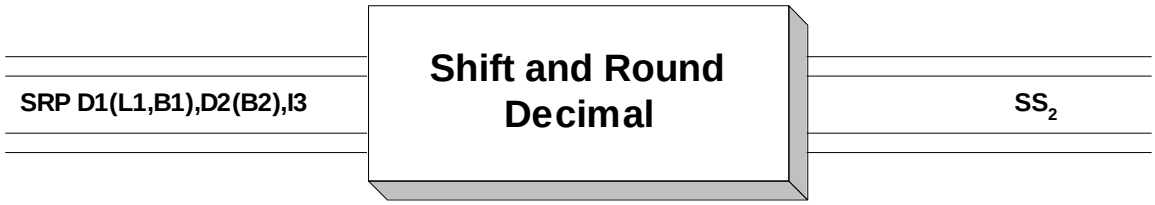
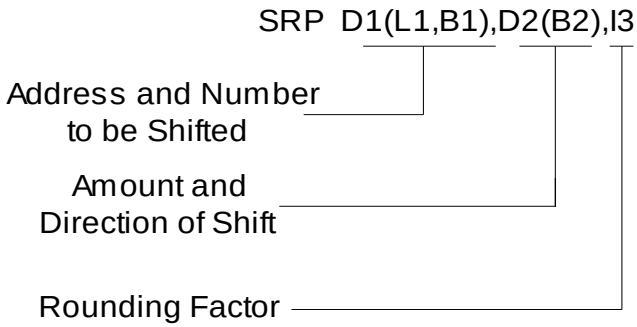


Op Code	L_1I_3	B_1D_1	D_1D_1	B_2D_2	D_2D_2

SRP is a SS_1 instruction designed to shift the digits in a packed decimal field either left or right while leaving the sign of the field fixed in its position. For right shifts, the last digit that is shifted off can be rounded using a rounding factor. As digits are shifted off during a left shift, 0's are shifted in on the right. On the other hand, as digits are shifted off during a right shift, 0's are shifted in on the left.

Consider the explicit format for this instruction.



As we can see from the diagram, operand 1 specifies the field to be shifted and its length, while the third operand is a rounding factor. This factor is added to the last digit which is shifted off during right shifts. Typically, you would code the third operand as 5 in order to round up on a digit of 5 or greater. Coding 0 as the third operand indicates no rounding. A rounding factor must always be coded, even when shifting left.

The second operand above receives special treatment. The notation D2(B2) is converted to an effective address and the last 6 bits of this address is treated as a two's complement number that indicates the number of digits to be shifted and in which direction. A positive number indicates a left shift and a negative number indicates a right shift. Because of the explicit notation, there are two standard ways to code the second operand.

1) Displacement Only -

```
SRP     XPK,3,0
```

In this case the shift factor, 3, is converted to 6-bit binary (B'000011') and the result is interpreted as 6-bit 2's complement. In this format B'000011' is considered to be positive 3 - a left shift by 3.

On the other hand consider the following example,

```
SRP     XPK,62,5
```

The 62 is converted to its binary form (B'111110') and interpreted as a 6-bit 2's complement integer. We see that B'111110' = -2 in base 10. This means the previous shift is a 2-digit right shift.

There is a simple way to deal with the shift factor on right shifts. Simply express the shift factor as 64 - n, where n is the number of digits you wish to shift to the right. For example, the previous example could be expressed as SRP XPK,64-2,5 . The 64 - n expression produces the correct decimal number which will be interpreted as a negative 2's complement integer.

2) Base Register Only -

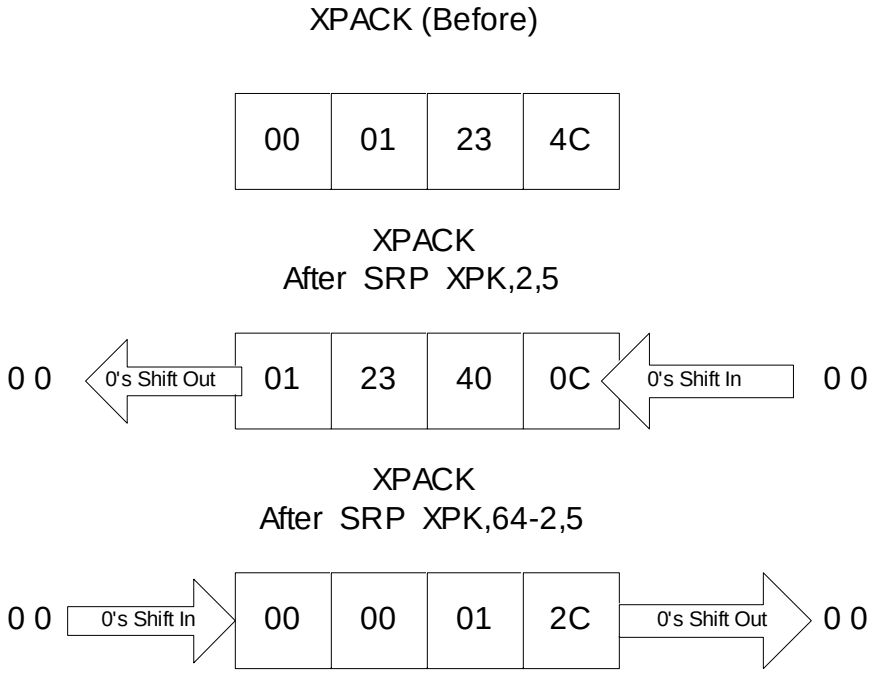
```
SRP     XPK,0(R8),5
```

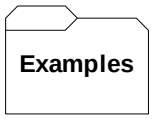
In this format the base register (R8) is initialized with a number representing the shift factor. Because 0 was coded for the displacement, the effective address will be computed to be the contents of the specified register. The number of digits to be shifted and the direction is solely determined by the contents of the register at the time the instruction is executed. This allows us to dynamically change the shift factor. Consider the code below.

```
LA    R6,10         SHIFT FACTOR = 10 DIGITS LEFT
SRP   XPK,0(R6),5   DYNAMIC SHIFT
```

The number 10 (indicating a 10-digit left shift) is loaded into R6. The effective address will be 10.

Here are two example SRP's that indicate what happens at the byte level.





Some unrelated SRP's:

```
APK      DC    PL4'126'          = X'0000126C'
BPK      DC    PL4'1122334'      = X'1122334C'
               RESULTS:
         SRP   APK,3,5       APK = X'0126000C'  LEFT SHIFT 3
         SRP   APK,3,0       APK = X'0126000C'  ROUNDING UNIMPORTANT
         SRP   APK,64-2,5    APK = X'0000001C'  RIGHT SHIFT
         SRP   APK,64-1,5    APK = X'0000013C'  ROUND UP >= 5
         SRP   BPK,1,5       OVERFLOW OCCURS ON LOSING SIG. DIGIT
         SRP   APK,-2,5      ASSEMBLY ERROR - NEGATIVE DISPLACEMENTS
                             ARE NOT ALLOWED
```

Tips

1. Keep in mind that you cannot left shift off a significant digit.

2. Be sure to use the 64-n notation when coding right shifts. It is easy to read, and provides excellent documentation.