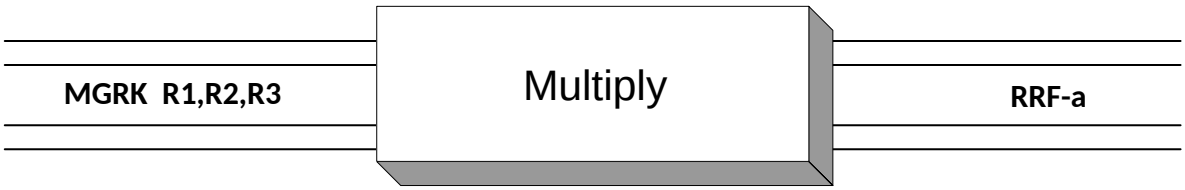

B9	EC	R_3//	R_1R_2

Multiply (**MGRK**) is used to multiply two 64-bit signed, 2's complement, binary fields stored in grande registers, and produces a signed 128-bit product that is stored in an even/odd consecutive grande register pair. R_1 names an even register of an even/odd grande register pair. The contents of the even/odd pair are ignored prior to the multiplication. The 128-bit product develops in R_1 and R_1 + 1. Before the multiplication, R_3 contains the 64-bit multiplicand, and R_2 contains the 64-bit multiplier. These two registers are unchanged by the operation.

The sign of the product is determined by the usual rules of arithmetic. The only exception is that a zero result always has a positive sign.

If the product is in the range -18,446,744,073,709,551,616 to +18,446,744,073,709,551,615 it can be found in R_1 + 1. Converting values outside this range to packed decimal requires special treatment which is covered in a separate article.

Consider the following example, MGRK R4,R8,R9 in which we multiply 5 x 10 = 50.

Before the multiplication:

R4 (Ignored)

11	22	33	44	55	66	77	88

R5 (Ignored)

FF	EE	DD	CC	BB	AA	99	88

R8 (Multiplier)

00	00	00	00	00	00	00	05

R9 (Multiplicand)

00	00	00	00	00	00	00	0A

MGRK R4,R8,R9

After the multiplication:

R4 (Product)

00	00	00	00	00	00	00	00

R5 (Product)

00	00	00	00	00	00	00	32

R8 (Multiplier)

00	00	00	00	00	00	00	05

R9 (Multiplicand)

00	00	00	00	00	00	00	0A

The condition code is unchanged by this operation

Examples

Some Unrelated MGRKs **UNDERSCORE _ FOR READABILITY**

```
R4 =  X'12121212_12121212'          IGNORED
R5 =  X'00000000_00000008'          IGNORED
R6 =  X'FFFFFFFF_FFFFFFFF'          -1
R7 =  X'00000000_00000010'          16
```

MGRK R4,R6,R7

After Multiply:

```
R4 =  X'FFFFFFFF_FFFFFFFF'      (R4 & R5 = -16) (R4 = -1) (R5 = -16))
R5 =  X'FFFFFFFF_FFFFFFF0'
R6 =  X'FFFFFFFF_FFFFFFFF'      R6 = -1
R7 =  X'00000000_00000010'      R7 = 16
```

Before Multiply:

```
R5 =  X'00000000_00000008'          IGNORED
R6 =  X'FFFFFFFF_FFFFFFFF'          -1
R7 =  X'00000000_00000010'          16
R8 =  X'00000000_000003FC'        1020
```

MGRK R6,R5,R8

After Multiply:

```
R5 =  X'00000000_00000008'         8
R6 =  X'00000000_OOOOOOOO'      (R6 & R7 = 8160) (R6 = 0) (R7 = 8160)
R7 =  X'00000000_00001FE0'
R8 =  X'00000000_000003FC'      1020
```

Tips

1) The product will usually be found in the odd register. Products that require two grande registers will require special handling for conversion to packed decimal.