# CMLFLOATARRAY

CmlFloatArray implements high-speed floating-point vector and array operations. Although optimized for the case of arrays of element-type single-float, the array operations are generic and will operate on arrays of any element-type.

CmlFloatArray uses special purpose microcode that exploits the full capabilities of the Weitek floating-point chip set, available on 1109s, for doing arithmetic operations on floating-point arrays. On machines without the Weitek floating-point chip set, such as the 1186, these operations will still usually be more efficient than the corresponding scalar implementation.

The functions described here operate on Common Lisp arrays, and may be thought of as extensions of the general Common Lisp sequence functions.

## Requirements

1186 or 1109 (1108 with the extended processor option) and the Weitek floating-point chip set.

## Installation

Load CMLFLOATARRAY.LCOM from the library.

## Functions

(MAP-ARRAY *RESULT MAPFN ARRAY1 ARRAY2. . . ARRAYN*) [Function]

MAP-ARRAY is a general mapping function over arrays and scalars.

Arrays of dimension greater than one are treated as vectors in row-major order; that is, an array A with dimension (2 2) is treated as a vector of length 4 and elements '#(,(aref a 0 0), (aref a 0 1), (aref a 1 0), (aref a 1 1)). All array arguments must be conformable; that is, of the same dimensions.

Scalars (non-arrays) are extended to the common dimension of the other array arguments by copy-on (that is, a scalar is treated as a vector of the appropriate length, each of whose elements is the scalar).

For example, a call to MAP-ARRAY with two arrays of dimensions (4 4) and one scalar and the function MAX as map function will invoke MAX 16 times, with the scalar the third argument in each call.

If *RESULT* is NIL, the map is for effect only (i.e., no array result is returned; MAP-ARRAY is of interest only due to a side effect).

If *RESULT* is a valid element-type, an array of the appropriate dimensionality and element-type will be created to hold the map results.

If *RESULT* is an array, it must be conformable with the other array arguments and it will be side effected by the mapping operation.

*MAPFN* is an arbitrary n-ary Lisp function; i.e., it takes as many arguments as there are arrays passed to MAP-ARRAY. It is unary (takes one argument) if one array is passed to MAP-ARRAY, binary if two arrays are passed, etc.. In the case of unary or binary operations, MAP-ARRAY recognizes certain functions and executes the corresponding operation particularly efficiently.

If the single array argument is of element-type single-float and the result array is of element-type single-float, the following unary operations are recognized and executed in microcode:

| | |
|---|---|
| - (MINUS) | Negates each element of the array argument. |
| ABS | Computes the absolute value of each element of the array argument. |
| TRUNCATE | The single array argument must be of element-type single-float, but the result array may be any element-type which will accomodate the integer results. Truncates (converts to integer, rounding towards zero) each element of the array argument. |
| FLOAT | The single array argument must be of element-type (unsigned-byte 16), and the result array may be of element-type single-float.<br><br>Converts each element of the array argument to a single precision floating point number. |

If both arguments are of element-type single-float and the result array is of element-type single-float, the following binary operations are recognized and executed in microcode.

| | |
|---|---|
| + (PLUS) | Computes the element-wise (element by element) sum of the two arguments. |
| - (MINUS) | Computes the element-wise difference of the two arguments. |
| * (TIMES) | Computes the element-wise product of the two arguments. |
| /(QUOTIENT) | Computes the element-wise quotient of the two arguments. |

(REDUCE-ARRAY *REDUCTION-FUNCTION ARRAY &OPTIONAL INITIAL-VALUE*) [Function]

REDUCE-ARRAY is similar to the sequence function REDUCE but is generalized for arrays of arbitrary dimensionality; that is, the binary mapping function is applied to each element of the single array argument, each time being passed the result of the previous application as well as the current array element. Arrays of dimensionality greater than one are treated as vectors in row-major order. The result of REDUCE-ARRAY is always a scalar.

If *INITIAL-VALUE* is provided, it is used as the starting value of the reduction operation, otherwise the first element of *ARRAY* is the starting value. In the degenerate case of arrays of size zero or one, the use of *INITIAL-VALUE* parallels that of the sequence function REDUCE. REDUCE-ARRAY recognizes certain mapping functions and executes the corresponding operation particularly effeciently.

If the single array argument is of element-type single-float, the following reduction operations are recognized and Executed in microcode.

| | |
|---|---|
| + (PLUS) | Computes the sum of all the array elements |
| * (TIMES) | Computes the product of all the array elements |
| MIN | Returns the smallest array element |
| MAX | Returns the largest array element |
| MIN-ABS | Returns the smallest array element in absolute value |
| MAX-ABS | Returns the largest array element in absolute value |

(EVALUATE-POLYNOMIAL *X COEFFICIENTS*) [Function]

This function calculates the value of a polynomial at the point X. The polynomial is described by a vector of coefficients, *COEFFICIENTS*, where *COEFFICIENTS[0]* corresponds to the coefficent of highest degree. If *COEFFICIENTS* is a vector of element-type single-float, then this operation is Executed in microcode.

(FIND-ARRAY-ELEMENT-INDEX *ELEMENT ARRAY*) [Function]

Returns the index of the first element of *ARRAY* that is EQL to *ELEMENT*, or NIL if there is no such element.

## Limitations

This version of CmlFloatArray does not support the FFT functionality of previous versions.

[This page intentionally left blank]