Subject: Edits for .ANDTHEN. and .ORELSE.
From: Van Snyder
Reference: 03-258r1, section 2.8.2; 04-193, 04-192, 04-357

1 Introduction

The concept of *and* and *or* operators that are guaranteed to short circuit evaluation was described in 03-258r1 and 04-193. On the "hate it, dislike it, like it, love it" scale it was rated 0, 1, 6, 3. On the "small, medium, large" scale it was rated 9, 2, 0. JOR later rated it "medium."

The brevity of the edits presented here suggest it really does belong at "small."

The precedence of .ANDTHEN. is proposed to be immediately below that of .AND., while the precedence of .ORELSE. is proposed to be immediately below that of .OR. If the precedence were the same, `A .AND. B .ANDTHEN. C` could be parsed as `(A .AND. B) .ANDTHEN. C` or as `A .AND. (B .ANDTHEN. C)`. In the first case, one can be certain that `C` is not evaluated if either `A` or `B` is false. In the second case, one can only be sure that `C` is not evaluated if `B` is false. Similarly, `A .ANDTHEN. B .AND. C` could be parsed either as `(A .ANDTHEN. B) .AND. C` or `A .ANDTHEN. (B .AND. C)`. In the first case, one can be certain that `B` is not evaluated if `A` is false, while in the second case one can be certain that neither `B` nor `C` is evaluated if `A` is false. Similar arguments apply to .ORELSE. The standard should not be so ambiguous.

It is not proposed to put the precedence of .ANDTHEN. and .ORELSE. below .EQV. and .NEQV. because it is likely that programmers will change .AND. to .ANDTHEN. or vice-versa, and similarly for .OR. and .ORELSE. The reason to change AND. to .ANDTHEN. is a discovery that something in the second operand is undefined if the first operand is false. The reason for the opposite change is a discovery that everything in the second operand is defined no matter whether the first operand is false, and using .ANDTHEN. causes performance problems. Assuming the parentheses used here to indicate precedence aren't actually present, it would be unwise to arrange that `(A .AND. B) .EQV. (C .AND. D)` becomes `A .ANDTHEN. (B .EQV. C) .ANDTHEN. D`, and vice-versa.

The semantical property of these operators that their second operand is not evaluated if the first is false (true) could be provided by conditional expressions (04-192) or a conditional-execution intrinsic function (04-357), *viz.* `A .ANDTHEN. B` could be represented `A ? B : .FALSE.` or `IF ( A, B, .FALSE. )` and `A .ORELSE. B` could be represented as `A ? .TRUE. : B` or `IF ( A, .TRUE., B )` . Thus, if the proposal for conditional expressions proceeds, this proposal is somewhat redundant.

2 Edits

Edits refer to 04-007. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by associated text, while a page and line number followed by + (-) indicates that associated text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

R719½ *andthen-op* **is** .ANDTHEN. 26:25+

R720½ *orelse-op* **is** .ORELSE. 26:26+

[Insert "and .ANDTHEN." after ".AND" and "and .ORELSE." after ".OR."."] 44:14

R714½ *andthen-operand* **is** [*andthen-operand and-op*] *and-operand* 120:5-6
R715 *or-operand* **is** [*or-operand andthen-op*] *andthen-operand*
R715½ *orelse-operand* **is** [*orelse-operand or-op*] *or-operand*
R716 *equiv-operand* **is** [*equiv-operand orelse-op*] *orelse-operand*

R719½ *andthen-op* **is** .ANDTHEN. 120:9+

R720½ *orelse-op* **is** .ORELSE. 120:10+

[Add ", .ANDTHEN." after ".AND." and ", .ORELSE." after ".OR." in the first column of Table 7.1.] 121:7+17

[Add ", .ANDTHEN." after ".AND." and ", .ORELSE." after ".OR.".] 121:20

[Replace "Once" by "For the .AND., .OR., .EQV., and .NEQV. operators, once".] 132:4

For the .ANDTHEN. operator the processor shall not evaluate the second operand if the first is false. For the .ORELSE. operator, the processor shall not evalate the second operand if the first is true. Otherwise, once the interpretation of an expression has been established in accordance with the rules given in 7.2.4, the processor may evaluate any other expression that is logically equivalent, provided that the integrity of parentheses in any expression is not violated. 132:8- New ¶

[Insert two new rows in Table 7.5:] 135:28+4,5+

.ANDTHEN.	Logical conjunction	x_1 .ANDTHEN. x_2	True if x_1 and x_2 are both true, but x_2 shall not be evaluated if x_1 is false
.ORELSE.	Logical inclusive disjunction	x_1 .ORELSE. x_2	True if either x_1 or x_2 is true, but x_2 shall not be evaluated if x_1 is true

[In the heading of Table 7.6, Add "x_1 .ANDTHEN. x_2" under "x_1 .AND. x_2" and "x_1 .ORELSE. x_2" under "x_1 .OR. x_2".] 136:1+2+

[In Table 7.7, replace the .OR. row] 136:5+13

Logical	.ANDTHEN.	·
Logical	.OR.	·
Logical	.ORELSE.	·