OAK RIDGE NATIONAL LABORATORY
operated by
UNION CARBIDE CORPORATION
for the
U.S. ATOMIC ENERGY COMMISSION

ORNL - TM - 221

COPY NO. - 4

DATE - May 4, 1962

A TRANSLATOR-ORIENTED INTERMEDIATE COMPUTER LANGUAGE

A. A. Grau

A Translator-Oriented Intermediate Computer Language

A. A. Grau

Description. The language whose operations are described is the result of considering the translation process for Algol programs based on the methods of Bauer and Samelson[1], and its requirements in terms of target language. The requirements made of the target language were that it be parenthesis-free, that it contain a minimum number of instruction types, and that it would simplify the translation process as much as possible. No close connection to any hardware considerations were considered in its design.

Since the translation of Algol programs is necessarily a recursive process, the language is framed in terms of a push-down list H in the target program, which consists of the memory cells, $\eta_1, \eta_2, \ldots, \eta_h, \ldots$ reserved for this purpose.

In static translation, the push-down counter h is in the translator; the push-down level is thus supplied to the target program at each point in the program where needed at translation time. If nested procedures, functions, and especially recursive procedures are to be implemented by the translator, the push-down counter must be put into the target program itself to permit dynamic use of it there. The effect of moving the push-down counter into the target program is that the execution of each binary or unary operation and each fetch or store operation is accompanied by an effect on this counter. This effect is, therefore, naturally made part of the intermediate operation as seen below.

In the design of this language, the stimulation of conversations with H. H. Bottenbruch must be acknowledged.

[1] Bauer and Samelson, Comm. Assoc. Comput. Mach. 3 (1960), pp. 76-82.

The Role of this Intermediate Language. The language described has three possible uses.

1. An intermediate language in the translation process. It is machine independent and therefore can serve as a universal intermediate language in the process of translation. The remaining part of the translation consists of replacing the instructions of the intermediate language by machine code in practically a one-to-one fashion. If the dynamic aspects are desired, provision for incorporating these in the target program must also be made.

2. An intermediate language to be handled by an interpreter. In this case Algol translation is resolved into a translation to this language phase, and an interpretation phase. In the latter the program in this intermediate language is placed in memory along with an interpreter program. Though execution time is slowed by this, the dynamic requirements on the push-down counter are automatically fulfilled by this method.

3. A guide for possible new machine operations. This language consists of the condensation of the instructions necessary on a computer in the translation of an algorithmic language such as Algol. Since the use of this language as target language simplifies the machine translation process, it follows that it also to some extent simplifies the writing of hand code dealing with numerical processes. For this reason, the feasibility of the direct implementation of this language on a computer by means of suitable hardware should be investigated.

Operations of the Intermediate Language

mbol	Counter	Operation	Counter	Comment
Arithmetic, Relational, and Boolean Operations.				
ω		$C(\eta_{h-1})\ \omega\ C(\eta_h) \rightarrow C(\eta_{h-1})$	$h := h-1$	Binary Operation or relation.
ω		$\omega\ C(\eta_h) \rightarrow C(\eta_h)$		Unary operation or standard function.
. Storage Operations.				
A m	$h := h+1$	$m \rightarrow C(\eta_h)$		Store address.
T m	$h := h+1$	$C(m) \rightarrow C(\eta_h)$		Store value.
V		$C(\eta_h) \rightarrow C(C(\eta_{h-1}))$		Assign value.
B		$C(C(\eta_h)) \rightarrow C(\eta_h)$		Store value of variable whose address was previously stored.
I. Linkages.				
G		go to $C(\eta_h)$	$h := h-1$	Transfer to stored address.
C		if $C(\eta_{h-1})$ = false then go to $C(\eta_h)$	$h := h-2$	Conditional transfer (if condition is not met).
. Halt.				
H		stop		Stop computer.
Counter control.				
D m			$h := h-C(m)$	Reduce counter.
. Subscripting Operation.				
S m		*(See note 3)	$h := h-r+1$	Store address of subscripted variable.

Notes.

1. The symbol ω. The symbol ω denotes generically any one of the operations $+ \uparrow - \times / \div \vee \wedge \subset \equiv \neg \oplus \ominus < \leq = \geq > \neq$. The unary functions include the basic operations listed in the Algol report for which identifiers are reserved, such as abs, sin, entier, etc.

2. The symbol $\rightarrow$. This symbol may be read "becomes." Thus $\omega\, C(\eta_h) \rightarrow C(\eta_h)$ may be read: "The result of operating with ω on the contents of η_h becomes the content of η_h."

3. The subscripting operation *. The subscripting operation uses the contents of m, m+1, ..., m+r and the contents of $\eta_h, \eta_{h-1}, \ldots, \eta_{h-r+1}$ to determine the address of the variable in question. The quantities stored in m ff. are:

$$\begin{array}{rl} m: & \text{address of the variable } I\,[0, 0, \ldots, 0]; \\ \left.\begin{array}{r} m+1: \\ m+2: \\ \cdots \\ m+r-1: \end{array}\right\} & \text{multipliers for } i_1, \ldots, i_{r-1}. \end{array}$$

The address of the variable is the address I [0, 0, ..., 0] + L where L is determined by the iterative relation:

```
L := 0 ;
for k := 1 step 1 until r-1 do
L := L + C(m+k) × C(η_{h-r+k});  L := L + η_h
```

This address is then stored in η_{h-r+1}.

The list suffices for the handling of all types of Algol statements not calling procedures. As far as has been ascertained at this time, the list should suffice for the handling of these also. No claim, however, is made that the list is exhaustive.

Distribution

1-3. DTIE, ORO
4. M. J. Skinner
5-6. Central Research Library
7. Document Reference Section
8. A. S. Householder