INTERNAL USE ONLY

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

Scientific Programming Department
ICT 1900 Series

FORTRAN NOTE 15
29.9.65.

The Statement Function and IF Mechanism in the ICT FORTRAN IV Compiler

General

Statements thought to be Assignment, and Statements known to be IF, are passed through the Expression Evaluat or at entry FPNE1. All other entries to the Expression Evaluat or are at either entry FPNE or FPNB.

If a construction of the form 'name (' is found at the start of a statement during use of the Expression Evaluator, entry FPNE1, the statement may be either a Statement Function Definition or an IF statement or an Assignment statement starting with an array element.

If and only if this construction is found an operator #120 is stored in the operator list before the '(' is put there. This operator is in place of the f or a operation which might otherwise be put into the list. This operator is the 'IFQ' operator and is used to delay having to make a commitment about the type of operation until the last possible moment. Further the 'name' is stored in a special area (SECRET) and no Main List Item is made for it. The PN list item for this name is a c/m to 'SECRET'.

When this operator (IFQ) is forced out of the Operator list the operands will be in the standard form for an Array or Function operation.

This operator is the one exception to the rule that a ')' forces out all operators until '(' and is then anihilated along with the ')'. If the operator which precedes '(' in the list is the operator IFQ, it is forced out by the influence of the ')'.

Considering the Operands:

If the top operand is mode LOGICAL, the statement is assumed to be a Logical IF statement.

If the field following the ')' in the statement is an integer, then the statement is assumed to be an Arithmetic IF statement.

If neither of these cases apply, the PN list item for the 'name' (which holds a c/m to SECRET) is used to enable the name to be looked for in the Compiler's Main List. If it is not there or is there but is undefined at to 'type', the statement is assumed to be a Statement Function Definition.

If none of these cases apply, the statement is considered an Assignment Statement which is introduced by an array element. In this case the PN list item for the 'name' is changed to that for the array name, and an array operator is put in the top of the operator list. This puts the PN and Operator lists back in a standard condition.

All IF statements are passed through the expression evaluator using entry FPNE1. This is due to the fact that in the early stages of investigation, certain logical IF statements can not be distinguished from Assignment statements.

Logical IF

At the stage in the expression evaluation at which the IFQ operator is recognized, the parenthesized logical expression of the IF will have been evaluated. The PN list item for the name in 'SECRET', which must be the second operand is looked at to see whether the name is 'IF'. If it is not, the construction is illegal. If it is IF, the expression is compiled in X6 unless it is already there, the current transfer address is stored in a stack (of length 9) and the subsidiary statement is generated. On exit from the Statement generator, a conditional branch from the stored T.A. to the current TA completes the IF statement.

A stack link system is used in the Statement Generator because it is being called recursively from itself. This is due to the Statement Generator finding IF in the first place and also having to generate the subsidiary operations.

Arithmetic IF

At the stage in the expression evaluation at which the IFQ operator is recognized, the parenthesized arithmetic expression of the IF will have been evaluated. The PN list item for the name in 'SECRET', which must be the second operand, is looked at to see whether the name is IF. If it is not, the construction is illegal. If it is IF, the expression is compiled into X6 unless it is already there. The IF successors are then investigated and suitable branch instructions compiled taking into consideration any useful facts known about the successors (e.g. two successors the same).

Statement Function Definition

At the time a Statement function definition is recognized, the PN list has processed all the arguments for that definition. For a suitable definition no operand may be ACC or X3. i.e. no operand was originally an expression.
Since no expressions have been evaluated, no items will have been assigned object program space during the scan of the argument. Thus at the time that the construct is recognized as the start of a Statement Function definition the argument can be reappraised without having to decompile any of the statement.

At this stage the statement up to the end of the construct 'name ( )' is rescanned outside the PN list (i.e. linearly). The name of the Statement Function is put in the main list (as a S.F.) and the arguments are processed into the Main List as S.F. arguments. If the character following the construct is not '=', the whole construction is illegal otherwise a subroutine prologue is compiled.

The statement is then scanned again from the beginning of the line, using the Expression Evaluator - this time using entry FPNE.

Eventually this scan will require that the f operation for 'name ( )' be evaluated. It is determined that the operation is a SF because the 'name' is so marked. It can be determined that it is a definition statement because the character following the ')' is '='. The f operation under these circumstances merely changes the 'name' operand to a special operand (#210/0 + mode in the counter) which indicates 'S.F. Definition'. This will eventually be processed by the '=' operator after the r.h.s. expression has been evaluated.

If the '=' operator discovers a bottom operand of this type, it will only perform mode auxiliary and lac operations; It will not generate store instructions.

V.K. Taylor.