Various notions of “format”

John Harrison and Peter Tang

IEEE 754 revision meeting

April 21, 2005

Summary

> 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”.

Questions

We *might* specify some formats but allow others to be implementation-dependent, e.g

- An *exchange*/*external*/*memory*/*storage* format
- An *internal*/*computational*/*evaluation*/*register* format.

This raises two obvious questions:

- What do these terms mean?
- What should we standardize and what not?

Notions of ‘exchange’

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. front-end clients talking to back-end mainframe.
- Procedure calls on 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: GBCD 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.

Strings: a standard exchange format?

It seems important to have at least some standard exchange format. How about strings?

- Intuitive canonical representation is possible (in binary this is more problematic)
- Compatible with existing practice
- Variable-length encoding may be quite compact in the common case (many short numbers)

On the negative side, the variable-length feature could be inconvenient and the worst-case space usage is poor.

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.

Explicit maniplation 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.