CGTM # 59
J. Ehrman
Jan. 1969

A Study of Attribute Notations in the IBM OS/360 Assembler Language

One of the more useful features of the Assembler Language provided by IBM for the System/360 series is the concept of an "attribute" of a macro operand. There are six attributes in all (ref. 1):

K (Count) — the number of characters forming the actual parameter in the macro call;

N (Number) — the number of operands either in the entire macro call operand list, or the number of operands in a sublist of a single operand;

L (Length) — the length attribute of a symbol which names an area of memory, this is the only attribute that can be used both at macro time and in normal assembly-time expressions;

S (Scale) — the scale attribute of a symbol naming an area of memory;

I (Integer) — the integer attribute of a symbol naming an area of memory;

T (Type) — the type attribute of an operand is used to determine what area of memory it names (if a symbol), whether the operand may be evaluated (as for self-defining terms), or other information of interest. It is the only attribute that is defined for all operands of a macro-instruction.

These attributes are of varying degrees of usefulness in writing macros. In order to obtain some reliable data on their utility, and therefore the frequency with which each is used, a program was written which scans macro definitions and counts the number of occurrences of each type of notation. The program was run twice, once using the standard run-time macro library SYS1.MACLIB, and again with the macro library used for OS Release 16 system generation, SYS1.GENLIB. The results are summarized in Table I.

Table I.

Frequency of Occurrence of Attribute Notations

Library	No. macros scanned	No. occurrences of attribute					
		T	L	K	N	S	I
MACLIB	282	517	12	328	144	0	0
GENLIB	264	212	0	126	152	0	0

The most interesting of the attributes is the Type attribute, so a further and more detailed study was made of the statements in which it was used. These results are summarized in table II.

Table II.

Frequency of Occurrence of Certain Type Attributes

Library	Usage	SETC*	O	N	U	M	T	Others
MACLIB	explicit	10	369	119	11	4	4	0
	implicit*		6	8	6	6	6	2 each*
GENLIB	explicit	0	171	41	0	0	0	0

* see discussion below

Before discussing these results, we may note the following features of the attributes: the Count and Number attributes of an operand are simply requirements of the gross structure of the macro language. The Count attribute is usually useful and is easily provided, being the number of characters in the operand, and the Number attribute is necessary to determine when the end of a list of operands has been reached. Thus both are best understood as normal features of a macro language, and not an important application of the attribute concept.

On the other hand, the Type attribute allows the programmer to ask questions of a much more detiled and informative nature about an operand. In particular, it can be seen from Table II that the most common uses are to ask (1) if the operand is a null string, or (2) if the operand may be used as a numeric quantity. (We will see shortly that the other cases are essentially trivial.) Unfortunately, the first of these questions is better asked by using a literal string comparison: for example, if &DUMMY is a symbolic parameter,

```
AIF    (T'&DUMMY  EQ  'Ø').NULL
```

requires more effort on the part of the Assembler than

```
AIF    ('&DUMMY'  EQ  '').NULL
```

since the latter need merely compare string lengths.

Under the heading "SETC " in Table II are collected those cases in which the type attribute of a symbolic parameter was assigned to a local character (LCLC) variable symbol. These were examined in detail, and the attribute tests to which they eventually led are given in the line labeled "implicit". Of those uses, all except 5 occurrences of the 'U' attribute were in macros (such as for TESTRAN) which were trying to determine if the area of memory named by the symbol given as the actual parameter was available for dumping or printing. Similarly, the occurrences under the heading "Other" were in macros (TEST and PRINTOUT (ref. 2)) which try to determine an appropriate printing format for variables to be dumped at execution time. Of the remaining 5 cases of the 'U' type attribute, all were used to determine whether a symbol should be assumed to be self-defining or not. (This is necessary because the symbol XX defined in the statement

```
XX   EQU   123
```

is treated as having type attribute 'U' by the macro passes.)

It can thus be said that the useful features of the currently-implemented attribute concept are (1) the Length attribute , and (2) the Type attribute, when used to inquire whether an operand has type 'N'. This latter is equivalent to the question of whether the operand may safely be converted from its character-string form to a numeric quantity that can be used for counting and similar internal arithmetic. Thus it is probably fair to say that the type attribute would better have been left out of the language, and that its useful features replaced by simpler constructions: for example, the question of convertability would be well determined by a Boolean attribute, so that the statement

```
AIF    (T'&DUMMY   EQ   'N').CONVERT
```

could have been written

```
AIF    (V'&DUMMY).CONVERT
```

where we assume that the "V" attribute of an operand is true if it has a self-defining value.

In summary:

1. The Integer and Scale attributes are worthless.

2. The Type attribute is badly designed. Its one really useful application (for 'N') is crucial to the macro language, but it could have been implemented much more cleanly. The other uses of the Type attribute are essentially trivial, and making provision for them is not worth the cost.

3. The Length attribute, though little used, is well designed.

4. The Number and Count attributes are valuable. It is ironic, however, that much of the use of the Count attribute is necessitated by the limitation of character variable symbols (GBLC and LCLC) to eight characters.

References:

1. Operating System/360 Assembler Language, Form C28-6514.
2. SLAC Library Routine C035, "PRINTOUT Macro-Instruction".