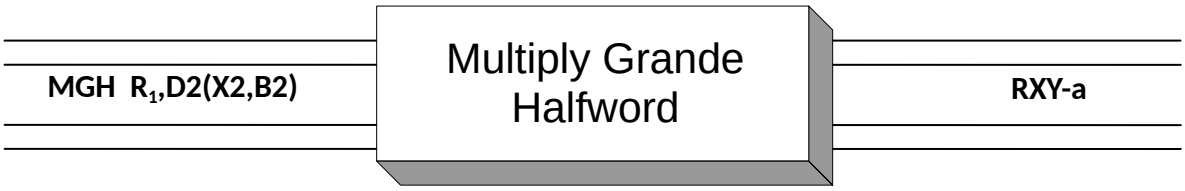


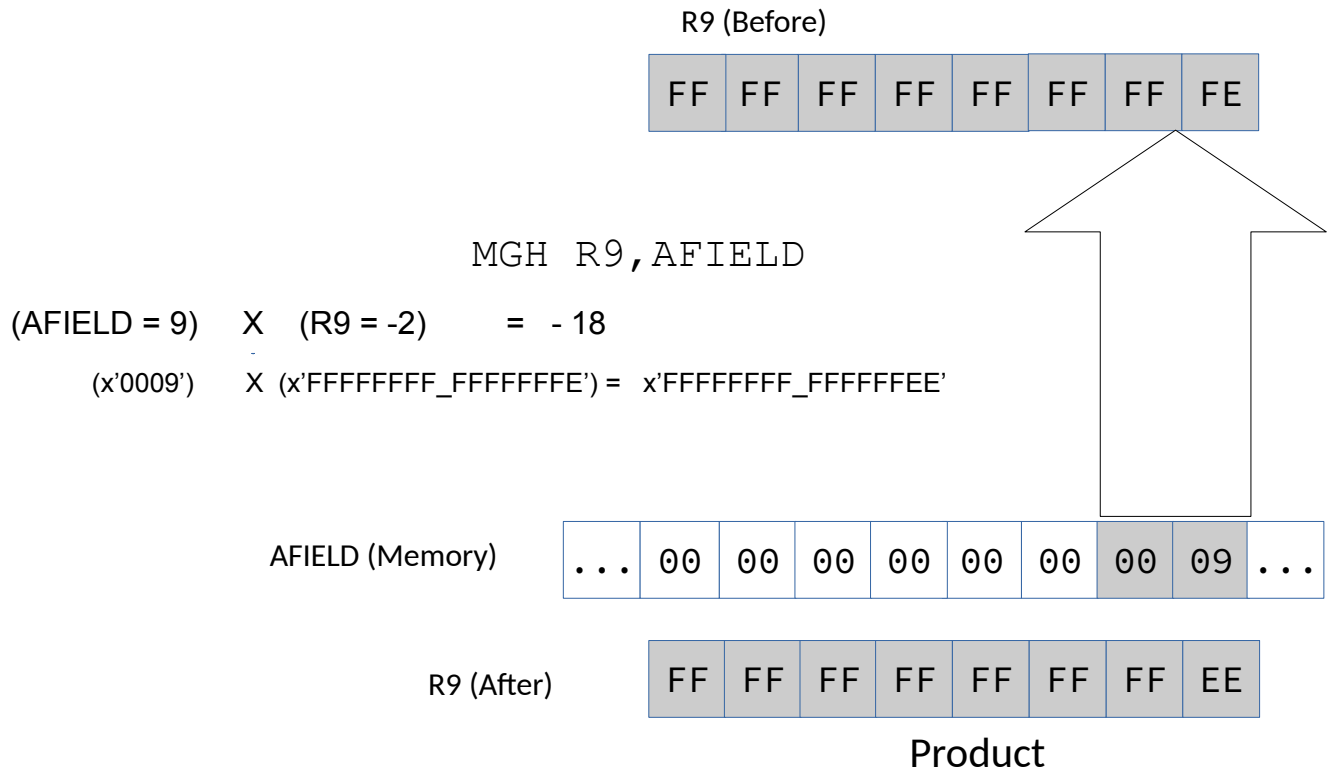
E3	R_1X_2	B_2DL_2	DL_2DL_2	DH_2DH_2	3C

MGH (Multiply Grande Halfword) computes the product of a signed-binary halfword in memory (operand 2) with a signed-binary doubleword in the bits 0-63 of the general register R1. The 64-bit product is a signed binary integer and is placed in bits 0-63 of R1.

The sign of the product is determined by the rules of algebra. A zero product produces a positive sign. Overflows occur are not detected. The condition code is unchanged by this operation.

MGH provides a five hexadecimal digit displacement, $DH_2DH_2DL_2DL_2DL_2$, which is treated as a 20-bit signed binary integer. As a result, operand 2 can be defined approximately one half megabyte before or after the base address in B2.

Consider the following example,



Since **MGH** is an RXY-a instruction, an index register may be coded as part of operand 2.

Examples

Some Unrelated Multiply Grandes Halfwords

```
R4 =  X'00000000_00000005'          +5
R5 =  X'FFFFFFFF_FFFFFFF8'          -8
R6 =  X'7FFFFFFF_FFFFFFFF'          9,223,372,036,854,775,807 =
                                    LARGEST POSITIVE VALUE

         DS 0D
AFIELD   DC  XL2'0002'              +2
BFIELD   DC  XL2'FFFF'              -1
CFIELD   DC  XL2'CDEF'              +52,719
```

After Operation Completes

```
         MGH   R4,AFIELD            R4 = X'00000000_0000000A'    +10

         MGH   R4,BFIELD            R4 = X'FFFFFFFF_FFFFFFFB'    -5

         MGH   R5,CFIELD            R5 = X'FFFFFFFF_FFF99088'    -421,752

         MGH   R6,AFIELD            R6 = X'FFFFFFFF_FFFFFFFE'    -2  OVERFLOW
                                                                     UNDETECTED
```

Tips

1) When working with 64-bit registers, it helps to have a programmer's calculator where you can control the number of bits.