SH R1,D2(X2,B2)

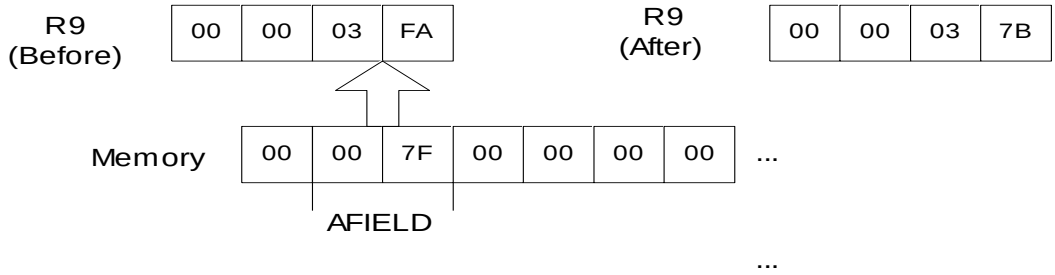
Subtract Halfword

RX

Op Code	R_1X_2	B_2D_2	D_2D_2

The Subtract Halfword instruction performs 2's complement binary subtraction. Operand 1 is a register containing a fullword integer. Operand 2 specifies a halfword in memory. The halfword in memory is sign extended (internally) and subtracted from the fullword in the register. The result remains in the register. The halfword in memory is not changed. Consider the following example,

```
SH    R9,AFIELD
```



The contents of the halfword "AFIELD", x'007F' = 127, are subtracted from register 9 which contains x'000003FA' = 1018. The difference, 891 = x'0000037B', destroys the previous value in R9. The halfword in memory is unchanged by this operation.

Since **SH** is an RX instruction, an index register may be coded as part of operand 2 (see **Explicit Addressing**).

Examples

Some Unrelated Subtract Halfwords

```
R4 =  X'FFFFFFF0'  -16 IN 2'S COMPLEMENT
R5 =  X'00000025'  +37 IN 2'S COMPLEMENT
R6 =  X'00000004'   +4 IN 2'S COMPLEMENT

DOG   DC   H'4',H'9',H'-25',H'3'   CONSECUTIVE HALFWORDS

SH    R4,=H'20'    R4 = X'FFFFFF1C' = -36
SH    R5,=H'20'    R5 = X'00000011' = +17
SH    R6,=H'4'     R6 = X'00000000' =  0
SH    R4,=H'-9'    R4 = X'FFFFFFF9' = -7
SH    R5,DOG       R5 = X'00000021' = +33
```

```
              SH    R5,DOG(R6)   R6 = X'0000003E' = +28 INDEXING
ALLOWED
```

Tips

A common error is to code a **SH** when the second operand is not a halfword. For example:

```
AMOUNT    DC   F'20'         AMOUNT = X'00000014'
          ...
          SH   R5,AMOUNT
```

The assembler will not complain about your code, but the halfword instruction will only access the first two bytes of the AMOUNT field (x'0000').