I wrote a program called flops.c that solves a system of linear equations $Ax = b$. Note, the solution obtained by Gaussian elimination satisfies

$$b_i = \sum_{j=1}^{n} A_{ij} x_j.$$

The program was this:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "tictoc.h"

void scale(int n,double v[],double alpha){
    for(int i=0;i<n;i++){
        v[i]*=alpha;
    }
}
void eliminate(int n,double v1[],
    double v2[],double alpha){
    for(int i=0;i<n;i++){
        v1[i]+=alpha*v2[i];
    }
}
void swap(int n,double v1[],double v2[]){
    for(int i=0;i<n;i++){
        double t=v1[i]; v1[i]=v2[i]; v2[i]=t;
    }
}
void matprint(int n,double A[n][n]){
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            printf("%g ",A[i][j]);
        }
        printf("\n");
    }
}
//  Factor A into PLU where L is lower triangular with 1's on
//  the diagonal and U is upper triangular.  Return the result
//  in A where the upper triangular part of A is now U and the
//  part of A below the diagonal holds L.
void lufact(int n,double A[n][n],int P[n]){
    for(int j=0;j<n;j++){
        P[j]=j;
    }
    for(int j=0;j<n-1;j++){
        //  Seach for the largest pivot in the column
```

```
        int i0=j;
        for(int i=j+1;i<n;i++){
            if(fabs(A[i][j])>fabs(A[i0][j])){
                i0=i;
            }
        }
        swap(n,A[j],A[i0]);
        int t=P[j]; P[j]=P[i0]; P[i0]=t;
        for(int i=j+1;i<n;i++){
            double alpha=-A[i][j]/A[j][j];
            eliminate(n-j-1,A[i]+j+1,A[j]+j+1,alpha);
            A[i][j]=-alpha;
        }
    }
}
//  Solve PLUx=b for x where LU is the matrix returned by lufact.
void lusolve(int n,int P[n],double LU[n][n],double x[n],double b[n]){
    double y[n];
    //  Forward substitute to solve Ly=b
    for(int i=0;i<n;i++){
        y[i]=b[P[i]];
        for(int j=0;j<i;j++){
            y[i]-=LU[i][j]*y[j];
        }
    }
    //  Backward substitute to solve Ux=y
    for(int i=n-1;i>=0;i--){
        x[i]=y[i];
        for(int j=i+1;j<n;j++){
            x[i]-=LU[i][j]*x[j];
        }
        x[i]/=LU[i][i];
    }
}
//  Compute b where b=Ax.
void multAb(int n,double b[n],double A[n][n],double x[n]){
    for(int i=0;i<n;i++){
        b[i]=0;
        for(int j=0;j<n;j++){
            b[i]+=A[i][j]*x[j];
        }
    }
}
//  Compute the norm |x-y|.
double dist(int n,double x[n],double y[n]){
    double s=0.0;
    for(int i=0;i<n;i++){
        double t=x[i]-y[i];
        s+=t*t;
```

```
90      }
91      return sqrt(s);
92  }
93
94  #define N 1000
95  double A[N][N];
96  double b[N];
97  double x[N],LU[N][N],b2[N];
98  int P[N];
99
100 int main(){
101     for(int i=0;i<N;i++){
102         for(int j=0;j<N;j++){
103             A[i][j]=2.0*random()/RAND_MAX-1.0;
104         }
105         b[i]=2.0*random()/RAND_MAX-1.0;
106     }
107     memcpy(LU,A,N*N*sizeof(double));
108     tic();
109     lufact(N,LU,P);
110     double tfact=toc();
111     tic();
112     lusolve(N,P,LU,x,b);
113     double tsolve=toc();
114     multAb(N,b2,A,x);
115     printf("N=%d\n",N);
116     printf("error=%.15e\n",dist(N,b,b2));
117     printf("tfact=%g sec\n",tfact);
118     printf("tsolve=%g sec\n",tsolve);
119     printf("total time=%g sec\n",tfact+tsolve);
120         printf("FLOPS=%g\n",2.0*(N/3.0+1.0)*N*N/(tfact+tsolve));
121     return 0;
122 }
123
```

The output was this:

```
N=1000
error=2.119188909422022e-11
tfact=0.633473 sec
tsolve=0.00241295 sec
total time=0.635886 sec
FLOPS=1.05155e+09
```