What is Going on with Debugging Info?

DWARF6 and Support for GPUs and Vectorized Code

Ben Woodard

Red Hat

Disclaimer

I'm presenting this but this is **certainly** not exclusively my work. This is the work of the DWARF for GPU's group. Participants include:

- Started AMD and they did the original design
- TotalView (John DelSignore) instrumental in the design
- Led by Cary Coutant the DWARF Committee chair
- Intel has made important contributions
- Nvidia is now participating as well.
- Red Hat

Source:
https://llvm.org/docs/AMDGPUDwarfExtensionsForHeterogeneousDebugging.html#introduction
https://github.com/ccoutant/dwarf-locations/tree/main
https://snapshots.sourceware.org/dwarfstd/dwarf-spec/2025-05-17_20-15_1747512902/dwarf6-20250517-2014.pdf

Overall status of DWARF

After several years of hiatus post DWARF5. The DWARF committee is meeting regularly being led by Cary Coutant (ELF standard maintainer).

- Administration change last year has been very positive.
- People are working constructively
- Issues are being worked through the process
- Draft standard is being published in the open.
- Some mailing lists are open.
- DWARF6 is probably about 1 year out (guess)
- GPU support is the marquee feature.

Source:
https://lists.dwarfstd.org/mailman/listinfo/dwarf-discuss
https://dwarfstd.org/issues.html
https://snapshots.sourceware.org/dwarfstd/dwarf-spec/

Simple changes

Basic needed cleanups

- Vector/Matrix Tensor types
- Vendor extensions
- Expression Context
- Lane operator

Vector/Matrix Tensor

Status: approved in the current draft standard

Original design as a GNU vendor attribute back in 2001. Just never made it into the original standard. Supports intrinsic vector types.

- Standardized and extended existing behavior
- Now supports multidimensional types. E.g. matrix registers
- Different than C arrays in that they don't decay into pointers
- Can be passed by value

Source:
https://sourceware.org/elfutils/DwarfExtensions
https://dwarfstd.org/issues/230413.1.html

Vendor extensions are now Producer extensions

Status: partially accepted

Three parts. Concept: tools no longer vertically integrated within a vendor supplied toolchain. Most tools and toolchains are open source.

- Numerous catalogs of vendor extensions gathered into a registry
- No longer thought of as “private agreement between producer and consumer within a vendor toolchain”. Consumers must support multiple producers.
- DWARF6 flag day - shot down

Source:
https://dwarfstd.org/issues/231110.2.html
https://dwarfstd.org/issues/231110.4.html - rejected
https://dwarfstd.org/issues/231110.3.html

Expression Context

Status: accepted

Excerpt from AMD's DWARF for GPUs proposal. Editorial in nature.

- Expected result of evaluation
- Initial stack (not always empty a kind of ABI)
- Compilation unit
- Target architecture
- Current thread
- Call frame
- PC
- Current lane
- Current object

Source:
lhttps://github.com/ccoutant/dwarf-locations/blob/main/001a-context.md
https://dwarfstd.org/issues/241011.1.html

Push lane

Status: accepted

Part of Intel's approach to support DWARF for vector registers.

- Much of their original approach to support vector superseded
- DW_OP_push_lane incorporated into expression context section.

Source:
https://dwarfstd.org/issues/211206.1.html

Supporting GPUs

CPUs vs GPUs

CPUs

- Single address space
- Relatively few registers
- Few vector registers
- Threads are independent

GPUs

- Multiple address spaces with different pointer sizes
- Many registers (hundreds)
- Many vector registers (huge)
- Threads are often fused (wave/warp)
- Weird things - doubles spanning registers

Locations on the DWARF stack – I

Status: final review

Conceptually simple but ended up being the biggest and most controversial change. Several alternative proposals. The DWARF committee now understands it.

- Huge reorganization of Chapter 2. Split making a new Chapter 3
- Pushing a location onto the stack allows it to be further modified.
- DW_OP_call is much more versatile
- Composites are now built on the stack rather on the side

Source:
https://github.com/ccoutant/dwarf-locations/blob/main/005-locations-on-stack.md
Alternate: https://dwarfstd.org/issues/250408.1.html

Locations on the DWARF stack – II

Status: final review

Think of locations as a tuple

- Either (storage, offset, bit_offset) or (storage, (offset, bit_offset))
- Locations reference storage i.e. registers, memory, undefined, implicit, and composite.
  - "Storage" is an abstraction for all types.
- Locations have an offset into that storage bytes, bits.
  - DW_OP_offset, DW_OP_bit_offset
  - Not the same as DW_OP_plus which is for values not locations
  - Registers can offsets too - more useful for vector registers

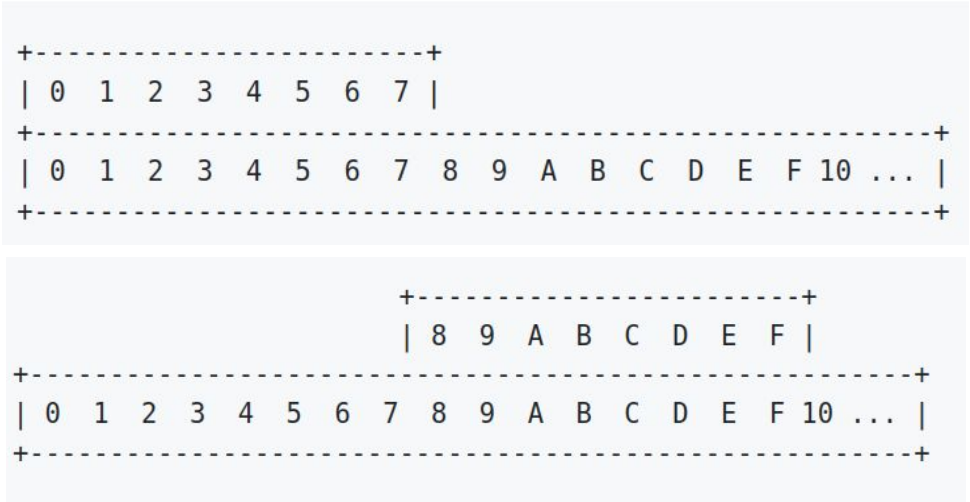
Overlays

Status: nearly ready for submission

A better way to make composites.

- No one liked DW_OP_piece - rarely used
- DW_OP_bit_piece had endian problems
- Originally designed for vector registers but uses have expanded

```
DW_OP_addr 0x100 #base
DW_OP_reg1     # overlay
DW_OP_breg0 # loop index
DW_OP_lit8    # width of type
DW_OP_mul   # offset
DW_OP_lit8    # width
DW_OP_overlay
```



Source:
lhttps://github.com/ccoutant/dwarf-locations/blob/main/016-overlay-composite-location-descriptions.md

Address Spaces

Status: on deck from GPU group (almost done)

GPUs have multiple address spaces e.g. LDS, GDS

- Not DW_OP_xderef (what nvidia tried)
- DW_OP_form_aspace_address - adds an address space number to a memory location.
- Many times **values** were used as an address through implicit conversion.
- Pointers in alternate address spaces are not necessarily the same width as in the system address space. Need a way to change a type for pointers.

Source:
lhttps://github.com/ccoutant/dwarf-locations/blob/main/013-generalize-address-spaces.md

Refined types

Status: not yet written

Name subject to change.

- Size of a pointer may change based on address space
- E.g. 80b extended float vs. 64b double
- Compiler proves that a smaller type can be used.

Other things on deck

Status: not complete

Still many changes to come

- CFI/CFA needs a bit of work for address spaces
- Need more operations to selective spill registers based upon exec mask DW_OP_extend DW_OP_select_bit_piece
  - Kind of like overlay but with bitmasks
  - Needed for divergent flow control
  - Needed for logical PCs in SIMT sections with fused threads
- Memory spaces - type modifier for source language e.g. “private”
- Semantics of operations when a loclist yields multiple locations
- Lots of clarifications and details

Source:
https://github.com/ccoutant/dwarf-locations/blob/main/003-clarifications-cfi.txt https://github.com/ccoutant/dwarf-locations/blob/main/017-call-frame-entry-registers.txt
https://github.com/ccoutant/dwarf-locations/blob/main/015-vector-composite-location-descriptions.txt
https://github.com/ccoutant/dwarf-locations/blob/main/022-memory-spaces.txt

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat

Crazy Ideas

Status: not yet written

Let’s see what we can do with these

- Make split-dwarf work for packages like RPM and packagers like spack
- Extend CFI table to refer to variables and expressions
- Formalize rules of arithmetic for values vs. locations
- Retire antiquated DWARF logo and replace it with