INTERNAL USE ONLY

INTERNATIONAL COMPUTERS AND TABULATERS LIMITED

Programming Languages Division

FORTRAN NOTE 25

8.12.66.

Array addressing and communication

This note replaces Fortran Note 9 which should be marked as obsolete or destroyed.

Array Addressing

1. Calculation of the address of an array element

Associated with each array is an "array header" which contains information about the structure of the array. The array header is not stored adjacent to the array itself; array headers are stored in upper preset area, whereas arrays are held in upper variable area (or upper preset if reference is made to them in a DATA statement).

For arrays with a single subscript the address of the specified array element is calculated by code generated by the compiler. For example, if A(I) is encountered and A is a real array then the code generated would be:

```
LDX   6   I
SLL   6   1
LDX   3   Address of array header
LDX   3   0(3)
ADX   3   6
```

The second instruction becomes 'SLL 6 2' for double precision or complex arrays and is omitted for integer or logical arrays compiled in 'Compress Integer and Logical' mode. When the subscript is an expression then 'LDX 6 I' is replaced by a series of instructions to calculate, in X6, the subscript value. For arrays with more than one subscript, the address of the array element is calculated by entering one of several subroutines contained within the Fortran arithmetic package, %FAP4. Three separate routines exist in order to minimise execution time and their entry points are;-

%FAP4 + 25 for arrays with two subscripts and 1 or 2 words per element (i.e. not complex or double precision elements).

%FAP4 + 20 for arrays with two or more subscripts and 4 words per element.

%FAP4 + 6 for arrays with two or more subscripts and 1 or 2 words per element.

On entry to any one of these subroutines

X3 = address of the array header

X1 = link

and the CALL must be followed by one instruction per subscript which, if OBEYed, will place the address of the subscript in X3.

On exit, the address of the desired array element is in bits 6 - 23 of X3 and the top 6 bits will be non zero. The calling routine will be re-entered at the location following the last subscript instruction.

If the object program segment is being compiled in 'TRACE' mode then all array element calculation will be carried out by a subroutine contained within the Fortran trace package %FMN4 with the entry points.

%FARRY + 0 for arrays with 1 or 2 word elements

%FARRY + 2 for arrays with 4 word elements

This subroutine requires the same parameters as the %FAP4 subroutines and gives the same result but in addition checks that the array element address is within the bounds of the array.

The Basic.Fortran compilers generate a call to either %FAP4 + 6 or %FAP4 + 20, depending on the type of element, for array elements with one or more subscripts (unless in TRACE mode when the call is to %FARRY or %FARRY+2).

2. The Array Header Format (See Footnote *)

The array header format is compatible with that used by Algol. It is made up as follows:

1st word Sign bit = 0 for 1 or 4 word elements
= 1 for 2 word elements
Next 5 bits = No.of dimensions in array
Next 15 bits = Base address (address of element 0,0,---0)

2nd word Sign bit = 1 for 4 word elements
Next 5 bits = Zero
Next 18 bits = Address of first element - always element 1,1,1,-----in Fortran, but may be dirrerent for Algol.

3rd word Sign bit = 0
Next 5 bits = 0
Next 18 bits = Address of first word past end of array.

This is the array header for a one-dimensional array. For an array of more than one dimension succeeding words give the partial products of dimensions as follows:

4th word = D_1 ⩾ 2 dimensions
5th word = D_1 D_2 ⩾ 3 dimensions

---

* The array header format is likely to be changed in the near future; details of the proposed change are given in the appendix.

last word = $D_1 . D_2$ ---- D_{n-1} n dimensions

where D_i is the maximum value of the ith subscript may take.

3. Communication of array information between segments.

The 'calling' segment may specify as an actual argument in a CALL statement either an array name or an array element in order to transmit an array or part of an array to the 'called' segment. In the case of an array name it is the address of the array header that is passed on whereas for an array element it is the address of the array element that is passed on. The two types are distinguishable by inspection of the top 6 bits which are zero in the case of an array header and non-zero for an array element.

It is also possible to transmit an array of Hollerith characters. The argument in the CALL statement takes the form of a TEXT constant (i.e. a string of hollerith characters preceded by nH) and the address of this string of characters is made to look like the address of an element by making the top 6 bits non zero.

The called segment will have as a corresponding argument in a FUNCTION or SUBROUTINE an actual array name. The ASA specification allows, and in fact insists on, the array being re-defined in a DIMENSION or 'type' statement. The subscripts may either be unsigned integers or integer variables in which case they must appear also as arguments of the subroutine or function. Thus the original array defined in the calling segment may be re-partitioned either statically or dynamically, the structure being defined by the values of the subscripts at the time that the subroutine is entered. An array must be given its maximum size in its original definition. Arrays may be repartioned in any manner but the type of actual anddummy array should correspond. It should be realised that an array, originally defined as A(10,10) when redefined as B(5,5) will not correspond to the top left hand corner of the original array but to the first 25 elements of the array taking in column order, i.e. B(5,5)≡ A(5,3).

Note that only arrays appearing as an argument in the FUNCTION or SUBROUTINE statement may have a variable structure.

4. Structuring of Dummy arrays.

The structuring of a dummy array is done by means of the FORTRAN subroutine %FARHD which on entry requires

(i) X3 to contain the address at which %FARHD is to generate a new array header.

(ii) X6 to co ntain the address of a word which contains the address of the original array header, or to contain the address of a word which contains the address of the first element of the array and the top 6 bits are non-zero.

(iii) X1 to contain the link.

%FARHD also requires a parameter list following the CALL instruction where;-

1st word: $b_0 - b_{11}$ = 0

$b_{12} - b_{17}$ = number of words per element (i.e.1,2 or 4)

$b_{18} - b_{23}$ = number of dimensions in new array structure

following words:- a series of instructions (one per dimension) which when OBEYed would place in X3 the address of a word containing that dimension.

On exit from %FARHD, a new array header will have been generated starting at the address originally specified in X3.

The word originally specified by the contents of X6 will now contain the address of the new array header.

L.R. Fairbrother.

Appendix to Fortran Note 25

1. Change of array header format

In the near future object programs will have to work in "Extended Data Mode" where the modifiable part of a word is the least significant 22 bits instead of 15 bits. When the array header format was originally designed allowance was made for extension only to 18 bits and therefore it will be necessary to change the array header format to work under "Extended Data Mode" conditions. Since there are a number of Fortran/PLAN users whose programs are dependant on the format of array headers it will be necessary to make the change in two stages:

(i) Provide subroutines which interrogate and generate array headers so that users may modify their programs to make use of these subroutines.

(ii) Change the array header format and at the same time modify these subroutines so that the user is unaffected.

2. Array header subroutines

Two subroutines are proposed for use in object programs at PLAN level:

a) GETAH to interrogate an array header, the calling sequence being;-

```
CALL    1  GETAH
LDX     3  'Address of array header'
LDX     3  'Address of information block'
```

where the information block consists of n + 4 words to receive details of the array header:

1st word	Number of dimensions (n)
2nd word	Number of words per element
3rd word	Base address
4th word	Address of first element
5th word	Address of first word past end of array
6th word	first partial product (D_1)
7th word	second partial product ($D_1 . D_2$)
(n+4)th word	last partial product ($D_1 . D_2 \cdots D_{n-1}$)

b) GENAH to generate an array header, the calling sequence being:-

```
CALL   1  GENAH
LDX    3  'Address of location at which array
          header is to be generated'
LDX    3  'Address of information block'
```

The information block must be set up as defined previously. On exit (return to link + 2) The array header will be generated at the address specified. If this address is the same as the address of the information block the effect will be to condense the information block into a standard array header. It should be realised that the number of words in an array header is likely to be increased when the array header is changed.

3. New array header format

Word	Bit	Content
0	B0	0 if 1 or 4 word element, 1 if 2 word element
	B1	1 (to distinguish array element from array header)
	B2-B23	Base address, i,e. address of element 0,0,0 -----
1	B0	0 if 1 or 2 word element, 1 if 4 word element
	B1	Undefined
	B2-B23	Address of first element - in Fortran always element 1,1,1,-----
2	B0	1 (to distinguish from old type of header)
	B1	Undefined
	B2-B23	Limit address, i.e. address of first word past end of array.
3	B0	Undefined
	B1	Undefined
	B2-B23	Number of dimensions (maximum 15)
4,5 etc		Partial products as now

The above format should be taken as definitive and any previously proposed new array header format should be ignored.