What will be in Fortran 2008

John Reid,

Convener ISO Fortran Working Group

Abstract

Following completion of the Fortran 2003 standard, WG5 decided that the next revision would be minor and come out five years later.

A preliminary choice of features (for the first draft) was made in 2005 and the final choice in 2006.

This talks aims to give an overview of these features.

Fifty Years of Fortran
BCS, 25 January, 2007

Repository

`ftp://ftp.nag.co.uk/sc22wg5/N1601-N1650/N1649.txt`

Large items described on later slides

- UK-01 Co-arrays for parallel programming
- J3-47 BITS
- J3-14 Intelligent macros

Medium items for enhanced performance

- J3-43 Contiguous attribute
- J3-46 `DO CONCURRENT`

  Iterations of the loop are independent

Item to be developed as a TR

- J3-41/2 Interoperability of pointers, allocatables, assumed-shape arrays, and optional arguments

Minor technical changes

- J3-03 Execute external program
- J3-13 Internal procedure as actual argument
- J3-18 Non-null initial targets for pointers
- J3-19 Extend intrinsics such as `ASIN` to complex arguments
- J3-38 Libm: Bessel, erf, gamma, hypot
- J3-39 Rank plus co-rank limited to 15.
- RU-03 Obsolescent: ENTRY
- UK-05 Guarantee support of `selected_int_kind(18)`
- UK-11 Elemental procedures that are not pure

Co-arrays

- SPMD – Single Program, Multiple Data
- Replicated to a number of **images**
- Number of images fixed during execution
- Each image has its own set of variables
- Images mostly execute asynchronously
- Co-arrays have second set of subscripts in [ ] for access between images
- Synchronization: `sync all`, `sync team`, `notify`, `query`, `allocate`, critical construct
- Collectives: `co_all`, `co_any`, ...
- Intrinsics: `this_image`, `num_images`

Full summary:

`ftp://ftp.nag.co.uk/sc22wg5/N1651-N1700/N1669.pdf`

Example

```
real :: p[*]
if (this_image()==1) then
  read(*,*) p
  sync all
else
  sync all
  p = p[1]
end if
```

Implementation model

The compiler may arrange that a co-array occupies the same set of addresses within each image.

Optimization

Between synchronizations, the compiler can optimize as if the image is on its own, using its temporary storage such as cache, registers, etc.

BITS

There will be a new intrinsic type, `BITS`. The number of bits is specified by the kind type parameter with default `NUMERIC_STORAGE_SIZE`.

Up to `4*NUMERIC_STORAGE_SIZE` bits must be supported. Processor may support more.

Concatenation operator `//` available.

`==`, `/=` available for bits with bits, real, integer, or complex.

`>`, `>=`, `<`, `<=`, available for bits with bits, real, or integer.

`.AND.`, `.OR.`, `.XOR.`, `.EQV.`, `.NEQV.` available for bits with bits or integer.

`.NOT.` available for bits.

If the kinds differ, the shorter is padded on the left with zeros.

Assignment to bits

Assignment to bits available from bits, real, integer, or complex. If the kinds differ, digits on the left are discarded or padded with zeros. For types other than bits, the internal representation is used.

Interoperability

There are 26 C types that are interoperable with bits.

New intrinsics

Lots, including:

```
BITS(A [,KIND])          Conversion to bits type
MERGE_BITS (I,J,MASK)    Merge bits under mask
SHIFTL (I, SHIFT)        Left shift
LEADZ (I [,KIND])        Number of leading zero bits
POPCNT (I [,KIND])       Number of one bits
IALL(ARRAY,DIM[,MASK]) Bitwise AND of array elements
or IALL(ARRAY[,MASK])
```

Intelligent macros

‘Intelligent’ macros know about Fortran and are scoped. Can create modules, types, procedures, and sections of code.

Example of macro definition:

```
DEFINE MACRO :: single_linked_list(type)
  TYPE type%%_list
    type :: value
    TYPE(type%%_list),POINTER :: next
  END TYPE
END MACRO
```

and later macro expansion:

```
EXPAND single_linked_list(real)
```

where the EXPAND statement is replaced by the sequence of statements

```
  TYPE real_list
    real :: value
    TYPE(real_list),POINTER :: next
  END TYPE
```

The %% are needed for token concatenation.