZ9%
```

9. (Refer to the Small Town Payroll database in More Datasets.)

   a. Use the `EMPL` and `THISPP` tables to determine gross pay per employee and as a whole for this pay period. Hourly employees are paid time-and-a-half for hours over forty. Salaried employees are paid a fixed amount regardless of the number of hours worked. Both files will need to be sorted and matched on employee number. Your output should appear as follows:

```
          1         2         3         4
1234567890123456789012345678901234567890
PAYROLL REPORT FOR WEEK ENDING 01/09/93

ENUM  Type   Hours     Rate       Gross
----  ----  -------   ------   ---------
XXX     X   BZZ9.99  BZZ9.99     BZZ9.99
XXX     X   BZZ9.99  BZZ9.99     BZZ9.99
----  ----  -------   ------   ---------
TOTALS      BZZ9.99           BZZ,ZZ9.99
```

   b. Modify the program from part (a) to append the results for this pay period (`01/09/93`) to the `HISTORY` file. (This will require creating a new file: call it `HISTORY.NEW`.)