

**The Basic Arithmetic Verbs**

All basic arithmetic operations of ADD, SUBTRACT, MULTIPLY, and DIVIDE require the fields operated on :-

- 1) Have numeric PICTURE clause
- 2) Actually have numeric data when the program is executed.

**ADD Statement**

Format 1

```
ADD      { Identifier-1 } ..... TO  Identifier-2
         { Literal-1  }
```

Format-2

```
ADD      { Identifier-1 } .... GIVING Identifier-2
         { Literal-1  }
```

Examples

1. Add Deposit TO Balance (Result stored in Balance)
2. Add 15.80 TO Tax.
3. Add 40, OverTime-Hrs Giving Total-Hrs.
4. Add OveTime TO 40.

Note:

- 1) In example 1 and 2, Result is stored in Balance and Tax.
- 2) In example 3, the result is stored in Total-Hrs, and Total-Hrs may contain \$, or decimal if it is printed because the word after Giving (Total-Hrs) is not part of the Addition Operation
- 3) In example 4, the statement is ***illegal*** because the resultant filed must be an identifier or data-name. It can not be a Literal.

**Deciding whether to use GIVING or TO Format:**

Use Giving format when the content of the operand are to be RETAINED otherwise, use TO.

**Producing More Than One Sum**

It is possible to perform several Add Operations with a single statement.

```
ADD Amt1 Amt2 TO Total1 Total2.
```

The result is in the same series of operations

```
ADD AMT1 AMT2 TO Total1.
ADD AMT1 AMT2 TO Total2.
```

**SUBTRACT Statement**

Format 1

```
SUBTRACT { Identifier-1 } ..... FROM Identifier-2
          { Literal-1 }
```

Format-2

```
SUBTRACT { Identifier-1 } .. FROM { Identifier-2 } GIVING
          { Literal-1 }           { Literal-2 }
          Identifier-3 ....
```

Example

```
Subtract 15.4 Tax Total FROM Amt.
```

	Tax	Total	Amt
Before Subtract	30^00	10^00	100^00
After Subtract	30^00	10^00	44^60

Example

```
Subtract Amt1 Amt2 Amt3 FROM Total1 Total2 Total3.
```

This is the same as

```
Subtract Amt1 Amt2 Amt3 FROM Total1.
Subtract Amt1 Amt2 Amt3 FROM Total2.
Subtract Amt1 Amt2 Amt3 FROM Total3.
```

**MULTIPLY and DIVIDE Statement.**

**MULTIPLY**

Format 1

```
MULTIPLY { Identifier-1 } BY Identifier-2 ...
          { Literal-1 }
```

Format-2

```
MULTIPLY { Identifier-1 } BY { Identifier-2 } GIVING
          { Literal-1 }      { Literal-2 }
          Identifier-3 ....
```

Example

```
MULTIPLY QTY BY PRICE.
MULTIPLY QTY BY PRICE GIVING TOTAL.

MULTIPLY AMT BY WS-TOTAL, WS-TOTAL2.
```

**DIVIDE**

Format 1

DIVIDE	{ Identifier-1 Literal-1 }	INTO	Identifier-2 ...
--------	-------------------------------	------	------------------

Format-2

DIVIDE	{ Identifier-1 Literal-1 }	INTO	{ Identifier-2 Literal-2 }	GIVING
	Identifier-3 ....			

Format3

DIVIDE	{ Identifier-1 Literal-1 }	BY	{ Identifier-2 Literal-2 }	GIVING
	Identifier-3 ....			

Examples:

DIVIDE 3 INTO GRADES.

DIVIDE 3 INTO GRADES GIVING AVG.

DIVIDE MINUTES BY 60 GIVING HOURS.

DIVIDE 60 INTO MINUTES GIVING HOURS.

Example: Convert the Celsius temperature to Fahrenheit according to the following formula

$$\text{FAHRENHEIT} = (9/5) \text{CELSIUS} + 32.$$

CELSIUS is a field in the input area, and FAHRENHEIT is a field in the output area. Both have numeric picture clauses in DATA DIVISION.

Multiply 9 by CELSIUS.

Divide 5 into CELSIUS.

Add 32, Celsius Giving FAHRENHEIT.

**Use of REMAINDER Clause in the DIVIDE Operation**

It is sometimes useful to store the remainder of a division operation for additional processing. The DIVIDE can be used for this purpose by including a **REMAINDER** clause.

Format-1

```

DIVIDE      { Identifier-1 } INTO { Identifier-2 } GIVING
            { Literal-1   }
            Identifier-3 REMAINDER Identifier-4.
```

Format-2

```

DIVIDE      { Identifier-1 } BY { Identifier-2 } GIVING
            { Literal-1   }
            Identifier-3 REMAINDER Identifier-4
```

Example

Divide 130 by 40 Giving WS-Total REMAINDER WS-Remainder.  
 WS-Total will have 3 and WS-Remainder will have 10.

**Summary of How Arithmetic Operations are performed**

Arithmetic Statement	Value After Execution of the Statement			
	A	B	C	D
Add A To B	A	B		
Add A B C To D	A	B	C	A+B+C+D
Add A B C Giving D	A	B	C	A+B+C
Add A To B C	A	A+B	A+C	
Subtract A From B	A	B-A		
Subtract A B From C	A	B	C - ( A + B )	
Subtract A B From C Giving D	A	B	C	C - ( A + B )
Multiply A By B	A	A * B		
Multiply A By B Giving C	A	B	A * B	
Divide A Into B	A	B/A		
Divide A By B Giving C	A	B	B/A	
Divide A By B Giving C	A	B	A/B	
Divide A By B Giving C Remainder D	A	B	Integer Value of B/A	Integer Remainder



**The Compute Statement**

The COMPUTE statement provides another method of performing arithmetic.

The COMPUTE statement uses arithmetic symbols rather than arithmetic verbs. The following symbols may be used in the COMPUTE statement.

SYMBOL USED IN A COMPUTER	
Symbol	Meaning
+	ADD
-	SUBTRACT
*	MULTIPLY
/	DIVIDE
**	Exponentiation. (There is not corresponding COBOL Verb )

Examples:

1. Compute Tax = 0.5 \* Amt.
2. Computer Daily-Sales **ROUNDED** = Qty \* Unit-Price / 5.
3. Compute Total = Amt1 + Amt2 - Amt3.

	Contents Before Execution	Contents After Execution
Total	100	95
Amt1	80	80
Amt2	20	20
Amt3	5	5

Notes :

- 1) The fields specified after the equal sign in a COMPUTE statement may be numeric, literal of data-names with numeric picture clauses.
- 2) The Fields specified to the right of the equal sign in a COMPUTER statement remains unchanged.
- 3) The Compute Statement may include more than one operation.

Format

COMPUTE	Identifier-1 [ <b>ROUNDED</b> ] ... =	$\left\{ \begin{array}{l} \text{Arithmetic Expression} \\ \text{Literal} \\ \text{Identifier-2} \end{array} \right\}$
---------	---------------------------------------	---

**Order of Evaluation**

The order in which arithmetic operations are performed will affect the results in A COMPUTE Statement.

Example:

Compute Unit-Price-Out = Amt1-In + Amt2-In / Qty-In.

Is This equivalent to

Adding Amt1-In and Amt2-In and dividing the result by Qty-In OR  
 Dividing Amt2-In By Qty-In and adding the result to AMT1-In.

The two are not Identical the results will different.

<p>THE SEQUENCE IN WHICH OPERATIONS ARE PERFORMED IN A COMPUTE STATEMENT</p> <p>**</p> <p>* OR / ( Which ever appears first from left to right )</p> <p>+ OR - ( Which ever appears first from left to right )</p> <p>The use of parentheses overrides rules 1 - 3. That is, operations within parentheses are performed first.</p>
---

Example:

Compute A = C + D \*\* 2.

Order of evaluation :-

- 1) D \*\* 2 Exponentiation is performed first.
- 2) C + ( D \*\* 2 ) Addition is performed Next.

Homework

Code The following using the COMPUTE Statement

1)  $W = A X + B X - C$

2)  $Z = \frac{A B + X}{C D}$

3)  $L = \frac{O P + Q Y - K H}{R}$