Data Computation

This chapter discusses arithmetic statements that are used for computing data:

- COMPUTE
- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE

In addition, the following statements are discussed which are used to transfer the value of an operand into one or more fields:

- MOVE
- COMPRESS

Important:
For optimum processing, user-defined variables used in arithmetic statements should be defined with format P (packed numeric).

The following topics are covered:

- COMPUTE Statement
- Statements MOVE and COMPUTE
- Statements ADD, SUBTRACT, MULTIPLY and DIVIDE
- Example of MOVE, SUBTRACT and COMPUTE Statements
- COMPRESS Statement
- Example of COMPRESS and MOVE Statements
- Example of COMPRESS Statement
- Mathematical Functions
- Further Examples of COMPUTE, MOVE and COMPRESS Statements

COMPUTE Statement

The COMPUTE statement is used to perform arithmetic operations. The following connecting operators are available:

**	Exponentiation
*	Multiplication
/	Division
+	Addition
-	Subtraction
( )	Parentheses may be used to indicate logical grouping.

Example 1:

```
COMPUTE LEAVE-DUE = LEAVE-DUE * 1.1
```

In this example, the value of the field LEAVE-DUE is multiplied by 1.1, and the result is placed in the field LEAVE-DUE.

Example 2:

```
COMPUTE #A = SQRT (#B)
```

In this example, the square root of the value of the field #B is evaluated, and the result is assigned to the field #A.

SQRT is a mathematical function supported in the following arithmetic statements:

- COMPUTE
- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE

For an overview of mathematical functions, see Mathematical Functions below.

Example 3:

```
COMPUTE #INCOME = BONUS (1,1) + SALARY (1)
```

In this example, the first bonus of the current year and the current salary amount are added and assigned to the field #INCOME.

Statements MOVE and COMPUTE

The statements MOVE and COMPUTE can be used to transfer the value of an operand into one or more fields. The operand may be a constant such as a text item or a number, a database field, a user-defined variable, a system variable, or, in certain cases, a system function.

The difference between the two statements is that in the MOVE statement the value to be moved is specified on the left; in the COMPUTE statement the value to be assigned is specified on the right, as shown in the following examples.

Examples:

```
MOVE NAME TO #LAST-NAME
COMPUTE #LAST-NAME = NAME
```

Statements ADD, SUBTRACT, MULTIPLY and DIVIDE

The ADD, SUBTRACT, MULTIPLY and DIVIDE statements are used to perform arithmetic operations.

Examples:

```
ADD +5 -2 -1 GIVING #A
SUBTRACT 6 FROM 11 GIVING #B
MULTIPLY 3 BY 4 GIVING #C
DIVIDE 3 INTO #D GIVING #E
```

All four statements have a ROUNDED option, which you can use if you wish the result of the operation to be rounded.

For rules on rounding, see *Rules for Arithmetic Assignment*.

The *Statements* documentation provides more detailed information on these statements.

Example of MOVE, SUBTRACT and COMPUTE Statements

The following program demonstrates the use of user-defined variables in arithmetic statements. It calculates the ages and wages of three employees and outputs these.

```
** Example 'COMPUX01': COMPUTE
************************************************************************
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 NAME
  2 BIRTH
  2 JOB-TITLE
  2 SALARY         (1:1)
  2 BONUS          (1:1,1:1)
*
1 #DATE            (N8)
1 REDEFINE #DATE
  2 #YEAR          (N4)
  2 #MONTH         (N2)
  2 #DAY           (N2)
1 #BIRTH-YEAR      (A4)
1 REDEFINE #BIRTH-YEAR
  2 #BIRTH-YEAR-N  (N4)
1 #AGE             (N3)
1 #INCOME          (P9)
END-DEFINE
*
MOVE *DATN TO #DATE
*
READ (3) MYVIEW BY NAME STARTING FROM 'JONES'
  MOVE EDITED BIRTH (EM=YYYY) TO #BIRTH-YEAR
  SUBTRACT #BIRTH-YEAR-N FROM #YEAR GIVING #AGE
  /*
```

```
  COMPUTE #INCOME = BONUS (1:1,1:1) + SALARY (1:1)
  /*
  DISPLAY NAME 'POSITION' JOB-TITLE #AGE #INCOME
END-READ
END
```

Output of Program COMPUX01:

```
Page      1                                                  04-11-11  14:15:54

        NAME                 POSITION          #AGE  #INCOME
-------------------- ------------------------- ---- ----------

JONES                MANAGER                     63      55000
JONES                DIRECTOR                    58      50000
JONES                PROGRAMMER                  48      31000
```

COMPRESS Statement

The COMPRESS statement is used to transfer (combine) the contents of two or more operands into a single alphanumeric field.

Leading zeros in a numeric field and trailing blanks in an alphanumeric field are suppressed before the field value is moved to the receiving field.

By default, the transferred values are separated from one another by a single blank in the receiving field. For other separating possibilities, see the COMPRESS statement option LEAVING NO SPACE (in the *Statements* documentation).

Example:

```
COMPRESS 'NAME:' FIRST-NAME #LAST-NAME INTO #FULLNAME
```

In this example, a COMPRESS statement is used to combine a text constant ('NAME:'), a database field (FIRST-NAME) and a user-defined variable (#LAST-NAME) into one user-defined variable (#FULLNAME).

For further information on the COMPRESS statement, please refer to the COMPRESS statement description (in the *Statements* documentation).

Example of COMPRESS and MOVE Statements

The following program illustrates the use of the statements MOVE and COMPRESS.

```
** Example 'COMPRX01': COMPRESS
************************************************************************
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
  2 MIDDLE-I
*
1 #LAST-NAME (A15)
1 #FULL-NAME (A30)
END-DEFINE
*
```

```
READ (3) MYVIEW BY NAME STARTING FROM 'JONES'
  MOVE NAME TO #LAST-NAME
  /*
  COMPRESS 'NAME:' FIRST-NAME MIDDLE-I #LAST-NAME INTO #FULL-NAME
  /*
  DISPLAY #FULL-NAME (UC==) FIRST-NAME 'I' MIDDLE-I (AL=1) NAME
END-READ
END
```

Output of Program COMPRX01:

Notice the output format of the compressed field.

```
Page      1                                                 04-11-11  14:15:54

          #FULL-NAME                FIRST-NAME      I          NAME
============================== ------------------- - --------------------

NAME: VIRGINIA J JONES         VIRGINIA            J JONES
NAME: MARSHA JONES             MARSHA                JONES
NAME: ROBERT B JONES           ROBERT              B JONES
```

In multiple-line displays, it may be useful to combine fields/text in a user-defined variables by using a COMPRESS statement.

Example of COMPRESS Statement

In the following program, three user-defined variables are used: #FULL-SALARY, #FULL-NAME, and #FULL-CITY. #FULL-SALARY, for example, contains the text 'SALARY:' and the database fields SALARY and CURR-CODE. The WRITE statement then references only the compressed variables.

```
** Example 'COMPRX02': COMPRESS
************************************************************************
DEFINE DATA LOCAL
1 VIEWEMP VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
  2 SALARY       (1:1)
  2 CURR-CODE    (1:1)
  2 CITY
  2 ADDRESS-LINE (1:1)
  2 ZIP
*
1 #FULL-SALARY   (A25)
1 #FULL-NAME     (A25)
1 #FULL-CITY     (A25)
END-DEFINE
*
READ (3) VIEWEMP BY CITY STARTING FROM 'NEW YORK'
  COMPRESS 'SALARY:' CURR-CODE(1) SALARY(1) INTO #FULL-SALARY
  COMPRESS FIRST-NAME NAME                  INTO #FULL-NAME
  COMPRESS ZIP CITY                         INTO #FULL-CITY
  /*
  DISPLAY 'NAME AND ADDRESS' NAME (EM=X^X^X^X^X^X^X^X^X^X^X^X)
  WRITE 1/5  #FULL-NAME
        1/37 #FULL-SALARY
        2/5  ADDRESS-LINE (1)
```

```
        3/5  #FULL-CITY
  SKIP 1
END-READ
END
```

Output of Program COMPRX02:

```
Page      1                                                  04-11-11  14:15:54

   NAME AND ADDRESS
-----------------------

R U B I N
    SYLVIA RUBIN                    SALARY: USD 17000
    2003 SARAZEN PLACE
    10036 NEW YORK

W A L L A C E
    MARY WALLACE                    SALARY: USD 38000
    12248 LAUREL GLADE C
    10036 NEW YORK

K E L L O G G
    HENRIETTA KELLOGG               SALARY: USD 52000
    1001 JEFF RYAN DR.
    19711 NEWARK
```

Mathematical Functions

The following Natural mathematical functions are supported in arithmetic processing statements (`ADD`, `COMPUTE`, `DIVIDE`, `SUBTRACT`, `MULTIPLY`).

Mathematical Function	Natural System Function
Absolute value of *field*.	ABS(*field*)
Arc tangent of *field*.	ATN(*field*)
Cosine of *field*.	COS(*field*)
Exponential of *field*.	EXP(*field*)
Fractional part of *field*.	FRAC(*field*)
Integer part of *field*.	INT(*field*)
Natural logarithm of *field*.	LOG(*field*)
Sign of *field*.	SGN(*field*)
Sine of *field*.	SIN(*field*)
Square root of *field*.	SQRT(*field*)
Tangent of *field*.	TAN(*field*)
Numeric value of an alphanumeric *field*.	VAL(*field*)

See also the *System Functions* documentation for a detailed explanation of each mathematical function.

Further Examples of COMPUTE, MOVE and COMPRESS Statements

See the following example programs:

- *WRITEX11 - WRITE (with nX, n/n and COMPRESS)*
- *IFX03 - IF statement*
- *COMPRX03 - COMPRESS (using parameters LC and TC)*