```
/*-----------------------------------------------------------------------*/
/* Skeleton 2D Electrostatic Vector PIC code */
/* written by Viktor K. Decyk, UCLA and Ricardo Fonseca, ISCTE */
#include <stdlib.h>
#include <stdio.h>
#include <complex.h>
#include <sys/time.h>
#include "vpush2.h"
#include "sselib2.h"
#include "ssepush2.h"

void dtimer(double *time, struct timeval *itime, int icntrl);

int main(int argc, char *argv[]) {
/* indx/indy = exponent which determines grid points in x/y direction: */
/* nx = 2**indx, ny = 2**indy */
   int indx =   9, indy =   9;
/* npx/npy = number of electrons distributed in x/y direction */
   int npx =  3072, npy =   3072;
/* ndim = number of velocity coordinates = 2 */
   int ndim = 2;
/* tend = time at end of simulation, in units of plasma frequency */
/* dt = time interval between successive calculations */
/* qme = charge on electron, in units of e */
   float tend = 10.0, dt = 0.1, qme = -1.0;
/* vtx/vty = thermal velocity of electrons in x/y direction */
/* vx0/vy0 = drift velocity of electrons in x/y direction */
   float vtx = 1.0, vty = 1.0, vx0 = 0.0, vy0 = 0.0;
/* ax/ay = smoothed particle size in x/y direction */
   float ax = .912871, ay = .912871;
/* idimp = number of particle coordinates = 4 */
/* ipbc = particle boundary condition: 1 = periodic */
/* sortime = number of time steps between standard electron sorting */
   int idimp = 4, ipbc = 1, sortime = 50;
/* wke/we/wt = particle kinetic/electric field/total energy */
   float wke = 0.0, we = 0.0, wt = 0.0;
/* kvec = (1,2) = run (autovector,SSE2) version */
   int kvec = 1;

/* declare scalars for standard code */
   int j;
   int np, nx, ny, nxh, nyh, nxe, nye, nxeh, nxyh, nxhy;
   int npe, ny1, ntime, nloop, isign;
   int irc = 0;
   float qbme, affp;

/* declare arrays for standard code: */
/* partt, partt2 = transposed particle arrays */
   float *partt = NULL, *partt2 = NULL, *tpartt = NULL;
/* qe = electron charge density with guard cells */
   float *qe = NULL;
/* fxye = smoothed electric field with guard cells */
   float *fxye = NULL;
/* ffc = form factor array for poisson solver */
```

```
   float complex *ffc = NULL;
/* mixup = bit reverse table for FFT */
   int *mixup = NULL;
/* sct = sine/cosine table for FFT */
   float complex *sct = NULL;
/* npicy = scratch array for reordering particles */
   int *npicy = NULL;

/* declare and initialize timing data */
   float time;
   struct timeval itime;
   float tdpost = 0.0, tguard = 0.0, tfft = 0.0, tfield = 0.0;
   float tpush = 0.0, tsort = 0.0;
   double dtime;

/* initialize scalars for standard code */
/* np = total number of particles in simulation */
/* nx/ny = number of grid points in x/y direction */
   np = npx*npy; nx = 1L<<indx; ny = 1L<<indy; nxh = nx/2; nyh = ny/2;
   nxe = nx + 2; nye = ny + 1; nxeh = nxe/2;
   nxyh = (nx > ny ? nx : ny)/2; nxhy = nxh > ny ? nxh : ny;
   ny1 = ny + 1;
/* nloop = number of time steps in simulation */
/* ntime = current time step */
   nloop = tend/dt + .0001; ntime = 0;
   qbme = qme;
   affp = (float) (nx*ny)/(float ) np;

/* allocate data for standard code */
   mixup = (int *) malloc(nxhy*sizeof(int));
   sct = (float complex *) malloc(nxyh*sizeof(float complex));

/* align memory for SSE */
   npe = 4*((np - 1)/4 + 1);
   nxe = 4*((nxe - 1)/4 + 1);
   nxeh = nxe/2;
   sse_fallocate(&partt,npe*idimp,&irc);
   if (sortime > 0)
      sse_fallocate(&partt2,npe*idimp,&irc);
   sse_fallocate(&qe,nxe*nye,&irc);
   sse_fallocate(&fxye,ndim*nxe*nye,&irc);
   sse_callocate(&ffc,nxh*nyh,&irc);
   sse_iallocate(&npicy,ny1,&irc);
   if (irc != 0) {
      printf("aligned allocation error: irc = %d\n",irc);
   }

/* prepare fft tables */
   cwfft2rinit(mixup,sct,indx,indy,nxhy,nxyh);
/* calculate form factors */
   isign = 0;
   cvpois22((float complex *)qe,(float complex *)fxye,isign,ffc,ax,ay,
            affp,&we,nx,ny,nxeh,nye,nxh,nyh);
/* initialize electrons */
```

```
   cdistr2t(partt,vtx,vty,vx0,vy0,npx,npy,idimp,npe,nx,ny,ipbc);

/* * * * start main iteration loop * * * */

L500: if (nloop <= ntime)
         goto L2000;
/*    printf("ntime = %i\n",ntime); */

/* deposit charge with standard procedure: updates qe */
      dtimer(&dtime,&itime,-1);
      for (j = 0; j < nxe*nye; j++) {
         qe[j] = 0.0;
      }
      if (kvec==1)
         cvgpost2lt(partt,qe,qme,np,npe,idimp,nxe,nye);
/* SSE2 function */
      else if (kvec==2)
         csse2gpost2lt(partt,qe,qme,np,npe,idimp,nxe,nye);
      dtimer(&dtime,&itime,1);
      time = (float) dtime;
      tdpost += time;

/* add guard cells with standard procedure: updates qe */
      dtimer(&dtime,&itime,-1);
      if (kvec==1)
         caguard2l(qe,nx,ny,nxe,nye);
/* SSE2 function */
      else if (kvec==2)
         csse2aguard2l(qe,nx,ny,nxe,nye);
      dtimer(&dtime,&itime,1);
      time = (float) dtime;
      tguard += time;

/* transform charge to fourier space with standard procedure: updates qe */
      dtimer(&dtime,&itime,-1);
      isign = -1;
      if (kvec==1)
         cwfft2rvx((float complex *)qe,isign,mixup,sct,indx,indy,nxeh,
                   nye,nxhy,nxyh);
/* SSE2 function */
      else if (kvec==2)
         csse2wfft2rx((float complex *)qe,isign,mixup,sct,indx,indy,
                      nxeh,nye,nxhy,nxyh);
      dtimer(&dtime,&itime,1);
      time = (float) dtime;
      tfft += time;

/* calculate force/charge in fourier space with standard procedure: */
/* updates fxye                                                     */
      dtimer(&dtime,&itime,-1);
      isign = -1;
      if (kvec==1)
         cvpois22((float complex *)qe,(float complex *)fxye,isign,ffc,
                  ax,ay,affp,&we,nx,ny,nxeh,nye,nxh,nyh);
```

```
/* SSE2 function */
      else if (kvec==2)
         csse2pois22((float complex *)qe,(float complex *)fxye,isign,
                     ffc,ax,ay,affp,&we,nx,ny,nxeh,nye,nxh,nyh);
      dtimer(&dtime,&itime,1);
      time = (float) dtime;
      tfield += time;

/* transform force to real space with standard procedure: updates fxye */
      dtimer(&dtime,&itime,-1);
      isign = 1;
      if (kvec==1)
         cwfft2rv2((float complex *)fxye,isign,mixup,sct,indx,indy,nxeh,
                   nye,nxhy,nxyh);
/* SSE2 function */
      else if (kvec==2)
         csse2wfft2r2((float complex *)fxye,isign,mixup,sct,indx,indy,
                      nxeh,nye,nxhy,nxyh);
      dtimer(&dtime,&itime,1);
      time = (float) dtime;
      tfft += time;

/* copy guard cells with standard procedure: updates fxye */
      dtimer(&dtime,&itime,-1);
      if (kvec==1)
         ccguard2l(fxye,nx,ny,nxe,nye);
/* SSE2 function */
      else if (kvec==2)
         csse2cguard2l(fxye,nx,ny,nxe,nye);
      dtimer(&dtime,&itime,1);
      time = (float) dtime;
      tguard += time;

/* push particles with standard procedure: updates part, wke */
      wke = 0.0;
      dtimer(&dtime,&itime,-1);
      if (kvec==1)
         cvgpush2lt(partt,fxye,qbme,dt,&wke,idimp,np,npe,nx,ny,nxe,nye,
                    ipbc);
/* SSE2 function */
      else if (kvec==2)
         csse2gpush2lt(partt,fxye,qbme,dt,&wke,idimp,np,npe,nx,ny,nxe,
                       nye,ipbc);
      dtimer(&dtime,&itime,1);
      time = (float) dtime;
      tpush += time;

/* sort particles by cell for standard procedure */
      if (sortime > 0) {
         if (ntime%sortime==0) {
            dtimer(&dtime,&itime,-1);
            if (kvec==1)
               cdsortp2ylt(partt,partt2,npicy,idimp,np,npe,ny1);
/* SSE2 function */
```

```
            else if (kvec==2)
               csse2dsortp2ylt(partt,partt2,npicy,idimp,np,npe,ny1);
/* exchange pointers */
            tpartt = partt;
            partt = partt2;
            partt2 = tpartt;
            dtimer(&dtime,&itime,1);
            time = (float) dtime;
            tsort += time;
         }
      }

      if (ntime==0) {
         printf("Initial Field, Kinetic and Total Energies:\n");
         printf("%e %e %e\n",we,wke,wke+we);
      }
      ntime += 1;
      goto L500;
L2000:

/* * * * end main iteration loop * * * */

   printf("ntime = %i, kvec = %i\n",ntime,kvec);
   printf("Final Field, Kinetic and Total Energies:\n");
   printf("%e %e %e\n",we,wke,wke+we);

   printf("\n");
   printf("deposit time = %f\n",tdpost);
   printf("guard time = %f\n",tguard);
   printf("solver time = %f\n",tfield);
   printf("fft time = %f\n",tfft);
   printf("push time = %f\n",tpush);
   printf("sort time = %f\n",tsort);
   tfield += tguard + tfft;
   printf("total solver time = %f\n",tfield);
   time = tdpost + tpush + tsort;
   printf("total particle time = %f\n",time);
   wt = time + tfield;
   printf("total time = %f\n",wt);
   printf("\n");

   wt = 1.0e+09/(((float) nloop)*((float) np));
   printf("Push Time (nsec) = %f\n",tpush*wt);
   printf("Deposit Time (nsec) = %f\n",tdpost*wt);
   printf("Sort Time (nsec) = %f\n",tsort*wt);
   printf("Total Particle Time (nsec) = %f\n",time*wt);

   sse_deallocate(npicy);
   sse_deallocate(ffc);
   sse_deallocate(fxye);
   sse_deallocate(qe);
   if (sortime > 0)
      sse_deallocate(partt2);
   sse_deallocate(partt);
```

```
    return 0;
}
```