Published: 11/10/66

Identification

Accessing, Specifiers, and Dope
R. M. Graham and M. D. McIlroy

Purpose

This section describes how the different types of data are accessed.

Non-String Scalars

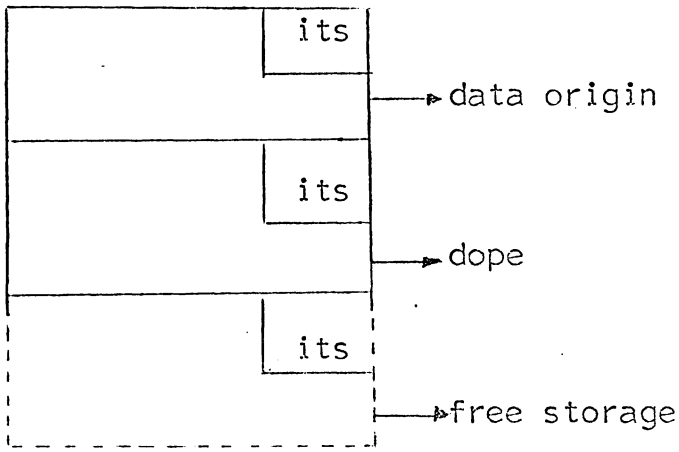
Non-string scalars are accessed directly. When a non-string scalar occurs as a procedure argument, the argument list contains an its pair which points to the first word of the datum.

String and Aggregates

Strings and aggregates are referenced using a specifier and dope. When a string or aggregate occurs as a procedure argument, the argument list contains an its pair which points to the first word of the specifier.

Specifiers

A specifier is address-dependant descriptive material. It consists of two or three its pairs. The first its points to the data origin, which is usually the first word of the data. The second its points to the dope. The third its points to the beginning of a free storage area and is present only in specifiers for varying strings.

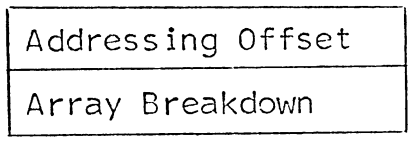


Varying strings are stored in a free storage area and the first its (data origin) points to further information which locates the string(s) in the free storage area.

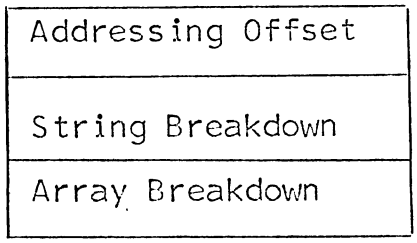
Dope

Dope is address-independent descriptive material. The first word of the dope is always an addressing offset. Following this is one or more breakdowns. The following five cases occur:

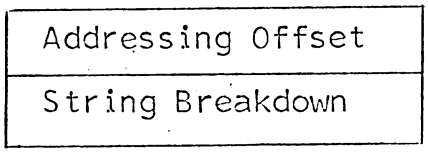
Array of Non-String Scalars



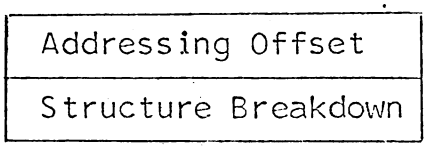
Array of Strings



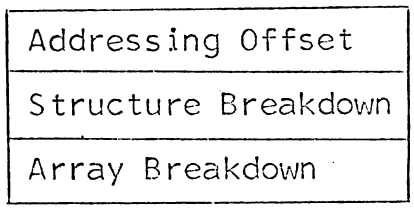
String Scalar



Structure



Array of Structures



Addressing Offset

The data origin (given in the specifier) is defined to be a machine location upon which the data is known to be aligned. For all data except parameters, defined data, and structure elements the data origin is the first word actually occupied by data. The addressing offset is measured in bits and right adjusted in the word if the data is packed or is a non-varying string; otherwise it is measured

in words and is in left half of the word. The addressing offset is the sum of one or more of the following

1) Distance from the data origin to the beginning of a non-varying string. A non-zero offset occurs when the datum is either; a substring which is not at the beginning of the string, or was defined using the "position" attribute with position greater than one.

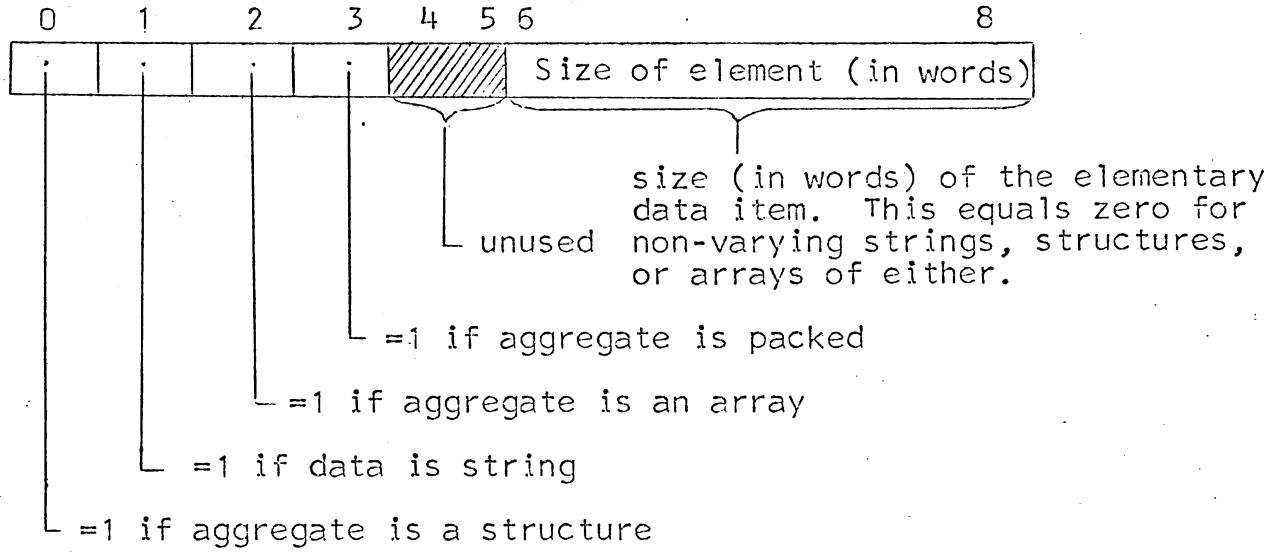
2) Distance from the first actual element of an array to the (perhaps hypothetical) element with subscript (0,...0).

3) The distance from an element of a structure to the beginning of the immediately containing structure.

The offset is maintained modulo the segment size, 2**18.

Identity Code

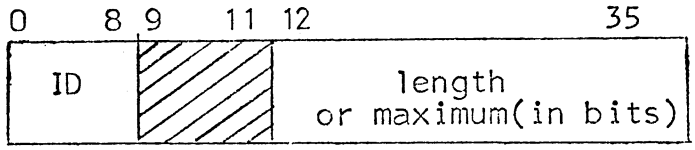
A nine bit identity code appears in the first word of array, string, and structure breakdowns as well as in the substructure pointer words. The bits of this code have the following meanings:



The identity code in a string or structure breakdown and an associated array breakdown are identical.

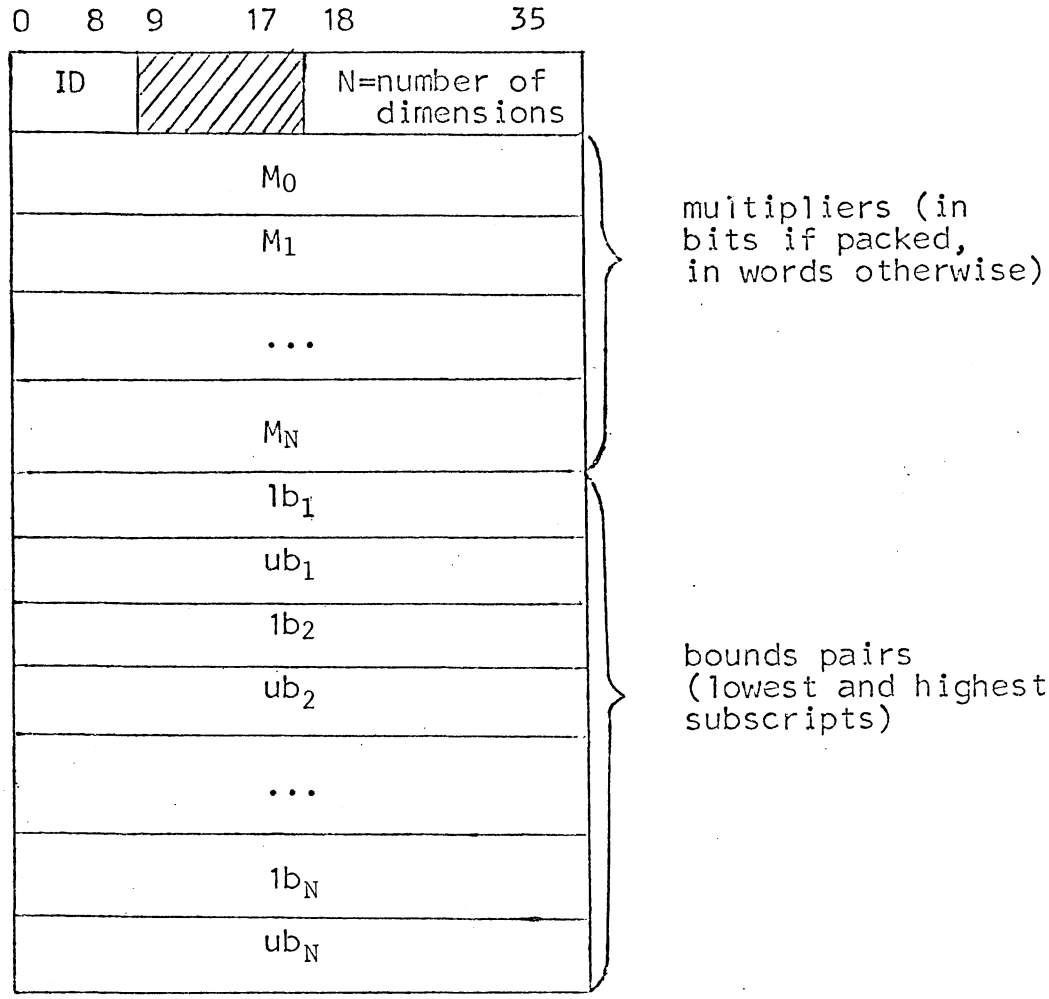
String Breakdown

A string breakdown is a single word. The first nine bits are the identity code. The remainder of the word contains the current length for a non-varying or the maximum length for a varying string. Both the length and maximum are expressed in bits.



Array Breakdown

The first word of an array breakdown contains the identity code in the first nine bits and the number of dimensions, N, in the right half.



Following the first word are N+1 words containing the multipliers. Following the multipliers are 2*N words containing the N bounds pairs. The bounds pair for each dimension occupies two words. The first word contains the lower subscript bound and the second word contains the upper subscript bound for that dimension. The multipliers read in words if the array is aligned and in bits if it is packed. M_0 is the smallest amount of contiguous space which will contain this array. M_i is the separation between two elements whose subscripts differ only in the i^{th} position by exactly one. The machine address of any element is determined by the following formulae.

$$\text{loc}\ [A(s_1,\ldots,s_n)] = \text{loc}[A(0,\ldots 0)] + \sum_{i=1}^{N} s_i\ M_i$$

$$= \left[\ \text{data origin} + \text{addressing offset} + \sum_{i=1}^{N} s_i\ M_i\ \right] \text{ modulo } 2^{**}18$$

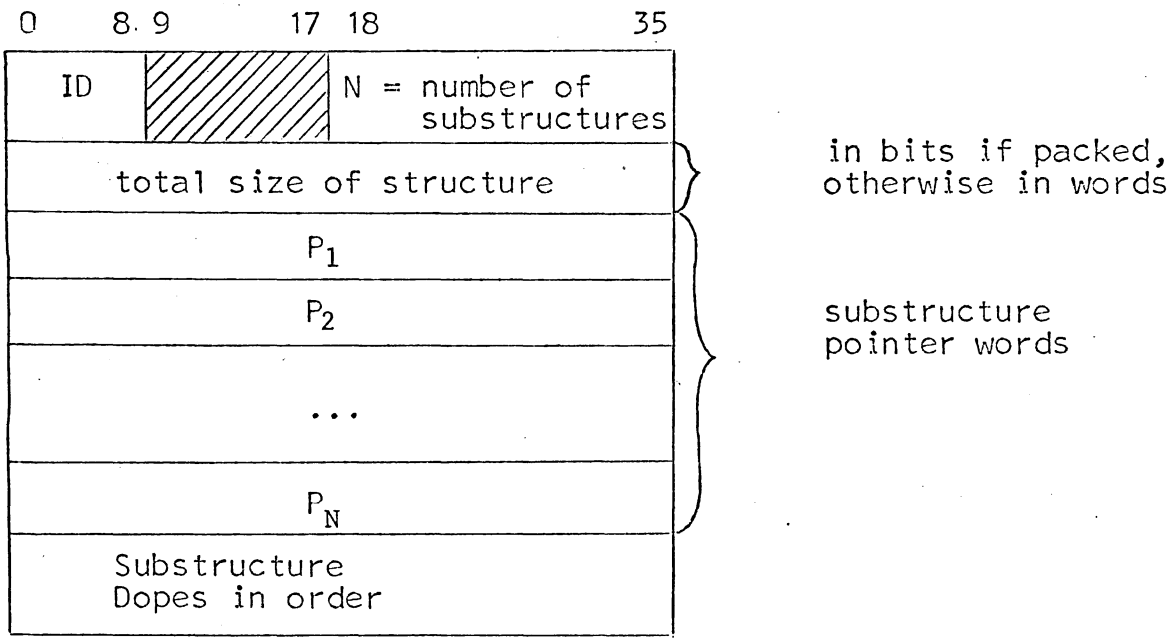
where the sum $\sum_{i=1}^{N} s_i\ M_i$ is modulo 2**18 if the M_i read in words otherwise it is modulo 36*2**18.

Note that M_N is generally equal to the size of elementary element given in the identity code. However in some cases (e.g., cross sections) it is greater. For arrays whose elements are contiguous, the M_i are calculated according to

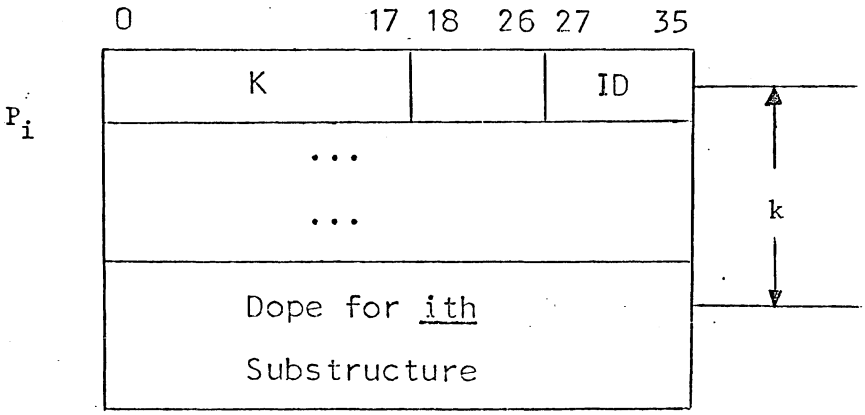
$$m_{i-1} = (ub_i - lb_i + 1) * M_i$$

Structure Breakdown

The first word of a structure breakdown contains the identity code in the first nine bits and the number of substructures (on the next inferior level) in the right half.



Following the first two words are N substructure pointer words. Following the substructure pointers is the dope for all substructures which require it. A substructure pointer is one word containing the identity code for the structure in the last nine bits. The left half of the substructure pointer word contains one of two quantities depending on whether the substructure has dope or not. If the substructure has dope (i.e., it is a string or aggregate), the left half contains a self-relative pointer to the location of this dope.



If the substructure does not have dope, the left half contains the location, relative to the beginning of the immediately (superior) containing structure, of this sub-structure.