The Add instruction performs 2’s complement binary addition. Operand 1 is a register containing a fullword integer. Operand 2 specifies a fullword in memory. The fullword in memory is added to the fullword in the register and the result is stored in the register. The fullword in memory is not changed. Consider the following example,

```
A    R9,AFIELD
```

<table>
<thead>
<tr>
<th>Memory</th>
<th>AFIELD</th>
</tr>
</thead>
</table>
| 00 00 00 0A 00 00 2F | ...

The contents of the fullword “AFIELD”, x’0000000A’ = 10, are added to register 9 which contains x’00000025’ = 37. The sum is 47 = x’0000002F’ and destroys the previous value in R9. The fullword in memory is unchanged by this operation.

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

**Examples**

**Some Unrelated Adds**

- R4 = x’FFFFFFFE’ -2 IN 2’S COMPLEMENT
- R5 = x’00000028’ +40 IN 2’S COMPLEMENT
- R6 = x’00000004’ +4 IN 2’S COMPLEMENT

- DOG   DC   F’4’
- CAT   DC   F’-4’

- A  R4,=F’20’  R4 = X’00000012’ = +18
- A  R5,=F’20’  R5 = X’0000003C’ = +60
- A  R6,=F’20’  R6 = X’00000018’ = +24
- A  R6,=F’-5’  R6 = X’FFFFFFFE’ = -1
- A  R6,CAT     R6 = X’00000000’ = 0
A R6, DOG   R6 = X'00000008' = +8
A R6, DOG(R6)   R6 = X'00000000'   INDEXING IS ALLOWED