# Enhanced debuggability support in LLVM for various Fortran language features

FOSDEM'22

AMD

Alok Kumar Sharma
Bhuvanendra Kumar N

# Agenda

- Array debugging
- Fortran String debugging
- Module debugging
- Fortran Namelist debugging

AMD

# Array debugging: What we had…

***Simple C array***

Defined as “arr[10]”, where count=10, lowerBound=0, upperBound=9

## Metadata / DWARF

- DICompositeType(tag: DW_TAG_array_type
  - Is converted to DW_TAG_array_type
  - Has fields
    - baseType
    - size
    - elements
- DISubrange
  - Is converted to DW_TAG_subrange_type
  - Mandatory field
    - count / DW_AT_count: Constant / Variable
  - Optional field
    - lowerBound / DW_AT_lower_bound: Constant

```
$ cat -n csimple.c
     1  #include<stdio.h>
     2  void fun(void) {
     3    int arr[10];
     4    printf("size of arr = %lu", sizeof(arr));
     5  }
```

```
!12 = !DILocalVariable(name: "arr", scope: !8, file: !1, line: 3, type: !13)
!13 = !DICompositeType(tag: DW_TAG_array_type, baseType: !14, size: 320, elements: !15)
!15 = !{!16}
!16 = !DISubrange(count: 10)
```

```
0x0000003f:      DW_TAG_variable
                   DW_AT_location        (DW_OP_fbreg -48)
                   DW_AT_name    ("arr")
                   DW_AT_decl_file       ("/home/alok/work/array/csimple.c")
                   DW_AT_decl_line       (3)
                   DW_AT_type    (0x0000004e "int[10]")

0x0000004d:      NULL

0x0000004e:    DW_TAG_array_type
                 DW_AT_type      (0x0000005a "int")

0x00000053:      DW_TAG_subrange_type
                   DW_AT_type    (0x00000061 "__ARRAY_SIZE_TYPE__")
                   DW_AT_count   (0x0a)

0x00000059:      NULL
```

AMD

# Array debugging: What we had…

***C/C++ Variable length arrays***

## Metadata

- DISubrange
  - Is converted to DW_TAG_subrange_type
  - Mandatory field
    - count / DW_AT_count: Constant / **Variable**
  - Optional field
    - lowerBound / DW_AT_lower_bound: Constant

| DISubrange | Mandatory ? | Constant | Variable | Expression |
|---|---|---|---|---|
| **count** | **Yes** | **Yes** | **Yes** | **No** |
| **lowerBound** | **No** | **Yes** | **No** | **No** |

```
1  #include<stdio.h>
2  void fun(unsigned long size) {
3    int arr[size];
4    printf("size of arr = %lu", sizeof(arr));
5    printf("arr[0] = %d", arr[0]);
6  }
```

```
!17 = !DILocalVariable(name: "__vla_expr0", scope: !8, type: !11, flags: DIFla…tificial)
!19 = !DILocalVariable(name: "arr", scope: !8, file: !1, line: 3, type: !20)
!20 = !DICompositeType(tag: DW_TAG_array_type, baseType: !21, elements: !22)
!22 = !{!23}
!23 = !DISubrange(count: !17)
```

```
0x0000004d:     DW_TAG_variable
                  DW_AT_location        (DW_OP_fbreg -24)
                  DW_AT_name    ("__vla_expr0")
                  DW_AT_type    (0x00000069 "unsigned long")
                  DW_AT_artificial      (true)

0x00000070:   DW_TAG_array_type
                DW_AT_type      (0x0000007f "int")

0x00000075:     DW_TAG_subrange_type
                  DW_AT_type    (0x00000086 "__ARRAY_SIZE_TYPE__")
                  DW_AT_count   (0x0000004d)
```

AMD

# Array debugging: Enhancement needed…

***Fortran explicit shape array***

Defined as arr(2:10),

where lowerBound=2,

upperBound=10 and

count=10-2+1=9

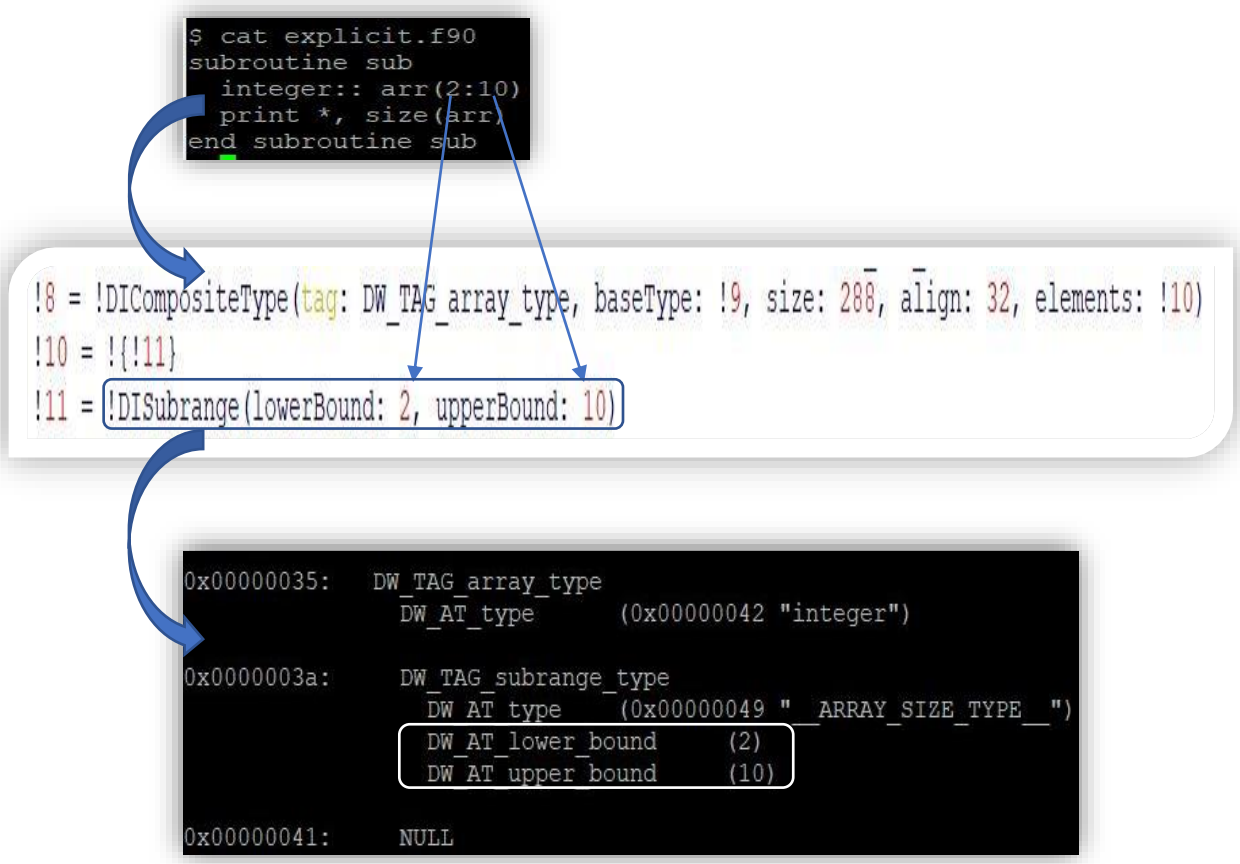
- Count is natural property for C/C++ and upperBound is natural for Fortran.

# Array debugging: Enhancement needed…

***Fortran explicit shape array***

## Metadata

- DISubrange
  - *Mandatory*
    - count: Constant / Variable, or
    - **upperBound / DW_AT_upper_bound: Constant**
  - *Optional*
    - lowerBound: Constant



# Array debugging: Enhancement needed…

***Fortran adjustable array*** is a dummy argument or local array with one or more of its dimensions or bounds as an expression of integer variables that are either themselves dummy arguments or are in a common block.

## Metadata

DISubrange

- *Mandatory*
  - count: Constant / Variable
  - upperBound: Constant **/ Variable**
- *Optional*
  - lowerBound: Constant **/ Variable**

```
$ cat -n adjustable.f90
     1  subroutine sub(arr, bound1, bound2)
     2    integer :: bound1, bound2
     3    real :: arr(5:5, bound1:bound2)
     4    print *, size(arr)
     5  end subroutine sub
```

```
!10 = !DICompositeType(tag: DW_TAG_array_type, baseType: !11, size: 32, align: 32, elements: !12)
!12 = !{!13, !14}
!13 = !DISubrange(lowerBound: 5, upperBound: 5)
!14 = !DISubrange(lowerBound: !15, upperBound: !17)
!15 = distinct !DILocalVariable(scope: !3, file: !4, line: 4, type: !16, flags: DIFlagArtificial)
!17 = distinct !DILocalVariable(scope: !3, file: !4, line: 4, type: !16, flags: DIFlagArtificial)
```

```
0x00000078:      DW_TAG_variable

0x00000082:      DW_TAG_variable

0x00000106:    DW_TAG_array_type
                 DW_AT_type      (0x00000120 "real")

0x0000010b:      DW_TAG_subrange_type
                   DW_AT_type    (0x00000127 "__ARRAY_SIZE_TYPE__")
                   DW_AT_lower_bound     (5)
                   DW_AT_upper_bound     (5)

0x00000112:      DW_TAG_subrange_type
                   DW_AT_type    (0x00000127 "__ARRAY_SIZE_TYPE__")
                   DW_AT_lower_bound     (0x00000078)
                   DW_AT_upper_bound     (0x00000082)
```

AMD

# Array Debugging: Inside gdb

```
9           print *, size(arr)
(gdb) ptype arr
type = real (5:4,0)
(gdb) print arr
$1 = ()
```

```
9           print *, size(arr)
(gdb) ptype arr
type = real (5:5,5)
(gdb) print arr
$1 = ((2) (2) (2) (2) (2))
```

AMD

# Array debugging: Enhancement needed…

***Assumed size array*** is a dummy argument, and which has an asterisk as the upper bound of the last dimension.

```
$ cat -n assumedSize.f90
     1  subroutine sub ( arr)
     2    integer :: arr(5, *)
     3    print *, size(arr,1)
     4  end subroutine sub
```

Metadata

- DISubrange
  - *Mandatory* ***(Optional for Fortran)***
    - count: Constant / Variable, or
    - upperBound: constant / Variable
  - *Optional*
    - lowerBound: Constant

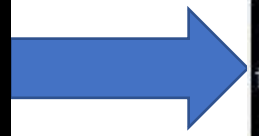
```
!8 = !DICompositeType(tag: DW_TAG_array_type, baseType: !9, align: 32, elements: !10)
!10 = !{!11, !12}
!11 = !DISubrange(lowerBound: 1, upperBound: 5)
!12 = !DISubrange(lowerBound: 1)
```

```
0x00000060:   DW_TAG_array_type
                DW_AT_type      (0x00000059 "integer")

0x00000065:     DW_TAG_subrange_type
                  DW_AT_type    (0x00000071 "__ARRAY_SIZE_TYPE__")
                  DW_AT_upper_bound     (5)

0x0000006b:     DW_TAG_subrange_type
                  DW_AT_type    (0x00000071 "__ARRAY_SIZE_TYPE__")

0x00000070:     NULL
```

AMD

# Array Debugging: Inside gdb

```
Breakpoint 2, main::sub (arr=...) at assumedSize.f90:8
8         print *, size(arr,1)
(gdb) ptype arr
type = integer (5,0)
(gdb) print arr(5,2)
no such vector element
```



```
Breakpoint 2, main::sub (arr=...) at assumedSize.f90:8
8         print *, size(arr,1)
(gdb) ptype arr
type = integer (5,*)
(gdb) print arr(5,2)
$3 = 1
```

AMD

# Array debugging: array descriptor

- Dynamic arrays have two properties
  - Array descriptor
  - Data address
- Extents can be accessed using mathematical expressions

  - Count1 = *(base_address + 80 + 48 + 8) = *(base_address + 136)
    - !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 112, DW_OP_deref)
  - Stride1= mulitplier1 * byte_length = *(base_address + 80+ 48 + 32) * *(base_address + 24)
    - !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, **160**, DW_OP_deref , DW_OP_push_object_address, DW_OP_plus_uconst, **24**, DW_OP_deref , DW_OP_mul)

| Header (size=0x50=80) | |
|---|---|
| base addr + 0x00 | Tag |
| base addr + 0x08 | rank |
| base addr + 0x10 | kind |
| base addr + 0x18 | byte length |
| base addr + 0x20 | |
| base addr + 0x28 | |
| base addr + 0x30 | |
| base addr + 0x38 | |
| base addr + 0x40 | |
| base addr + 0x48 | |

| DIM(0) (size=0x30=48) | |
|---|---|
| base addr + 0x50 | lower |
| base addr + 0x58 | extent |
| base addr + 0x60 | |
| base addr + 0x68 | |
| base addr + 0x70 | multiplier |
| base addr + 0x78 | upper |

| DIM(1) | |
|---|---|
| base addr + 0x80 | lower |
| base addr + 0x88 | extent |
| base addr + 0x90 | |
| base addr + 0x98 | |
| base addr + 0xa0 | multiplier |
| base addr + 0xa8 | upper |

| DIM(n) | |
|---|---|
| base addr + size(Header) + n * size(DIM) + 0x00 | lower |
| base addr + size(Header) + n * size(DIM) + 0x08 | extent |
| base addr + size(Header) + n * size(DIM) + 0x10 | |
| base addr + size(Header) + n * size(DIM) + 0x18 | |
| base addr + size(Header) + n * size(DIM) + 0x20 | multiplier |
| base addr + size(Header) + n * size(DIM) + 0x28 | upper |

AMD

# Array debugging: Enhancement needed…

***Assumed shape array*** is dummy argument which has different lower bound than actual argument, but extent and stride are passed from caller.

## Metadata

- **DW_OP_push_object_address**
- DICompositeType(tag: DW_TAG_array_type
  - …
  - **dataLocation / DW_AT_data_location : variable**
- DISubrange
  - *Mandatory (optional for Fortran)*
    - Count / upperBound: Constant / Variable **/ expression**,
  - *Optional*
    - lowerBound: Constant / variable
    - **stride / DW_AT_byte_stride: expression**

```
$ cat -n assumedShape.f90
     1  subroutine sub(arr)
     2    real :: arr(2:,:)
     3    print *, size(arr)
     4  end subroutine sub
```

```
!23 = !DICompositeType(tag: DW_TAG_array_type, baseType: !9, size: 32, align: 32, elements: !24, dataLocation: !20)
!24 = !{!25, !26}
!25 = !DISubrange(count: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 88, DW_OP_deref), lowerBound: !12,
 stride: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 112, DW_OP_deref, DW_OP_push_object_address,
DW_OP_plus_uconst, 24, DW_OP_deref, DW_OP_mul))
!26 = !DISubrange(count: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 136, DW_OP_deref), lowerBound: 1,
 stride: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 160, DW_OP_deref, DW_OP_push_object_address,
DW_OP_plus_uconst, 24, DW_OP_deref, DW_OP_mul))
```

```
0x000000f1:   DW_TAG_array_type
                DW_AT_data_location     (0x0000003f)
                DW_AT_type      (0x00000129 "real")

0x000000fa:     DW_TAG_subrange_type
                  DW_AT_type    (0x00000130 "__ARRAY_SIZE_TYPE__")
                  DW_AT_lower_bound     (0x00000059)
                  DW_AT_count   (DW_OP_push_object_address, DW_OP_plus_uconst 0x58, DW_OP_deref)
                  DW_AT_byte_stride     (DW_OP_push_object_address, DW_OP_plus_uconst 0x70, DW_OP_deref,
DW_OP_push_object_address, DW_OP_plus_uconst 0x18, DW_OP_deref, DW_OP_mul)

0x00000112:     DW_TAG_subrange_type
                  DW_AT_type    (0x00000130 "__ARRAY_SIZE_TYPE__")
                  DW_AT_count   (DW_OP_push_object_address, DW_OP_plus_uconst 0x88, DW_OP_deref)
                  DW_AT_byte_stride     (DW_OP_push_object_address, DW_OP_plus_uconst 0xa0, DW_OP_deref,
 DW_OP_push_object_address, DW_OP_plus_uconst 0x18, DW_OP_deref, DW_OP_mul)

0x00000128:     NULL
```

# Array Debugging: Inside gdb

```
Breakpoint 2, main::sub (arr=...) at assumedShape.f90:8
8           print *, size(arr)
(gdb) ptype arr
type = real (0,0)
(gdb) print arr
$1 = ()
```

```
Breakpoint 3, main::sub (arr=...) at assumedShape.f90:8
8           print *, size(arr)
(gdb) ptype arr
type = real (2:6,5)
(gdb) print arr
$1 = ((1, 1, 1, 1, 1) (1, 1, 1, 1, 1) (1, 1, 1, 1, 1) (1, 1, 1, 1, 1) (1, 1, 1, 1, 1))
```

AMD

# Array debugging: Enhancement needed…

***Allocatable array:*** descriptor, data, validity.

## Metadata

- DICompositeType(tag: DW_TAG_array_type
  - …
  - **allocated / DW_AT_allocated: Variable / Expression**
- DISubrange
  - *Mandatory (Optional for Fortran)*
    - Count: Constant / Variable/expression,
    - upperBound: Constant / variable / expression
  - *Optional*
    - lowerBound: Constant / Variable / expression
    - stride: Constant / variable / expression

```
$ cat -n allocated.f90
     1  program main
     2  integer, allocatable :: arr(:)
     3  allocate(arr(2:20))
     4  arr(2)=99
     5  print *, arr
     6  end program main
```

```
!20 = !DICompositeType(tag: DW_TAG_array_type, baseType: !10, size: 32, align: 32, elements: !21,
dataLocation: !8, allocated: !12)
!21 = !{!22}
!22 = !DISubrange(lowerBound: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 80, DW_OP_deref),
upperBound: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 120, DW_OP_deref),
stride: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 112, DW_OP_deref,
DW_OP_push_object_address, DW_OP_plus_uconst, 24, DW_OP_deref, DW_OP_mul))
```

```
0x0000004b:      DW_TAG_variable
                   DW_AT_location        (DW_OP_fbreg +224)
                   DW_AT_decl_file       ("/home/alok/work/array/allocated.f90")
                   DW_AT_decl_line       (2)
                   DW_AT_type    (0x00000115 "logical")
                   DW_AT_artificial      (true)

0x00000061:      DW_TAG_variable
                   DW_AT_location        (DW_OP_fbreg +96)
                   DW_AT_name    ("arr")
                   DW_AT_decl_file       ("/home/alok/work/array/allocated.f90")
                   DW_AT_decl_line       (6)
                   DW_AT_type    (0x00000136 "integer[]")

0x00000136:    DW_TAG_array_type
                 DW_AT_data_location     (0x00000040)
                 DW_AT_allocated (0x0000004b)
                 DW_AT_type      (0x0000010e "integer")

0x00000143:      DW_TAG_subrange_type
                   DW_AT_type    (0x0000012f "__ARRAY_SIZE_TYPE__")
                   DW_AT_lower_bound     (DW_OP_push_object_address, DW_OP_plus_uconst 0x50, DW_OP_deref
                   DW_AT_upper_bound     (DW_OP_push_object_address, DW_OP_plus_uconst 0x78, DW_OP_deref
                   DW_AT_byte_stride     (DW_OP_push_object_address, DW_OP_plus_uconst 0x70, DW_OP_deref
DW_OP_push_object_address, DW_OP_plus_uconst 0x18, DW_OP_deref, DW_OP_mul)
```

AMD

# Array Debugging: Inside gdb

```
Breakpoint 2, main () at allocated.f90:3
3       allocate(arr(2:20))
(gdb) ptype arr
type = integer (0)
(gdb) print arr
$1 = ()
(gdb) next 2
5       print *, arr
(gdb) print arr
$2 = ()
(gdb) ptype arr
type = integer (0)
```

```
Breakpoint 1, main () at allocated.f90:3
3       allocate(arr(2:20))
(gdb) ptype arr
type = integer, allocatable (:)
(gdb) print arr
$3 = <not allocated>
(gdb) next 2
5       print *, arr
(gdb) print arr
$4 = (99, 0, <repeats 18 times>)
(gdb) ptype arr
type = integer, allocatable (2:20)
```

AMD

# Array debugging: Enhancement needed…

***Pointer array***: descriptor, data, validity.

## Metadata

- DICompositeType(tag: DW_TAG_array_type
  - …
  - **associated / DW_AT_associated: Variable / Expression**
- DISubrange
  - *Mandatory (Optional for Fortran)*
    - Count: Constant / Variable/expression,
    - upperBound: Constant / variable / expression
  - *Optional*
    - lowerBound: Constant / Variable / expression
    - stride: Constant / variable / expression

```
$ cat -n associated.f90
     1  program main
     2  integer, pointer :: arr(:)
     3  allocate(arr(2:20))
     4  arr(2)=99
     5  print *, arr
     6  end program main
```

```
!20 = !DICompositeType(tag: DW_TAG_array_type, baseType: !10, size: 32, align: 32, elements: !21,
 dataLocation: !8, associated: !12)
!21 = !{!22}
!22 = !DISubrange(lowerBound: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 80, DW_OP_deref),
 upperBound: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 120, DW_OP_deref),
 stride: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 112, DW_OP_deref,
 DW_OP_push_object_address, DW_OP_plus_uconst, 24, DW_OP_deref, DW_OP_mul))
```

```
0x000000f6:   DW_TAG_array_type
                DW_AT_data_location     (0x00000040)
                DW_AT_associated        (0x0000004b)
                DW_AT_type      (0x000000ce "integer")

0x00000103:     DW_TAG_subrange_type
                  DW_AT_type    (0x000000ef "__ARRAY_SIZE_TYPE__")
                  DW_AT_lower_bound     (DW_OP_push_object_address, DW_OP_plus_uconst 0x50,
DW_OP_deref)
                  DW_AT_upper_bound     (DW_OP_push_object_address, DW_OP_plus_uconst 0x78,
DW_OP_deref)
                  DW_AT_byte_stride     (DW_OP_push_object_address, DW_OP_plus_uconst 0x70,
DW_OP_deref, DW_OP_push_object_address, DW_OP_plus_uconst 0x18, DW_OP_deref, DW_OP_mul)
```

AMD

# Array Debugging: Inside gdb

```
Breakpoint 2, main () at associated.f90:3
3       allocate(arr(2:20))
(gdb) ptype arr
type = integer (0)
(gdb) print arr
$1 = ()
(gdb) next 2
5       print *, arr
(gdb) print arr
$2 = ()
(gdb) ptype arr
type = integer (0)
```

```
Breakpoint 1, main () at associated.f90:3
3       allocate(arr(2:20))
(gdb) ptype arr
type = integer (:)
(gdb) print arr
$1 = <not associated>
(gdb) next 2
5       print *, arr
(gdb) print arr
$2 = (99, 0, <repeats 18 times>)
(gdb) ptype arr
type = integer (2:20)
```

AMD

# Array debugging: Enhancement needed…

***Assumed rank***: Rank not constant.

## Metadata

- **DW_OP_over**
- DICompositeType(tag: DW_TAG_array_type
  - …
  - **rank / DW_AT_rank: Variable / Expression**
- **DIGenericSubrange / DW_TAG_generic_subrange**
  - ***Mandatory (any one of below)***
    - **Count: Constant / Variable/expression,**
    - **upperBound: Constant / variable / expression**
  - ***Mandatory (all)***
    - **lowerBound: Constant / Variable / expression**
    - **stride: Constant / variable / expression**

```
$ cat -n assumedRank.f90
     1  subroutine sub(arank)
     2    real :: arank(..)
     3    print *, RANK(arank)
     4  end
```

```
!20 = !DICompositeType(tag: DW_TAG_array_type, baseType: !9, size: 32, align: 32, elements: !21, dataLocation: !17,
rank: !DIExpression(DW_OP_push_object_address, DW_OP_plus_uconst, 8, DW_OP_deref, DW_OP_constu, 7, DW_OP_and))
!21 = !{!22}
!22 = !DIGenericSubrange(lowerBound: !DIExpression(DW_OP_push_object_address, DW_OP_over, DW_OP_constu, 48,
DW_OP_mul, DW_OP_plus_uconst, 80, DW_OP_plus, DW_OP_deref),
upperBound: !DIExpression(DW_OP_push_object_address, DW_OP_over, DW_OP_constu, 48, DW_OP_mul, DW_OP_plus_uconst,
120, DW_OP_plus, DW_OP_deref),
stride: !DIExpression(DW_OP_push_object_address, DW_OP_over, DW_OP_constu, 48, DW_OP_mul,
DW_OP_plus_uconst, 112, DW_OP_plus, DW_OP_deref, DW_OP_push_object_address, DW_OP_plus_uconst, 24, DW_OP_deref,
DW_OP_mul))
```

```
0x0000006a:   DW_TAG_array_type
                DW_AT_data_location     (0x00000032)
                DW_AT_rank      (DW_OP_push_object_address, DW_OP_plus_uconst 0x8, DW_OP_deref,
DW_OP_lit7, DW_OP_and)
                DW_AT_type      (0x000000a3 "real")

0x0000007a:     DW_TAG_generic_subrange
                  DW_AT_type    (0x000000a7 "__ARRAY_SIZE_TYPE__")
                  DW_AT_lower_bound     (DW_OP_push_object_address, DW_OP_over, DW_OP_constu 0x30,
DW_OP_mul, DW_OP_plus_uconst 0x50, DW_OP_plus, DW_OP_deref)
                  DW_AT_upper_bound     (DW_OP_push_object_address, DW_OP_over, DW_OP_constu 0x30,
DW_OP_mul, DW_OP_plus_uconst 0x78, DW_OP_plus, DW_OP_deref)
                  DW_AT_byte_stride     (DW_OP_push_object_address, DW_OP_over, DW_OP_constu 0x30,
DW_OP_mul, DW_OP_plus_uconst 0x70, DW_OP_plus, DW_OP_deref, DW_OP_push_object_address,
DW_OP_plus_uconst 0x18, DW_OP_deref, DW_OP_mul)
```

AMD

# Array Debugging: Inside gdb

```
Breakpoint 1, main::sub (arank=...) at assumedRank.f90:8
8           print *, RANK(arank)
(gdb) ptype arank
type = real (2:5,3:5)
(gdb) print arank
$1 = ((1, 1, 1, 1) (1, 1, 1, 1) (1, 1, 1, 1))
(gdb) up
#1  0x0000000000201d13 in main () at assumedRank.f90:4
4           call sub(myarr)
(gdb) ptype myarr
type = real (2:5,3:5)
```

AMD

# Array debugging: Recap

- Added support for DWARF operators
  - ***DW_OP_push_object_address***
  - ***DW_OP_over***
- Added DIGenericSubrange

| New DIGenericSubrange | Mandatory | Constant | Variable | Expression |
|---|---|---|---|---|
| count / upperBound | ***Yes*** | ***Yes*** | ***Yes*** | ***Yes*** |
| lowerBound | ***Yes*** | ***Yes*** | ***Yes*** | ***Yes*** |
| stride | ***Yes*** | ***Yes*** | ***Yes*** | ***Yes*** |

AMD

# Array debugging: Recap

- Upgraded DICompositeType
- Upgraded DISubrange

| Updated DICompositeType(tag: DW_TAG_array_type (DW_TAG_array_type) | | | | |
|---|---|---|---|---|
| **Metadata** | ***dataLocation*** | ***allocated*** | ***associated*** | ***rank*** |
| **DWARF** | DW_AT_data_location | DW_AT_allocated | DW_AT_associated | DW_AT_rank |

| New DIGenericSubrange | Mandatory | Constant | Variable | Expression |
|---|---|---|---|---|
| count / upperBound | ***Yes*** | ***Yes*** | ***Yes*** | ***Yes*** |
| lowerBound | ***Yes*** | ***Yes*** | ***Yes*** | ***Yes*** |
| stride | ***Yes*** | ***Yes*** | ***Yes*** | ***Yes*** |

AMD

# String debugging

Fortran Character or string type may have string length parameter either assumed or deferred.

## Metadata

- **DIStringType(stringLength, stringLengthExpression**
  - *Mandatory*
    - **StringLength:** Variable
- Special thanks: **Eric Schweitz**

```
(gdb) ptype string
type = character (5)
(gdb) p string
$1 = 'Hello'
```

```
$ cat -n assumedLenString.f90
     1  program assumedLength
     2    call sub('Hello')
     3    contains
     4     subroutine sub(string)
     5       implicit none
     6       character(len=*) :: string
     7       print *, string
     8     end subroutine sub
     9  end program assumedLength
```

```
!15 = !DIStringType(name: "character(*)!2", stringLength: !16, stringLengthExpression: !DIExpression(), size: 32)
!16 = !DILocalVariable(arg: 2, scope: !10, file: !3, line: 4, type: !17, flags: DIFlagArtificial)
```

```
0x0000007d:       DW_TAG_formal_parameter
                    DW_AT_location      (DW_OP_fbreg +16)
                    DW_AT_type  (0x0000009e "integer*8")
                    DW_AT_artificial    (true)

0x000000ac:   DW_TAG_string_type
                DW_AT_name      ("character(*)!2")
                DW_AT_string_length     (0x0000007d)
```

# Module debugging

Fortran ***Module*** is a collections of declarations and subprograms that may be imported to other program units.

- Metadata
  - DIModule
    - …
    - *Mandatory*
      - **file**
      - **line**

```
$ cat -n module.f90
     1  module mod1
     2    integer :: var_i = 1
     3  end module mod1
     4
     5  program module
     6    use mod1
     7    print *, var_i
     8  end
```

```
!1 = distinct !DIGlobalVariable(name: "var_i", scope: !2, ..., isDefinition: true)
!2 = !DIModule(scope: !4, name: "mod1", file: !3, line: 1)
!4 = distinct !DICompileUnit(language: DW_LANG_Fortran90, ..., imports: !7)
!7 = !{!8}
!8 = !DIImportedEntity(tag: DW_TAG_imported_module, scope: !9, entity: !2, file: !3, line: 5)
!9 = distinct !DISubprogram(name: "module", scope: !4, ...)
```

```
0x0000002a:   DW_TAG_module
                DW_AT_name      ("mod1")
                DW_AT_decl_file ("/home/alok/work/other/module.f90")
                DW_AT_decl_line (1)

0x00000031:     DW_TAG_variable
                  DW_AT_name    ("var_i")
                  DW_AT_type    (0x00000047 "integer")
                  DW_AT_external        (true)
                  DW_AT_decl_file       ("/home/alok/work/other/module.f90")
                  DW_AT_decl_line       (2)
                  DW_AT_location        (DW_OP_addr 0x0)

0x0000004e:   DW_TAG_subprogram
                DW_AT_name      ("module")
                ...
0x0000006e:     DW_TAG_imported_module
                  DW_AT_decl_file       ("/home/alok/work/other/module.f90")
                  DW_AT_decl_line       (5)
                  DW_AT_import  (0x0000002a)
```

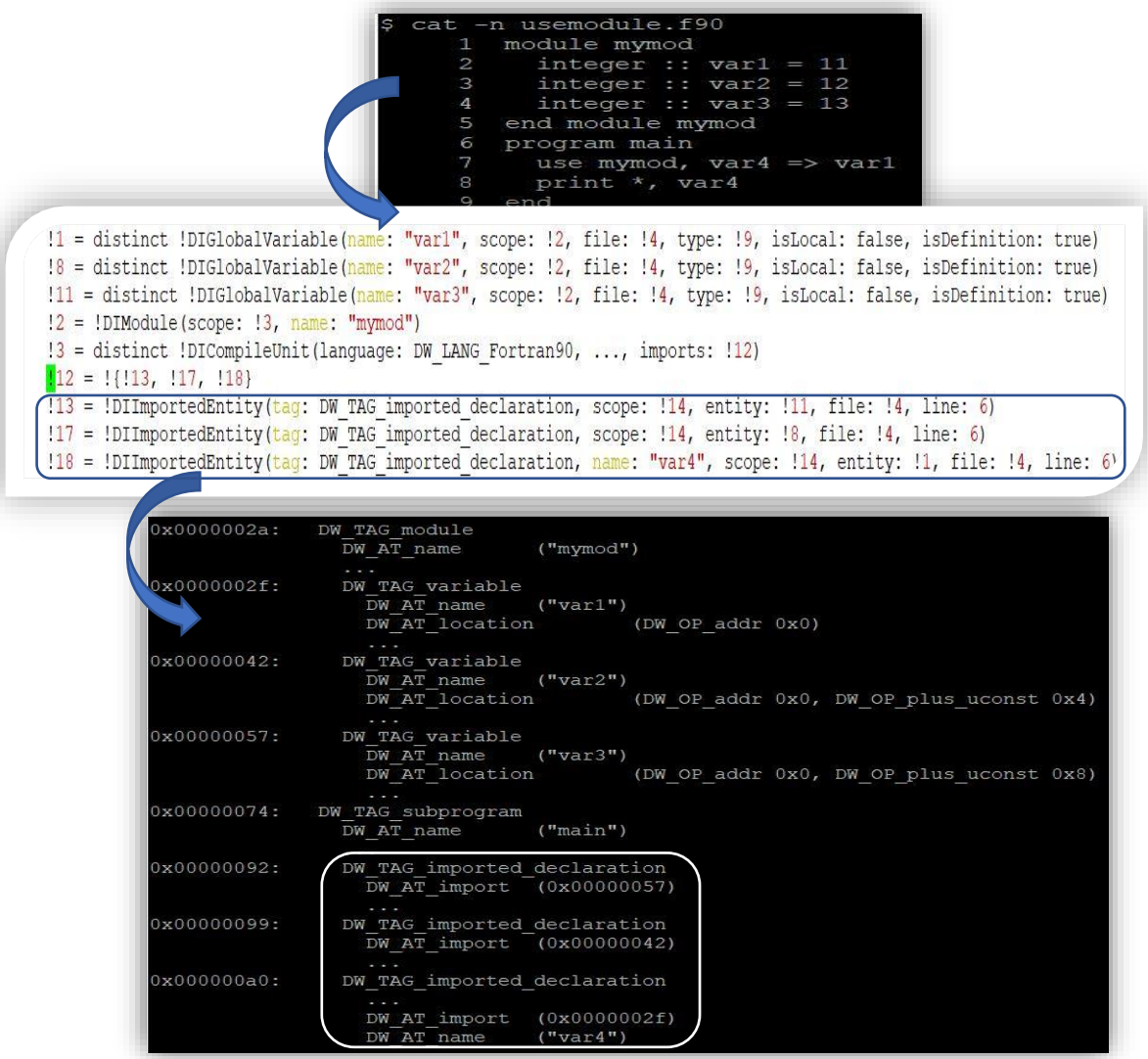
```
(gdb) info modules
All defined modules:
(gdb)
```

```
(gdb) info modules
All defined modules:

File module.f90:
1:      mod1
(gdb)
```

# Module debugging

***Renamed use*** of module can be represented using DIImportEntity.

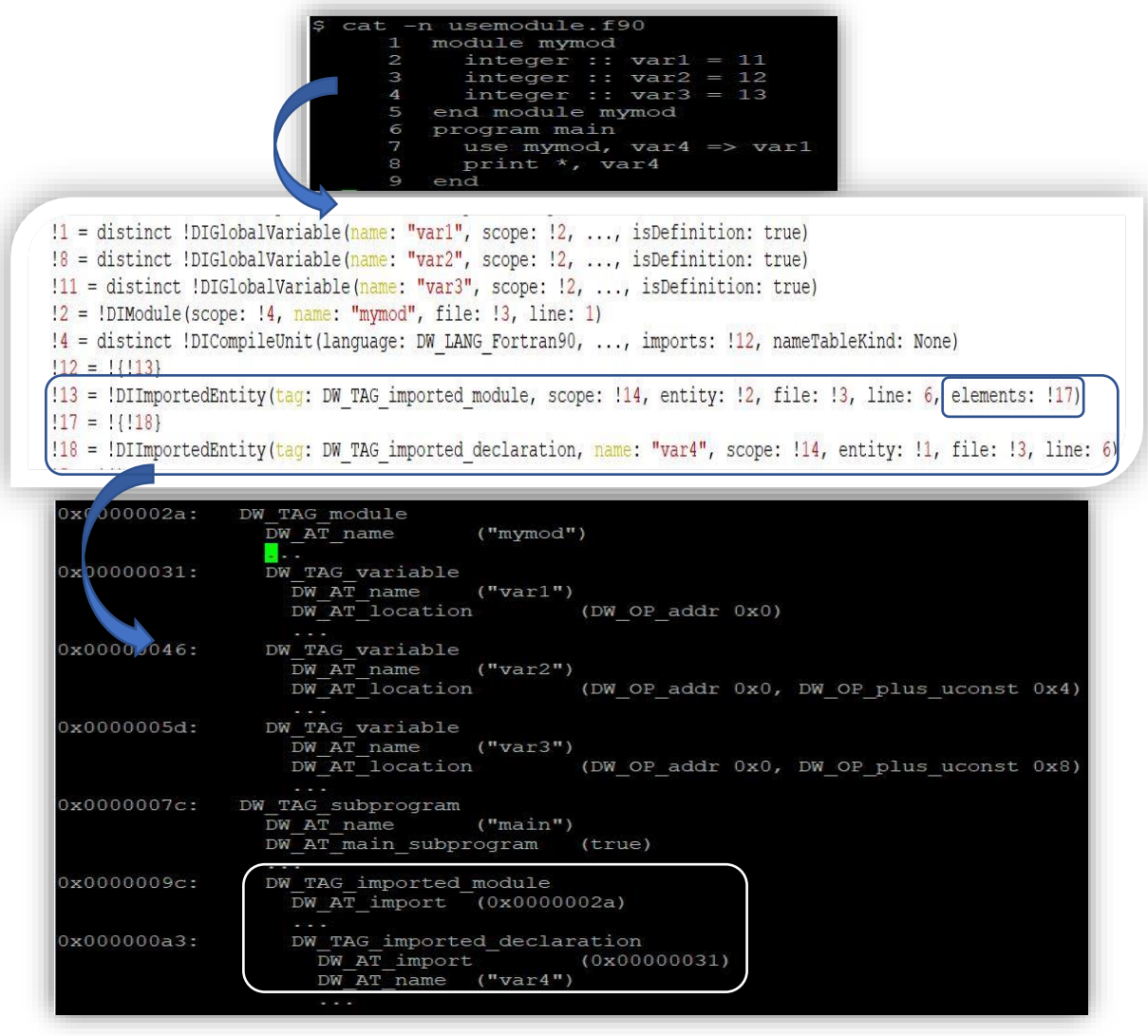
Do you see a problem here?




# Module debugging

***Renamed use*** of module can be represented using DIImportEntity.

Renamed variable having child parent relation ship with module can optimize debug info size.



# Namelist debugging

Fortran ***Namelist*** is group of variables. These variables called as namelist items can be local, global, parameter variables.

## Metadata

- DICompositeType(**tag: DW_TAG_namelist**
  - *Mandatory*
    - elements

```
(gdb) n
7           Write(*,nml)
(gdb) print nml
$1 = ( a = 10, b = 20 )
(gdb) ptype nml
type = Type nml
    integer :: a
    integer :: b
End Type nml
```

```
$ cat -n namelist.f90
     1  program main
     2    integer :: a, b
     3    namelist /nml/ a, b
     4    a = 10
     5    b = 20
     6    Write(*,nml)
     7  end program main
```

```
!19 = !DICompositeType(tag: DW_TAG_namelist, name: "nml", scope: !5, file: !3, elements: !20)
!20 = !{!8, !11}
!8 = !DILocalVariable(name: "a", scope: !5, file: !3, line: 2, type: !9)
!11 = !DILocalVariable(name: "b", scope: !5, file: !3, line: 2, type: !9)
!9 = !DIBasicType(name: "integer", size: 32, align: 32, encoding: DW_ATE_signed)
```

```
0x00000040:     DW_TAG_variable
                  DW_AT_location        (DW_OP_fbreg +36)
                  DW_AT_name    ("a")
                  DW_AT_type    (0x0000008c "integer")

0x0000004e:     DW_TAG_variable
                  DW_AT_location        (DW_OP_fbreg +32)
                  DW_AT_name    ("b")
                  DW_AT_type    (0x0000008c "integer")

0x0000007b:     DW_TAG_namelist
                  DW_AT_name    ("nml")

0x00000080:       DW_TAG_namelist_item
                    DW_AT_namelist_item (0x00000040)

0x00000085:       DW_TAG_namelist_item
                    DW_AT_namelist_item (0x0000004e)
```

# Conclusion

- Enhanced debug experience for Fortran.
- Any new Fortran front end can leverage these LLVM enhancements
- Any other language front end can make use of it.

AMD

# Thank You

Any Questions?