Assignment 1: Bone-headed Timer

Due: 08/28/18, Multiplication Factor: 1.0

All our assignments will require you to log in remotely to various machines. The first assignment will be developed on `burrow.sice.indiana.edu`. All common OSes have methods of getting remote terminal access:

- **Linux/OS X**: use **ssh**. Open up a terminal window, and type:
  `ssh -X username@burrow.sice.indiana.edu`
- **Windows**: install and use **PuTTY** from the course website.

Once logged in, get the tarfile you need by, and then untar it by issuing following two commands in the directory you want the files to appear::

```
wget http://homes.sice.indiana.edu/rcwhaley/teach/iseHPO_F18/ASG/asg01.tar.bz2
bunzip2 -c asg01.tar.bz2 | tar xvmf -
```

Note that you can work on this assignment another machine, but make sure you test it with the standard Makefile on `burrow`, since that is how I will be grading.

Your first assignment is to complete a simple timer for a dot product kernel (the kernel can be found in `ddot.c`). Note that dot produce does $2N$ flops (1 add and 1 multiply for each element in N-length vector).

Examine the provided Makefile, and understand what it is doing. I have provided a timer skeleton for you to start from in dottime.c; all your changes must be written to this file, and this is the only file you will submit for this assignment. Read and understand the timer skeleton.

At the top of every assignment, there should be a comment that minimally identifies you by name, describes the assignment, and has any general comments that you want the grader to see.

You will submit the completed `dottime.c` on canvas. Note that you can keep resubmitting right up to the deadline without penalty, so once you get something working, submit it, and then submit each improvement, so even if you are working near the deadline and have network issues, you have something turned in.

I expect the submitted `dottime.c` to work in the framework as given, so change only this file for correctness. I have filled in the basics of a crude timer for you, and you must complete two routines:

1. `my_time()`: First, add either wall or CPU timer, whichever is easiest to understand for you, then add the other once the whole timer works. Both timers must work like a stop watch, in that the first call returns an arbitrary number, which is subtracted from a later invocation to give elapsed time. See the file for further details.
2. `DoTime()`: This routine does the actual timing of ddot(). I have left in my variable and function declarations, feel free to change (you do not have to use any statements internal to DoTime: write however you like as long as it has the required functional-

ity). Make sure your timer functions as discussed. In particular, it must repeat the operation in order to perform at least `mflop` MFLOPS, and should return the time (on average) of a single call to ddot().

Useful information can be found by:

```
man [getrusage,gettimeofday,uname]     # general info
cat /proc/cpuinfo                      # linux
```

Run the timer on `burrow` using both CPU and WALL time. Do you understand the output? What does/does not make sense to you? Do the timings make sense, and can you make them repeatable? Do you see differences among the timers, and do such differences depend on the setting of mflop? Are there differences amongst platforms/compilers, and do they act like you would expect? Can you vary the results in a meaningful way by changing the compiler flags, or adding a sleep call to ddot?

Part of this assignment is to familiarize you with how we'll be doing things, so be sure to fully understand the build process, and what all the files are doing.