Various notions of „format“
Hegel seems to me to be always wanting to say that things which look different are really the same. Whereas my interest is in showing that things which look the same are really different. I was thinking of using as a motto for my book a quotation from King Lear: "I'll teach you differences." Drury, "Conversations with Wittgenstein."
We might specify some formats but allow others to be implementation-dependent, e.g.

This raises two obvious questions:

• An internal/external/memory/storage format.
• An exchange/external/memory/storage format.
• What should we standardize and what not?
• What do these terms mean?
Three (at least) significantly different ideas of exchange:

- Passing data in files or over a network, perhaps between different computers / organizations / continents
- Passing data around in a cluster of local machines, e.g.
- Procedure calls in the same platform in a mixed-language environment, e.g., calling C or Fortran from Java.

The requirements for these might be different.
How internal is internal?

Is it feasible or meaningful to maintain a distinct internal and external format?

What representational optimizations are available to the compiler?

- Is explicitly maintaining two formats too much load for the programmer?
- Do you need to force memory format when the programmer applies the ‘address of’ operator in C?
- Is a conversion to memory format compulsory when spilling intermediate results?

In general, what’s visible in the architectural state, and how does it depend on the platform?
Proposed terminology

Computational — invisible to ordinary programs (example: binary coefficient and exponent in separate registers)

Internal — default format stored in memory (example: binary coefficient and exponent packed in 64-bit integer)

Memory — used for inter-language calls (example: GBD or DPD)

Exchange — used for external representation (example: strings)

We probably don’t want these to be all different, but in principle it’s possible.
It seems important to have at least some standard exchange format.

Strings: a standard exchange format?
Should we standardize anything else?

Arguably, standardizing the other formats is much less important. Using a common hardware/software layer to implement decimal, no cross-language compatibility issues should arise. No ordinary user writing modular code would ever notice the internal representation. Exceptions: perhaps overall size and alignment requirements.

Explicit manipulation of bit patterns is not something we want to encourage — we should treat data types more abstractly. No ordinary user writing modular code would ever notice the internal representation. Exceptions: perhaps overall size and alignment requirements.