

Assignment 10: DGEMM Kernel Tuning, PowerPC 970FX

Due: Monday 04/18/05, Multiplication Factor: 4.0

In this assignment, we will be optimizing the double precision matmul kernel for the PPC970FX. We will assume $\alpha = \beta = 1.0$ for the kernel, and the function name will be `ATL_USERMM`. For more details, see the example kernel on animal.cs.fsu.edu at:

```
~rwhaley/teach/cis5930/ASG10/mmkern_g5.c
```

To get the files you need, perform the following on `drteeth.cs.fsu.edu`:

```
cp ~rwhaley/teach/cis5930/ASG10/Makefile .
make archdirs
```

I have provided a tester and timer for this routine which can be built and run by:

```
make [test,time] nb=<block factor>
```

Sometimes it helps to optimized the function computationally first, and then add the memory optimizations once the computation runs at peak rate assuming things are in cache. The timer supports this. To make the timer give you in-cache results, choose an NB small enough to fit, and tell it not to move the arrays around, e.g.:

```
make time nb=36 moves=""
```

You can also test/time differing filenames. If you have a file saved in `../bob.c`, you would test/time it by suffixing `urout=../bob.c` to the appropriate make command. You can change between CPU and walltimes by varying the `TIME` make definition (an empty `TIME` gives you CPU).

You may want to optimize the loop computationally with in-cache data, and then improve it further using prefetch, where you time everything in the default (out-of-cache) way. I have provided you with an include file that provides you with the basic prefetch commands. To use it, simply `#include "atlas_prefetch.h"` at the top of your file. I have defined three basic prefetch functions: `ATL_pf11R` (prefetch 1 line for read), `ATL_pf11W` (prefetch 1 line for write), and `ATL_pfST` (prefetch a stream for read). Remember that a cache line is 128 bytes on this architecture. If you want to use more advanced prefetch commands, feel free to add your own in-line cpp macros at the top of your `mmkern_g5.c` file.

Here are some things you can assume about your kernel and its operands:

1. The A matrix is in transpose format, with size $KB \times MB$, and $lda = KB$. The next block of A (usually to be called in next kernel invocation) is at position $A+KB*MB$.
2. The B matrix is in notrans format, with size $KB \times NB$, and $ldb = KB$. The next block of B (usually to be called in next kernel invocation) is at position $B+KB*NB$.
3. The C matrix has an unconstrained ldc.
4. You must use square blocking factors, $(M_B = K_B = N_B) \leq 80$.

You should e-mail your file `mmkern.g5.c` to `whaley@cs.fsu.edu` by 10AM on the due date. If you use different compiler flags than I did, be sure to provide them in the body of the message.