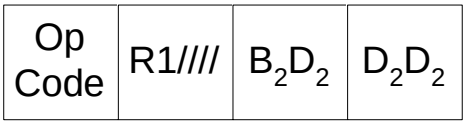
SRA R1,D2(B2)

Shift Right Algebraic

RS



SRA is used to shift the 32 bits in the register specified by Operand 1 to the right. The number of bits that are shifted is indicated by Operand 2. The second operand address is **not** used to address data; instead, the base/displacement address is computed and the rightmost 6 bits are treated as a binary integer which represents the number of bits to be shifted. We will call this value the "shift factor". This leads to two distinct ways of coding the shift factor:

1) **Directly** - The shift factor is coded as a displacement. Consider the example below.

```
        SRA   R9,5
```

In the above shift, the second operand, 5, is treated as a base/displacement address where 5 is the displacement and the base register is omitted. The effective address is 5. (See **Explicit Addressing**.) When represented as an address the rightmost 6 bits still represent the number 5, and so the bits in register 9 are shifted to the right by 5 bits.

2) **Indirectly** - The shift factor is placed in a register and the register is mentioned as the base register in the base/displacement address.

```
        L     R5,FACTOR     PUT SHIFT FACTOR IN REG
        SRA   R9,0(R5)      NOW SHIFT INDIRECTLY
        ...
FACTOR  DC    F'8'          SHIFT FACTOR IS 8 BITS
```

In this case, the effective address is computed by adding the contents of base register 5 (which is 8), with the displacement of 0. The effective address is again 8, and the rightmost 6 bits of this address indicate that the shift factor is 8.

Each method has its uses. The direct method is useful in situations where the number of bits you want to shift is fixed. Coding directly allows you to look at the instruction to determine the shift factor. On the other hand, the indirect method allows the shift factor to be determined while the program is executing. If the shift factor cannot be determined until the program is running, the indirect method must be used.

When shifting algebraically, bits shifted out on the right are lost, while bits equal to the sign bit replace vacated bits on the left. The sign bit in Operand 1 remains fixed, preserving the sign of the integer.

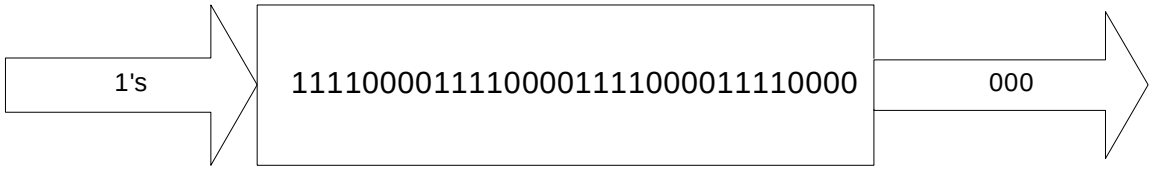
The condition code is set by this instruction in all cases:

Condition Code	Meaning	Test With
0	Result = 0	BZ, BE
1	Result < 0	BL, BM
2	Result > 0	BH, BP

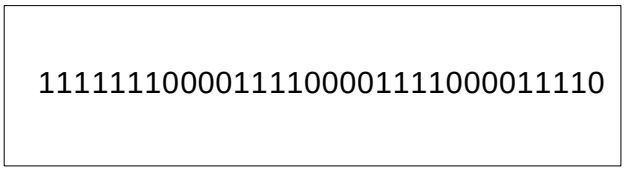
Consider the following instruction and the diagram below.

```
SRA  R8,3
```

This instruction represents a right algebraic shift of register 8 using a shift factor of 3. The shift factor has been coded directly. As a result, 3 bits, 000, are shifted out of the register on the right. Vacated bit positions on the left are replace by 1's (the sign bit is negative). This is illustrated in the diagram below. The condition code is set to 1, indicating that the resulting binary integer is negative.



R8 (Before)



R8 (After)

This instruction has an **RS** format but the 4 low-order bits of the second byte are unused.

Examples

Some Unrelated SRA's

```
R5 = B'11111111111111111111111110000'
R6 = B'00001111000011110000111100001111'

SRA  R5,1     R5 = B'11111111111111111111111111111000'  Cond.Code = 1
SRA  R5,2     R5 = B'11111111111111111111111111111100'  Cond.Code = 1
SRA  R5,3     R5 = B'11111111111111111111111111111110'  Cond.Code = 1
```

```
SRA  R5,4     R5 = B'11111111111111111111111111111111'  Cond.Code = 1
SRA  R5,2     R5 = B'11111111111111111111111111111100'  Cond.Code = 1
SRA  R6,4     R6 = B'00000000111100001111000011110000'  Cond.Code = 2

L    R9,=F'3'
SRA  R6,0(R9) R6 = B'00000000111100001111000011110001'  Cond.Code = 2

L    R3,=F'5'
SRA  R6,0(R3) R6 = B'00000000111100001111000011111000'  Cond.Code = 2
```

Tips

1) Shifting a binary number to the right one digit is equivalent to dividing it by 2 (using integer arithmetic). Using **SRA,** it is a simple matter to divide by powers of 2. For instance, to divide by 2^5, shift right 5 digits.