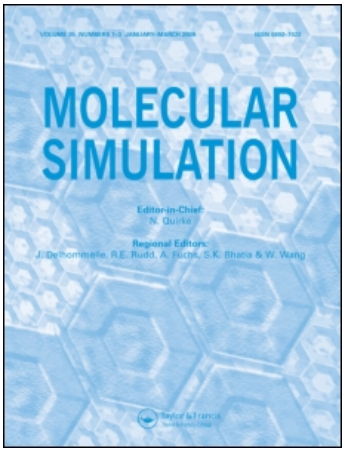

Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

Minimum Image Convention Coding of Microcomputers

Ulrich K. Deiters[a]
[a] Lehrstuhl für Physikalische Chemie II, Ruhr-Universität Bochum, Bochum 1, FRG

To cite this Article Deiters, Ulrich K.(1989) 'Minimum Image Convention Coding of Microcomputers', Molecular Simulation, 3: 5, 343 — 344
To link to this Article: DOI: 10.1080/08927028908031386
URL: http://dx.doi.org/10.1080/08927028908031386

Molecular Simulation, 1989, Vol. 3, pp. 343–344
Reprints available directly from the publisher
Photocopying permitted by license only

Note

MINIMUM IMAGE CONVENTION CODING OF MICROCOMPUTERS

ULRICH K. DEITERS

Lehrstuhl für Physikalische Chemie II, Ruhr-Universität Bochum, D-4630 Bochum 1, FRG

(*Received January 1989, accepted February 1989*)

KEY WORDS: Minimum image convention, microcomputers

Whenever computer simulations on fluids are performed with periodic (cubic) boundary conditions, one of the most frequently performed numerical operations is the calculation of the distance between a given molecule and the nearest image of a neighbour particle. Often this is accomplished by a computer code like this:

```
BOX:   box length
RA(3): coordinates of particle A
RB(3): coordinates of particle B
DIST2: square of effective distance between A and B

      DIST2 = 0.0
      DO 100 I = 1, 3
         DC = RA(I) - RB(I)
         DIST2 = DIST2 + (DC - BOX * ANINT(DC/BOX))/**2
  100 CONTINUE
```

On microcomputers containing the Motorola coprocessor MC68881, the lengthy Fortran expression (DC − BOX * ANINT(DC/BOX)) can be translated into the single assembler instruction FREM (floating point remainder [1]). However, most Fortran compilers are not aware of this possibility; instead, they generate code that performs a division, a jump to a Fortran intrinsic function, a multiplication, and a subtraction. This way of coding not only involves a larger number of floating point operations, but also requires the saving and restoring of address registers and interferes with code optimization.

In order to implement the FREM instruction, it is not necessary to do extensive assembler programming. Some Fortran compilers have the option of generating readable assembler code from a Fortran source. In this case it is relatively easy to identify and then to edit the piece of code corresponding to the above set of Fortran statements. The allocation of stack registers to Fortran variables can be read off the symbol table. As the compiler provides the user with a correct – although not optimal

– assembler code, it is neither necessary nor desirable to change any memory allocations or to interfere with address arithmetics. Therefore the hazardous part of assembler programming is avoided.

In addition to replacing a sequence of assembler instructions by a single FREM instruction, it is also possible to eliminate a number of stack save/get instructions which had been necessary because of the intrinsic function call and which have now become obsolete. It must be noted that the coprocessor needs more time to access a stack register than to access one of its internal registers, because stack calls also imply a data type conversion.

Performing the "operation" described above on the central subroutine of a Monte Carlo program (mixtures of hard non-spherical molecules) led to a reduction of the CPU time consumption by more than a factor of 3 on a Hewlett-Packard UNIX workstation HP9000/318M using Fortran 77 version 6.2.

Reference

[1] Motorola Inc., "MC68881 Floating-Point Coprocessor User's Manual", 1st edition, 1985, p. 3–87.