End of Tape Markers

The end of a section of tape is indicated by

* * * (*x*)

where *x* is *Z*, *A*, *B*, *C* or *T*.

The marker

* * * *Z* indicates the genuine end of the tape/stack of cards

* * * *A* indicates "abandon previous incomplete section, if any" (this may be required by a machine operator)

* * * *B* indicates that a binary tape follows

* * * *C* indicates the end of a section, and that there is another section following on the same tape

* * * *T* indicates a temporary stop within a section.

The number of characters, *n*, on a binary tape may be indicated by

(*n*) * * * *B*

where *n* is a decimal number.

On reading the marker * * * *Z* the peripheral equipment is disengaged by the computer. When the operator next engages this equipment, a new section (with the appropriate heading and title) is read. The marker * * * *C* indicates the end of a section of tape, but the equipment is not disengaged and the next section is automatically read.

On reading the marker * * * *T* for a temporary stop, the equipment is disengaged as for * * * *Z*. However, when the operator next engages this equipment, a continuation of the current section (without a new heading) is read. Finally, on reading the marker * * * *B*, the computer reads the information following, in binary, without testing for further end-of-tape markers.

A better method of specifying the continuation of a section of data, without use of the marker * * * *T*, is by means of a modified "data" heading

DATA/(*n*)

where *n* is the number of the continuation of the section of data; e.g. for a program with data on two distinct paper tapes, the data may be headed

DATA/1
(the title of the data)

and

DATA/2 END
(the same title)

and each tape ends with the marker * * * *Z*. The continuation data tapes may be read into the computer in any order.

Acknowledgements

This work forms part of the Atlas project. It has benefited from many helpful discussions with the authors' colleagues at Manchester University and Ferranti Ltd., whose permission to publish is acknowledged.

Correspondence

To the Editor,
The Computer Journal.

Dear Sir,

"The LISP programming System"

From the paper by Woodward and Jenkins (1961) it appears that McCarthy's LISP system is aimed at bridging the gap between machine instructions and logical propositions in a way which on the machine side is dominated by questions of store utilization and access. It is widely agreed that any "generalized autocode" should have a recursive structure, and there are great advantages in a two-level system of direct language and meta-language. But as soon as one looks at the specific structure of LISP one notices that in the simplest presentation the hardware of the machine store must be such that two parts of each location are separately addressable. The subdivision of computer "words" into "syllables," which is becoming common in new machines, provides this facility; but it must be remembered that this adds to the complexity of the addressing system and so involves some additional cost. It is also essential that the "atoms" be small enough to pack two to a word, though in non-mathematical work the "atoms" may often be addresses of the storage of multi-word blocks of data (e.g. in sorting with detached keys) and two addresses per word is reasonable. Alternatively, the scheme could presumably be used with any machine code which has a Next Instruction Source address associated with each operand address, i.e. a 1 + 1 address code.

But if we are to be sufficiently mathematically minded to retain the idea of a *function*, I would suggest that it is useful to retain the distinction between "quantities" and "operators" in S-language. (I regard a function as a compound or "molecular" operator for this purpose.) Thus the authors' list X, +, COS, 7, (includes two quantities X and + and three operators, namely "add", "take the cosine of" and the opening bracket which is an organizational operator. In algebra a pair of brackets can be described as an operator meaning "treat the list between the mating pair of brackets as a single molecule." This is still true in the comma-and-bracket notation for LISP but cannot reasonably be applied to the dot notation, when a list (C, D, E) which would naturally be regarded as a molecule is represented by (C.(D.(E.Nil))). Looking at the machine format, it is possible to regard opening brackets as the equivalent of

(*Continued on p. 241*)