CRAY The Supercomputer Company

Fortran 2003 and Beyond

Bill Long, Cray Inc.

17-May-2005

Fortran 2003 and Fortran 2008

- Fortran is now ISO/IEC 1539:1-2004
- Common name is Fortran 2003 or f03
- This “cancels and replaces” the previous standard, f95.
- Official date is 15-Nov-2004.

- Same content is available at j3-fortran.org
- Document number is 04-007.
- Also available on-line internally at Cray.
- formats: postscript, pdf, ASCII text

- Process for Fortran 2008 has begun.
- Preliminary content selection at WG5 meeting, May 9-13, 2005.

Organization

- Overview of selected f03 features
- Implementation status in the X1/BW ftn compiler
  - status does not apply to the MTA compiler
  - status does not apply to the PGI Opteron compiler
- Selected f08 features

New f03 features

- Allocatable components and dummy arguments
- C interoperability
- Procedure pointers
- IEEE arithmetic support
- Asynchronous and Stream I/O
- Parameterized derived types
- Object Oriented Programming
- ISO character support

- Better environment support
- I/O statement keywords
- Longer names, more continuation lines

Allocatable components, dummy args

- Structure components can be allocatable
- Dummy arguments can be allocatable
- Function results can be allocatable

- Pointers were allowed in old versions
- This is a logical extension that is often more efficient.

- Implemented long ago.

C Interoperability

- Standard syntax for interoperating with a companion C processor.
  Access by use,intrinsic :: iso_c_binding

Function calls:

BIND(C) attribute on Fortran interface

BIND(C, name='…') attribute

VALUE attribute for dummy arguments

Global data:

BIND(C) attribute for module variables

Implemented

Procedure pointers and declarations

- Evolution of dummy procedure arguments

- procedure(interf),pointer :: pname
- procedure(interf) :: gname
- pname => gname

- procedure(real) :: fname

- Abstract interfaces

- Procedure pointers can be structure components

- Implemented in cftn 5.4

IEEE arithmetic support - overview

- Many types and routines defined in intrinsic modules.
  Details in Section 14 of the standard.

  - control floating point exception handling

  - set hardware rounding modes

  - inquire about data values (ieee_is_nan(x))

  - inquire about support (we do not support denormals)

  - scheduled for cftn 5.5

IEEE floating point exceptions

- Overflow (huge(x) * 1000.)
- Underflow (tiny(x) * 0.0001)
- Divide by Zero (1./0.)
- Invalid (0./0. or NaN - NaN)
- Inexact (1./3.)

The X1 hardware flushes underflows to 0.

IEEE floating point control register

- Has three sets of flag bits:

  - Is recording of an exception enabled?

  - Does a trap on an exception occur?

  - Specify the rounding mode

    Round to nearest (the default)
    Round up
    Round down
    Round towards zero

IEEE floating point status register

- Has two sets of flag bits:

  - Did an exception occur (signal)? (only if recording enabled)

  - Trap bits that trigger an exception fault (only if trap enabled)

  - can force an interrupt by setting a trap bit

  - Accessing the status register includes a fp sync instruction

IEEE Get and Set routines

- Can get and set the overall status - used for save/restore

```
use,intrinsic :: ieee_exceptions
type(ieee_status_type) :: S

call ieee_get_status(S)
call ieee_set_status(S)
```

IEEE - predefined status modes

- Three predefined modes are added as a Cray extension:

  All three modes clear the status register and set round to nearest

  ieee_cri_silent_mode : set control to no record, no trap

  ieee_cri_nostop_mode: set control to record, no trap

  ieee_cri_default_mode: record and trap overflow,
  divide by zero,
  invalid
  no record or trap underflow, inexact

New I/O capabilities

- IOMSG keyword - returns message text (in cftn 5.4)
- Additional keywords on open, read, write, inquire statements
- Asynchronous I/O (planned for cftn 5.5)
- Stream I/O (planned for cftn 5.5)

Parameterized derived types

- Extension of KIND system for intrinsic types to derived types
- Enables reusable type definitions

```
type  grid(k,n)
   integer,kind :: k
   integer,len  :: n
   real(k),dimension(n,n) :: cell
end type

type(grid(8,1000)) :: X
```

parameters can be deferred if object is allocatable

scheduled for cftn 5.5

Extended types - Object oriented programming

- Simple type extension - single inheritance
- Type bound procedures, may be overridden on extension
- Polymorphic dummy arguments and allocatable objects
- Planned for after cftn 5.5

ISO character support

- Support for ISO 10646 character set (32-bit characters)
- Planned for after cftn 5.5

Fortran 2008 - process

- WG5 is ISO body - sets policy and feature list
- J3 is implementation body - writes standard document

- WG5 meeting last week considered proposals for Fortran 2008
  Final stamp will come at the February, 2006 meeting

  Group A: Want in the standard -> required

  Group B: Would like in the standard -> approved

Final status determined by progress made in J3 by next February.

Fortran 2008 Group A features

- Co-Arrays
- Submodules
- Concurrent construct
- Contiguous attribute
- intent(scratch) attribute
- internal procedures as actual arguments and pointer targets
- require 64-bit integer support
- standard form of ‘call system()’
- rank up to 15
- additional math intrinsics
- enhanced STOP statement
- decimal arithmetic
- empty contains / type(intrinsic-type)

Fortran 2008 Group B features

- Parameterized modules (generic programming)
- BITS data type (was ‘typeless’ data)
- mechanisms for C interoperability with allocatable, pointer, optional
- EXIT from additional labeled constructs
- non-null pointer initialization
- additional intrinsics from libm and HPF
- CSV file support
- parameter size from initialization expressions
- generic resolution based on pointer/allocatable
- generic resolution based on procedure/variable
- update section 14 for IEEE 754R standard
- define parts of complex variable separately
- IO_UNIT data type

Parameterized Modules

- New class of modules with arguments that are constants or type-specs
- Instantiate an actual module by supplying actual values
- Similar to templates or a built in macro facility
- Aimed at more reusable code
- Based on the model in ADA
- Logical completion of the OOPS facility in Fortran 2003.

Last Slide!

Thanks to the Fortran compiler group at Cray

Contact info:

Bill Long
longb@cray.com

J3 web site:

www.j3-fortran.org