Support for F2x feature emulation in F95

Aleksandar Donev

June 25, 2002

1 Module *F2x_Emulation*

In trying to emulate some of the desperately needed F2x features, I rely heavily on FWEB macros. However, some runtime support is also needed in order to make life easier. Bits of code needed for this purpose are given here, while the macros are in the always included *Fortran.F2x.hweb*.

In particular, procedure pointers are supported via a new derived datatype, and also array pointer reshaping in pointer assignment is supported. For now, for simplicity, I only support array-pointer bounds-remapping for contiguous pointers–calling these routines with non-contiguous data-target will produce *wrong* results! With just a bit more work strides can easily be supported: One needs a mechanism to find the actual stride and then use these when assigning the array pointer to the properly scaled assumed-size argument in the *MakeArrayPointer* functions. However, since the utility of this is not clear to me, I will skip it for now...

"WEAVE.f90" 1 ≡

```
MODULE F2x_Emulation
   USE Precision
   USE ISO_C_BINDING
   IMPLICIT NONE
PRIVATE
   PUBLIC :: AssignArrayPointer, ReshapeArrayPointer      // Pointer reshaping in assignment
   _TYPE, PUBLIC :: Procedure_Pointer
      _C_PTR :: c_pointer = C_NULL_PTR
      _F_PROC_PTR, POINTER :: f_pointer => NULL()
   ENDTYPE
   ⟨ AssignArrayPointer_Generic 1.2.1 ⟩
   ⟨ ReshapeArrayPointer_Generic 1.1.1 ⟩
CONTAINS
   ⟨ ReshapeArrayPointer_Implementation 1.1.2 ⟩
   ⟨ AssignArrayPointer_Implementation 1.2.2 ⟩
END MODULE F2x_Emulation
```

1.1 Array pointer bounds remapping

"WEAVE.f90" 1.1 ≡

```
@m _ReshapeArrayPointer_Body(type_, kind_, rank_)

    SUBROUTINE ReshapeArrayPointer_@&kind_@&rank_(array_pointer, array_target, lower_bounds,
          upper_bounds)
      INTEGER (KIND = i_wp), DIMENSION (rank_), INTENT (IN) :: lower_bounds, upper_bounds
      type_(KIND = kind_), DIMENSION (_FullExtent(rank_)), POINTER :: array_pointer
      type_(KIND = kind_), DIMENSION (:), INTENT (IN), TARGET :: array_target
            // This can be an array section i.e. non-contiguous

      _C_PTR :: base_address
      INTEGER (KIND = i_wp), DIMENSION (rank_) :: pointer_shape
      INTEGER (KIND = C_INT), DIMENSION (rank_) :: strides

      pointer_shape = upper_bounds − lower_bounds + 1
      IF (_SIZE(array_target, i_wp) < PRODUCT(pointer_shape)) THEN
            // This is an error, so return a null pointer
        NULLIFY (array_pointer)
        RETURN
      END IF

      strides = 1
      IF (_SIZE(array_target, i_wp) > 1) THEN
        strides_1 = C_STRIDE(array_target_1, array_target_2)     // Find the actual stride using C
      END IF
      pointer_shape *= strides     // Allow for the extra size needed for the stride

      base_address = C_LOC(array_target)
      CALL C_F_POINTER(CPTR = base_address, FPTR = array_pointer, SHAPE = pointer_shape)

        // Now we need to change the bounds and strides accordingly
      CALL AssignArrayPointer(array_pointer = array_pointer,
          array_target = array_pointer(_STRExtent(rank_, strides)), lower_bounds = lower_bounds)

    END SUBROUTINE

@m _ReshapeArrayPointer_Body_TKR(tk_number_, rank_, ...)
    _ReshapeArrayPointer_Body(_AllIntrinsicTypes(tk_number_), _AllIntrinsicKinds(tk_number_), rank_)
```

⟨ ReshapeArrayPointer_Generic 1.1.1 ⟩ ≡

```
_GenerateInterface_TKR_Array(N_TK_ALL, ReshapeArrayPointer)
```

This code is used in section 1.

⟨ ReshapeArrayPointer_Implementation 1.1.2 ⟩ ≡

```
_GenerateBody_TKR_Array(N_TK_ALL, _ReshapeArrayPointer_Body_TKR)
```

This code is used in section 1.

1.2 Array pointer assignment with explicit lower bounds

"WEAVE.f90" 1.2 ≡

@m _AssignArrayPointer_Body(*type_*, *kind_*, *rank_*)

```
SUBROUTINE AssignArrayPointer_@&kind_@&rank_(array_pointer, array_target, lower_bounds)
  INTEGER (KIND = i_wp), DIMENSION (rank_), INTENT (IN) :: lower_bounds
  type_(KIND = kind_), DIMENSION (_FullExtent(rank_)), POINTER :: array_pointer
  type_(KIND = kind_), DIMENSION (_LBExtent(rank_, lower_bounds)), INTENT (IN),
        TARGET :: array_target

  array_pointer => array_target

END SUBROUTINE
```

@m _AssignArrayPointer_Body_TKR(*tk_number_*, *rank_*, ...)
_AssignArrayPointer_Body(*_AllIntrinsicTypes*(*tk_number_*), *_AllIntrinsicKinds*(*tk_number_*), *rank_*)

⟨ AssignArrayPointer_Generic 1.2.1 ⟩ ≡

_GenerateInterface_TKR_Array(*N_TK_ALL*, *AssignArrayPointer*)

This code is used in section 1.

⟨ AssignArrayPointer_Implementation 1.2.2 ⟩ ≡

_GenerateBody_TKR_Array(*N_TK_ALL*, _AssignArrayPointer_Body_TKR)

This code is used in section 1.

§. `@I "C_Interop.hweb"` *Section(s) skipped...*
§. `@I "Macros.hweb"` *Section(s) skipped...*

§. `@I "Fortran.formatting.hweb"` *Section(s) skipped...*

§. `@I "Fortran.macros.hweb"` *Section(s) skipped...*

§. `@I "Fortran.F2x.hweb"` *Section(s) skipped...*