Editor's Note

In the July 1964 issue of the Communications, this department solicited papers, notes, remarks, and the like on the subject of "A Successor to ALGOL". While there have been several items received and published on the subject, the response has been, on the whole, rather meager considering the amount of heat that gets generated on the subject. We therefore want to again request such material, and this time solicit particularly material which bears on the "environmental" aspects of programming languages: coping with the generation of computers featuring various kinds of parallelism, dealing with files, features desired in a time-sharing facility, and so on. The following paper by Opler addresses one such question.—T.E.C.

Procedure-Oriented Language Statements to Facilitate Parallel Processing

ASCHER OPLER
Computer Usage Company, Inc., New York, N. Y.

Two statements are suggested which allow a programmer writing in a procedure-oriented language to indicate sections of program which are to be executed in parallel. The statements are DO TOGETHER and HOLD. These serve partly as brackets in establishing a range of parallel operation and partly to define each parallel path within this range. DO TOGETHERs may be nested. The statements should be particularly effective for use with computing devices capable of attaining some degree of compute-compute overlap.

Computers with parallel processing capabilities are seldom used to full advantage. In some systems, more than one single program is processed with simultaneity while in others, different portions of a single program are processed in parallel. In the former, inefficiency often results because the mix of individual programs, each written for sole occupancy of a computer, is unlikely to demand equal loading of each parallel element. In the latter case, the distribution of program functions to hardware elements is frequently left to computer logic (e.g. input-output commands are sent to a special processor, floating-point arithmetic commands to another, and so forth.)

The following is directed toward better utilization of computer systems which process a single program by performing functionally different portions with separate computing elements. The Bull Gamma 60 and the CDC 6000 series are representative of this class.

Procedure-oriented languages developed for serial computation have serious limitations when used to express problem solutions involving parallelism since the control statements (GO TO, DO, FOR, IF, etc.) define a single serial path for the computation.

Two statements are suggested as possible additions to these languages (ALGOL, FORTRAN, COBOL, etc.) to facilitate the efficient application of parallel computers. The statements provide the analyst with a tool for stating which procedures may be executed in parallel. They also provide the compiler designer with a language element that allow the compiler to produce object programs that can properly use parallel multiple processing computer logic.

The two statements are DO TOGETHER and HOLD. One suggested format is described below. The effect of these statements is to establish a *range* of parallel operation and to define two or more parallel *paths* within this range. The range begins with the DO TOGETHER and ends at the HOLD referenced by the former. The object program will not continue execution beyond the HOLD until the executable statements in all paths have been processed (see Figures 1 and 2).

Format

Label_1 DO TOGETHER
Label_2 , Label_3 , · · · , Label_{n-1} (Label_n)

Label_1 (optional) is the tag of the beginning of the range.
Label_n (required) is the tag of the HOLD that terminates the range. It is always enclosed in parentheses.
Label_2 to Label_{n-1} are tags of the first statements in each of the n-2 paths.

The description of a path is terminated when either the label starting another path or the referenced HOLD is encountered:

Label HOLD

The Label is mandatory and must be referenced by one or more DO TOGETHERs.

Some Rules for DO TOGETHER

1. Each path must be logically self-contained. Intrapath branching is permitted; interpath branching is not allowed.
2. Branching into or out of the range of a DO TOGETHER is not permitted.
3. Paths in the same DO TOGETHER may reference the same variables but must not alter these.
4. DO TOGETHERs may be nested. A path in the range of a DO TOGETHER may itself contain the range of another DO TOGETHER.
5. Nested DO TOGETHERs may share the same HOLD.
6. Logical decisions made within a path should set switches for interrogation after the HOLD. This avoids conflict with rule 2.
7. Subroutine or procedure calls may be made within a path. Paths in the same range may not call the same subroutine unless it is re-entrable.

Implementation

1. The object code for each path is compiled.
2. A completion notification mechanism is compiled for each path.
3. Depending upon the degree and nature of parallelism permitted by the object computer, the various instruction strings are merged to produce an optimum parallel program.
4. At the sequential location corresponding to the HOLD statement, an interlock mechanism (as dictated by the logical design of the computer) is compiled. This interlock can only be released when all paths have forwarded their termination signal.
5. Suitable variations will permit processing of nested DO TOGETHERs.

Sample Applications

For computers with read-compute, compute-write and/or read-compute-write "overlap," the use of these statements is relatively straightforward. Compilers for use with such computers have generally been designed to take advantage of *implicitly* declared parallelism. With these two statements *explicitly* directed parallelism may be used.

For devices with compute-compute parallel capability, these statements should lead to better analysis. For the computer-oriented analyst, these provide a means for dividing a single task into subtasks that may be performed in parallel or for arranging for the concurrent processing of two independent tasks. For the numerical analyst it should lead to study and identification of computational aspects of a solution that can be performed with simultaneity. For instance, computing the check sum of a row, finding pivot points and operation on another row of a matrix may be performed simultaneously. In a matrix multiplication, several row-column combinations can be worked on simultaneously (see Figures 3a and 3b).

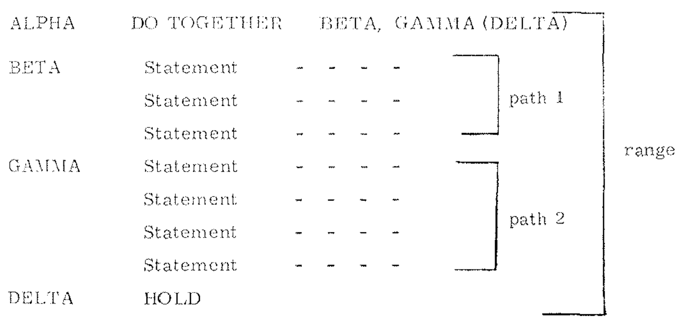

FIG. 1. Structure of a DO TOGETHER

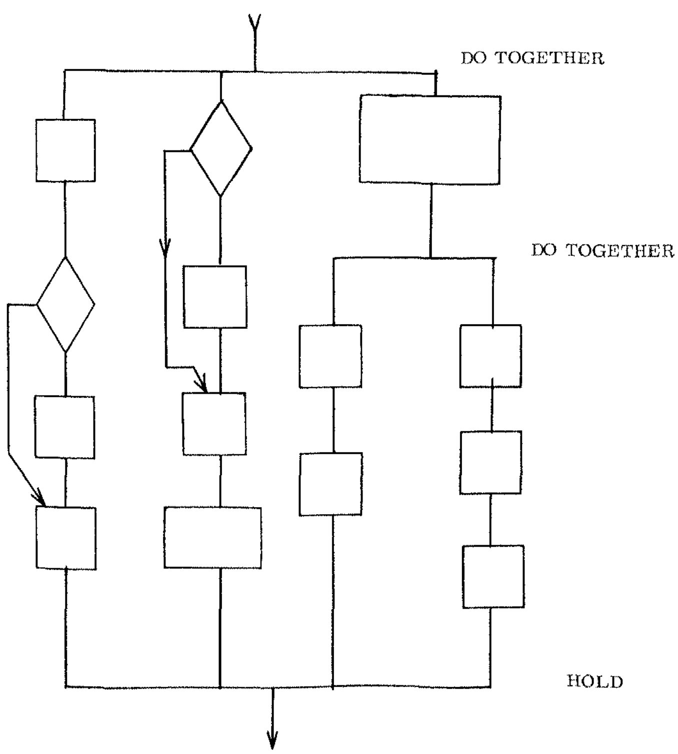

FIG. 2. Nested DO TOGETHERs

```
      DO 6   I=1,21
      DO 6 J=1,21
      DO 6 K=1,21
6     C(I,J)= C(I,J)+A(I,K)*B(K,J)
```

FIG. 3a. 21st-order matrix-multiplication statements in a (serial) FORTRAN program (multiplication performed serially 9261 times)

```
77    DO TOGETHER 1,2,3,4,5(6)
1     DO 11 I1=1,21,5
      DO 11 J1=1,21
      DO 11 K1=1,21
11    C(I1,J1)= C(I1,J1)+A(I1,K1)*B(K1,J1)
2     DO 22 I2=2,17,5
      DO 22 J2=1,21
      DO 22 K2=1,21
22    C(I2,J2)= C(I2,J2)+A(I2,K2)*B(K2,J2)
3     DO 33 I3=3,18,5
      DO 33 J3=1,21
      DO 33 K3=1,21
33    C(I3,J3)= C(I3,J3)+A(I3,K3)*B(K3,J3)
4     DO 44 I4=4,19,5
      DO 44 J4=1,21
      DO 44 K4=1,21
44    C(I4,J4)= C(I4,J4)+A(I4,K4)*B(K4,J4)
5     DO 55 I5=5,20,5
      DO 55 J5=1,21
      DO 55 K5=1,21
55    C(I5,J5)= C(I5,J5)+A(I5,K5)*B(K5,J5)
6     HOLD
```

FIG. 3b. 21st-order matrix-multiplication statements in a FORTRAN program for a computer with 5 multiplication units and other parallel operating registers (multiplication performed 1764 times in each of four units and 2205 times in the fifth)

Acknowledgments. The idea of DO TOGETHER was first mentioned (1959) by Mme. Jeanne Poyen in discussing the AP3 compiler for the BULL Gamma 60 computer. The two statements proposed here may be considered source language relatives of Conway's Fork and Join instructions [1].

RECEIVED JUNE, 1964; REVISED SEPTEMBER, 1964

REFERENCE

1. CONWAY, M. E. A multiprocessor system design. Proc. Fall Joint Comput. Conf. *24*, Spartan Books, Baltimore, 1963.