

## Assignment 3: Achieve Peak

**Due: Monday 01/31/05, Multiplication Factor: 1.0**

You should attempt to get as close to theoretical floating point peak as you can using `gcc` on `program` (UltraSPARC). Your code can count only floating point computations, but the computations need not be useful (i.e., you do not have to achieve peak doing something like `ddot`). Because your computations may not be used, you will probably need to use the tricks we have discussed to ensure that your results are not optimized away. You may want to examine the generated assembly to see that all computations in your loop are actually present in the assembly to ensure that this hasn't happened (note that getting better than peak is one of the more obvious signs this has occurred). You should use your knowledge of architecture to choose the appropriate computations to perform, as well as how many will be needed in a instruction window. You should not be afraid of very heavy loop unrollings. Your timer should take one optional parameter (default value 200), `mflop`, which is used as in previous timings. You must repeat your basic computational loop enough times to perform at least `mflop` megaflops. A skeleton of the simple timer required is shown below.

Your compiler must be `gcc`, and you must use the flags as in:

```
gcc -mcpu=ultrasparc -mtune=ultrasparc -fomit-frame-pointer -O3 \  
-o xpeak peak.c
```

You can split the work into multiple files if you find it necessary, though I did not. My own code achieved almost 98% of peak.

You will send the complete file(s) to `whaley@cs.fsu.edu` by 10AM by the due date, and I expect it to work with the above style of compilation. Your baseline grade for this assignment will be the % you achieve compared to the fastest available implementation *when the flops you actually perform are used to correctly calculate your achieved flop rate*.

After you get it working with `gcc`, can you improve performance using the sun compiler? If so, why (looking at assembly can help here)? My own performance went to almost 99% with the compile:

```
/opt/SUNWspro/bin/cc -O -o xtst peak.c
```

What affect do differing compiler flags have on each, and is higher optimization always better? What affect does heavy unrolling have on compile time, and does this also vary with flags?

You can generate an assembly file (`peak.s` for both of the below) by:

```
gcc -mcpu=ultrasparc -mtune=ultrasparc -fomit-frame-pointer -O3 -S peak.c \  
/opt/SUNWspro/bin/cc -O -S peak.c
```

```

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <sys/time.h>

#define my_time gethrtime
double ClickToSec(long long clicks)
{
    return(clicks*1.0e-9);
}

double GetMflop(int mflop, double *dum)
{
    const int lflops = XXXXX; /* flops performed in your loop */
    long long t0, t1;
    int nrep, i;

    nrep = (mflop * 1000000 + lflops-1)/lflops;
    if (nrep<1) nrep=1;

    t0 = my_time();
    for (i=nrep; i; i--)
    {
    }
    t1 = my_time();
    return(nrep*1.0e-6*lflops/ClickToSec(t1-t0));
}

main(int nargs, char **args)
{
    int mflop = 200, i;
    double dum[16];
    for (i=0; i < 16; i++) dum[i] = 0.0;
    if (nargs > 1) mflop = atoi(args[1]);
    printf("Peak mflop = %.2f\n", GetMflop(mflop, dum));
}

```