Since R and T consist of x's with one occurrence of :=, P_i for $i > 0$ must take the form

begin real $x^{(f(i))}$; $x^{(f(i))} := x^{(f(i))} := \cdots := x^{(f(i))}$ **end**

containing i occurrences of :=, $i+2$ occurrences of $x^{(f(i))}$, and therefore $(i+2)(f(i))$ occurrences of x. Since the number of occurrences of x is a linear function $a + bi$ of i, we have

$$f(i) = \frac{a + bi}{2 + i} = \frac{a - 2b + b(2 + i)}{2 + i} = b + \frac{a - 2b}{2 + i}$$

always integer-valued. Then $a - 2b$ is zero, so $f(i) = b$; and the number of occurrences of x in P_i for all $i \geqq 0$ is $(i + 2)b$. Then P_1 is

begin real $x^{(b)}$; $x^{(b)} := x^{(b)}$ **end**

and P_0 is

begin real $x^{(2b)}$; **end**

which is not a subsequence of P_1 and cannot be obtained from P_1 by deletions.

The conclusion to be drawn is that it is not possible to state the formation rules of ALGOL 60 as a phrase structure grammar, so that there must necessarily be syntactic rules stated in other ways. The principal examples are the rules requiring the declaration of all variables, procedures, arrays and switches. It seems likely that similar considerations would apply to any other reasonable language in which all variables must be declared.

REFERENCES

1. NAUR, PETER (Ed.) ET AL. Report on the algorithmic language ALGOL 60. *Comm. ACM 3*, 5 (1960), 299–314.
2. CHOMSKY, N. On certain formal properties of grammars. *Information and Control 2* (1959), 137–167; A note on phrase structure grammars. *Information and Control 2* (1959), 393–395.
3. BAR-HILLEL, Y., PERLES, M., AND SHAMIR, E. On formal properties of simple phrase structure grammars. *Zeit. Phonetik, Sprachwissenschaft und Kommunikationsforschung 14* (1961), 143–172.

TALL—A List Processor for the Philco 2000 Computer

Julian Feldman

System Development Corporation, Santa Monica, California

Several of the computer languages that are oriented toward problems in symbol manipulation use a list type of memory organization.[1] The advantages of such a memory organization have been discussed elsewhere and will not be repeated here. The purpose of this note is to describe the method used in realizing a list language on the Philco 2000.

Information Processing Language V (IPL-V) was chosen as the source language for the list processor for the 2000 because this language has been well documented and has been implemented on several computers.[2] Heretofore, IPL-V has been implemented as an interpretive system. The interpretive system has three major components: (1) a loader which translates card images into internal machine words; (2) an interpreter which decodes instructions; and (3) a set of primitive processes, the "J's," which make up the bulk of the instruction vocabulary. The implementation of such an interpretive system has been a rather lengthy procedure usually estimated as taking six man-months.

IPL-V has been implemented on the 2000 as a set of macro-operations, subroutines and conventions supplementing TAC (*T*ranslator-*A*ssembler-*C*ompiler, the assembly language for the 2000).[3] These macro's, subroutines and conventions will be referred to as TALL (*TAC Li*st *L*anguage). TALL uses the loading facilities of TAC, the IPL-V primitive processes, and a set of subroutines performing the work of the interpreter. The macros aid in the translation from IPL-V to TAC. The macros and the primitive processes, the J's, can be placed on the TAC subroutine library tape and called in as required during assembly.

The implementation of IPL-V in this fashion has several advantages: (1) the time required to get a basic IPL-V system running on the 2000 was only three man-weeks; (2) symbolic machine language instructions can easily be inserted into TALL programs; (3) IPL-V statements can be used in conjunction with FORTRAN statements or JOVIAL statements;[4] and (4) no additional work is required to make TALL compatible with any monitor system for the 2000. A brief description of the TALL representations of IPL-V program and data follows.

TALL Program

The IPL-V program word has the format

P Q SYMB LINK

where P is an octal digit representing an operation code, Q is an octal digit specifying the degree of indirection represented by SYMB, SYMB is a machine address, and LINK is the machine address of the next instruction. In the TALL system, the P-Q combinations are represented as macro-operations which have SYMB and LINK as inputs. Thus the IPL-V program word is represented by the following line of TAC code:

L COMMAND ADDRESS

PQnn SYMB; LINK

The macro PQnn expands this line of code into two computer words. The first word has SYMB in the address of the left half-word and LINK in the address of the right half-word. The second word has a left half-word instruction which loads the first word into the A-register and a right half-word instruction which transfers to the subroutine PQnnX which finds its input parameters, SYMB and LINK, in the A-register. The conversion of program from IPL-V format to TALL format is a rather simple and straightforward procedure that can easily be accomplished by EAM equipment (an example is provided in the Appendix).

TALL Data

The IPL-V data word takes on various forms. The format for IPL-V symbolic data is the same as the format for program. The TALL format for symbolic data is the same as the program format with the exception that a "D" is added after the "PQnn."

[1] Program and Preprints of the ACM Conference on Symbol Manipulation. *Comm. ACM 3*, 4 (1960).

[2] NEWELL, A., ET AL. *Information processing language V manual*. Englewood Cliffs, Prentice-Hall, 1961.

[3] Philco 2000 TAC Manual. Philco Corp., Computer Div., Willow Grove, Penn., May 1961.

[4] Philco 2000 ALTAC Manual. Philco Corp., Computer Div., Willow Grove, Penn., Feb. 1961; C. J. SHAW, JOVIAL Manual. TM-555, System Development Corp., Santa Monica, Calif., 1961.

IPL-V PROGRAM				TALL PROGRAM				COMMENTS	
T	NAME	S	PQ SYMB LINK	L	LOCATION	COMMAND	ADDRESS		
1	ACKERMANN-S		FUNC	I		ACKERMANN-S FUNC	$	IDENTIFY PROGRAM	00
2	EO		1					DEFINE IPL-V REGIONS	01
2	AO		1					NOT REQUIRED IN TALL.	02
2	MO		1						03
2	NO		1						04
2	KO		1						05
5		00				NAME EO	$	BEGIN FIRST ROUTINE	06
	EO	03	AO	C	EO	PQ03 AO ; (P) + 1	$	THE LINK "(P) +1" MUST BE	07
		10	NO			PQ10 NO ; (P) + 1	$	SUPPLIED FOR TALL.	08
		00	J152 0			PQ00 J152; 0	$		09
5		00				NAME AO	$		10
	AO	14	MO	C	AO	PQ14 MO ; (P) + 1	$	THE "C" IDENTIFIES AO AS	11
		00	J117			PQ00 J117; (P) + 1	$	A COMMON SYMBOL.	12
		10	NO			PQ10 NO ; (P) + 1	$		13
		70	9-1			PQ70 9L1 ; (P) + 1	$	THE "-" SIGN MUST BE	14
		00	J125 J8			PQ00 J125; J8	$	REPLACED BY AN ALPHA	15
	9-1	00	J117		9L1	PQ00 J117; (P) + 1	$	CHARACTER	16
		70	9-2			PQ70 9L2 ; (P) + 1	$		17
		10	NO			PQ10 NO ; (P) + 1	$		18
		00	J125 9-3			PQ00 J125; 9L3	$		19
	9-2	10	K1		9L2	PQ10 K1 ; (P) + 1	$		20
		10	NO			PQ10 NO ; (P) + 1	$		21
		10	NO			PQ10 NO ; (P) + 1	$		22
		00	J111			PQ00 J111; (P) + 1	$		23
		00	AO			PQ00 AD ; (P) + 1	$		24
	9-3	50	K1		9L3	PQ50 K1 ; (P) + 1	$		25
		10	MO			PQ10 MO ; (P) + 1	$		26
		10	MO			PQ10 MO ; (P) + 1	$		27
		00	J111			PQ00 J111; (P) + 1	$		28
		00	AO			PQ00 AO ; (P) + 1	$		29
		00	J125 J8			PQ00 J125; J8	$		30
5		01				NAME DATA	$	BEGIN DATA	31
	K1	01	1	C	K1	PQ01D 0;1	$	THE "D" INDICATES DATA.	32
	MO	01	1	C	MO	PQ01D 0;1	$	THE "0;1" STANDS FOR "+1."	33
	NO	01	1	C	NO	PQ01D 0;1	$		34
5			EO			END START	$	TRANSFER CARD	35

FIG. 1. IPL-V program and TALL program for Ackermann's function

For example the IPL-V data word

P Q SYMB LINK

is represented in TALL in the following manner:

L COMMAND ADDRESS

PQOOD SYMB; LINK

This line of TAC code is expanded by the macro PQOOD into a word which has SYMB in its left-hand address and LINK in its right-hand address. The other IPL-V data terms, decimal, floating point, octal, and alphanumeric, are converted into appropriate TAC constants by macros. The conversion from IPL-V data to TALL data is also a straightforward procedure (see Appendix).

Conclusion

The flexibility of currently available symbolic assembly programs allows the rapid development and modification of advanced computer languages. When these assembly languages are fully utilized, development costs of higher order languages are greatly reduced. In the particular case described here, a list processing language, IPL-V was implemented on the Philco 2000 as a set of macro-operations, subroutines, and conventions within TAC, the assembly program for the 2000. The implementation was accomplished rapidly and also enables the user to combine TAC, FORTRAN, and JOVIAL statements with IPL-V statements. An interpretive version of IPL-V, coded in JOVIAL, has also been implemented on the 2000, see page 479 of this issue. These two methods of implementing IPL-V are currently being studied. The results of this study will be presented in the near future.

ACKNOWLEDGMENTS

I am grateful to C. J. Mossman, S. S. Shaffer, D. P. Haggerty, W. E. Meyer, M. N. Kostiner, H. L. Quon, P. Teas and J. Marx for their assistance in implementing TALL.

APPENDIX

In order to demonstrate the conversion of IPL-V code to TALL code the IPL-V program for computing Ackermann's Function[5] has been translated into TALL. The IPL-V code and the TALL code are presented in Figure 1. The discrepancies between IPL-V and TALL not previously mentioned are noted in the "Comments" column.

[5] NEWELL, A. ET AL. *Op. cit.*, p. 42.