Loop Optimizations - When Interchange Won't Work*

Charles Severance
Kevin Dowd

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 3.0†

In the matrix multiplication code, we encountered a non-unit stride and were able to eliminate it with a quick interchange of the loops. Unfortunately, life is rarely this simple. Often you find some mix of variables with unit and non-unit strides, in which case interchanging the loops moves the damage around, but doesn't make it go away.

The loop to perform a matrix transpose represents a simple example of this dilemma:

```
DO I=1,N
  DO J=1,M
    A(J,I) = B(I,J)
  ENDDO
ENDDO
```

```
DO 20 J=1,M
  DO 10 I=1,N
       A(J,I) = B(I,J)
  ENDDO
ENDDO
```

Whichever way you interchange them, you will break the memory access pattern for either A or B. Even more interesting, you have to make a choice between strided loads vs. strided stores: which will it be?[1] We really need a general method for improving the memory access patterns for *both*`A` and `B`, not one or the other. We'll show you such a method in .

*Version 1.3: Aug 25, 2010 11:02 am -0500

†http://creativecommons.org/licenses/by/3.0/

[1] I can't tell you which is the better way to cast it; it depends on the brand of computer. Some perform better with the loops left as they are, sometimes by more than a factor of two. Others perform better with them interchanged. The difference is in the way the processor handles updates of main memory from cache.