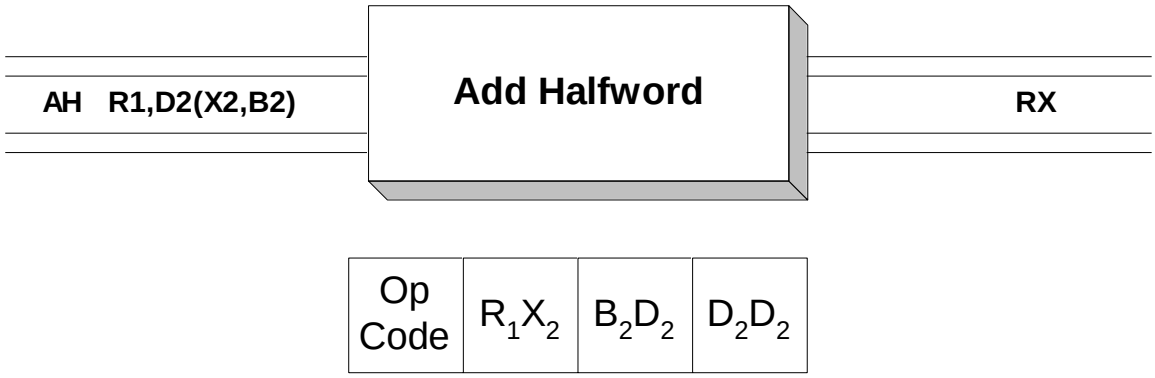
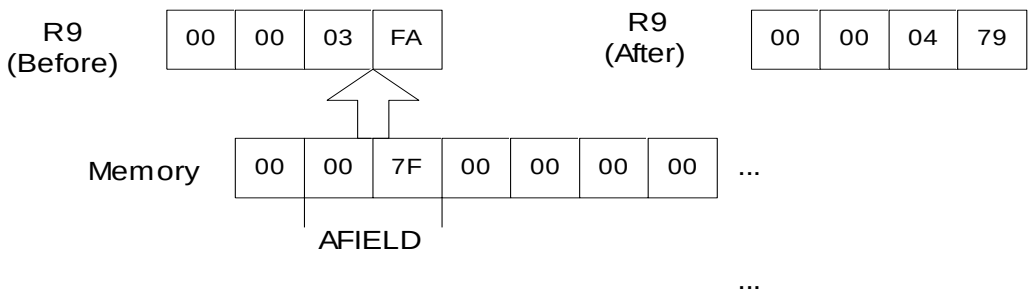


The Add Halfword instruction performs 2's complement binary addition. Operand 1 is a register containing a fullword integer. Operand 2 specifies a halfword in memory. The halfword in memory is sign extended (internally) and added to the fullword in the register. In other words, it is converted to an arithmetically equivalent fullword before the addition. The result is stored in the register, while the halfword in memory is not changed. Consider the following example,

```
AH    R9,AFIELD
```



The contents of the halfword "AFIELD", x'007F' = 127, are added to register 9 which contains x'000003FA' = 1018. The sum is 1145 = x'00000479' and destroys the previous value in R9. The halfword in memory is unchanged by this operation.

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

Examples

Some Unrelated Add Halfwords

```
R4 =  X'FFFFFFF6'  -10 IN 2'S COMPLEMENT
R5 =  X'00000031'  +49 IN 2'S COMPLEMENT
R6 =  X'00000008'  +8  IN 2'S COMPLEMENT
DOG   DC   H'4',H'9',H'-25',H'3'   CONSECUTIVE HALFWORDS

AH    R4,=H'20'    R4 = X'0000000A' = +10
AH    R5,=H'20'    R5 = X'00000045' = +69
AH    R6,=H'20'    R6 = X'0000001C' = +28
AH    R6,=H'-9'    R6 = X'FFFFFFFF' = -1
AH    R4,DOG       R6 = X'FFFFFFFA' = -6
AH    R6,DOG       R6 = X'0000000C' = +12
```

```
              AH    R4,DOG(R6)   R6 = X'FFFFFFDD' = -35 INDEXING
ALLOWED
```

Tips

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

```
        AMOUNT    DC    F'20'         AMOUNT = X'00000014'
                  ...
                  AH    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').