```
!-----------------------------------------------------------------------
! Skeleton 2-1/2D Electromagnetic OpenMP PIC code
! written by Viktor K. Decyk, UCLA
      program mbpic2
      use mbpush2_h
      use omplib_h
      implicit none
      integer, parameter :: indx =   9, indy =   9
      integer, parameter :: npx =  3072, npy =   3072
      integer, parameter :: ndim = 3
      real, parameter :: tend = 10.0, dt = 0.04, qme = -1.0
      real, parameter :: vtx = 1.0, vty = 1.0, vx0 = 0.0, vy0 = 0.0
      real, parameter :: vtz = 1.0, vz0 = 0.0
      real :: ax = .912871, ay = .912871, ci = 0.1
! idimp = dimension of phase space = 5
! relativity = (no,yes) = (0,1) = relativity is used
      integer :: idimp = 5, ipbc = 1, relativity = 1
      real :: wke = 0.0, we = 0.0, wf = 0.0, wm = 0.0, wt = 0.0
! sorting tiles, should be less than or equal to 32
      integer :: mx = 16, my = 16
! fraction of extra particles needed for particle management
      real :: xtras = 0.2
! declare scalars for standard code
      integer :: np, nx, ny, nxh, nyh, nxe, nye, nxeh, nxyh, nxhy
      integer :: mx1, my1, mxy1, ntime, nloop, isign
      real :: qbme, affp, dth
!
! declare scalars for OpenMP code
      integer :: nppmx, nppmx0, ntmax, npbmx, irc
      integer :: nvp
!
! declare arrays for standard code
      real, dimension(:,:), pointer :: part
      real, dimension(:,:), pointer :: qe
      real, dimension(:,:,:), pointer :: cue, fxyze, bxyze
      complex, dimension(:,:,:), pointer :: exyz, bxyz
      complex, dimension(:,:), pointer :: ffc
      integer, dimension(:), pointer :: mixup
      complex, dimension(:), pointer :: sct
!
! declare arrays for OpenMP (tiled) code
      real, dimension(:,:,:), pointer :: ppart, ppbuff
      integer, dimension(:), pointer :: kpic
      integer, dimension(:,:), pointer :: ncl
      integer, dimension(:,:,:), pointer :: ihole
!
! declare and initialize timing data
      real :: time
      integer, dimension(4) :: itime
      real :: tdpost = 0.0, tguard = 0.0, tfft = 0.0, tfield = 0.0
      real :: tdjpost = 0.0, tpush = 0.0, tsort = 0.0
      double precision :: dtime
!
      irc = 0
```

```
! nvp = number of shared memory nodes (0=default)
      nvp = 0
!     write (*,*) 'enter number of nodes:'
!     read (5,*) nvp
! initialize for shared memory parallel processing
      call INIT_OMP(nvp)
!
! initialize scalars for standard code
      np = npx*npy; nx = 2**indx; ny = 2**indy; nxh = nx/2; nyh = ny/2
      nxe = nx + 2; nye = ny + 1; nxeh = nxe/2
      nxyh = max(nx,ny)/2; nxhy = max(nxh,ny)
      mx1 = (nx - 1)/mx + 1; my1 = (ny - 1)/my + 1; mxy1 = mx1*my1
      nloop = tend/dt + .0001; ntime = 0
      qbme = qme
      affp = real(nx*ny)/real(np)
      dth = 0.0
!
! allocate and initialize data for standard code
      allocate(part(idimp,np))
      allocate(qe(nxe,nye),fxyze(ndim,nxe,nye))
      allocate(cue(ndim,nxe,nye),bxyze(ndim,nxe,nye))
      allocate(exyz(ndim,nxeh,nye),bxyz(ndim,nxeh,nye))
      allocate(ffc(nxh,nyh),mixup(nxhy),sct(nxyh))
      allocate(kpic(mxy1))
!
! prepare fft tables
      call WFFT2RINIT(mixup,sct,indx,indy,nxhy,nxyh)
! calculate form factors
      isign = 0
      call MPOIS23(qe,fxyze,isign,ffc,ax,ay,affp,we,nx,ny,nxeh,nye,nxh, &
     &nyh)
! initialize electrons
      call DISTR2H(part,vtx,vty,vtz,vx0,vy0,vz0,npx,npy,idimp,np,nx,ny, &
     &ipbc)
!
! initialize transverse electromagnetic fields
      exyz = cmplx(0.0,0.0)
      bxyz = cmplx(0.0,0.0)
!
! find number of particles in each of mx, my tiles: updates kpic, nppmx
      call DBLKP2L(part,kpic,nppmx,idimp,np,mx,my,mx1,mxy1,irc)
      if (irc /= 0) then
         write (*,*) 'DBLKP2L error, irc=', irc
         stop
      endif
! allocate vector particle data
      nppmx0 = (1.0 + xtras)*nppmx
      ntmax = xtras*nppmx
      npbmx = xtras*nppmx
      allocate(ppart(idimp,nppmx0,mxy1))
      allocate(ppbuff(idimp,npbmx,mxy1))
      allocate(ncl(8,mxy1))
      allocate(ihole(2,ntmax+1,mxy1))
! copy ordered particle data for OpenMP
```

```
      call PPMOVIN2L(part,ppart,kpic,nppmx0,idimp,np,mx,my,mx1,mxy1,irc)
      if (irc /= 0) then
         write (*,*) 'PPMOVIN2L overflow error, irc=', irc
         stop
      endif
! sanity check
      call PPCHECK2L(ppart,kpic,idimp,nppmx0,nx,ny,mx,my,mx1,my1,irc)
      if (irc /= 0) then
         write (*,*) 'PPCHECK2L error: irc=', irc
         stop
      endif
!
      if (dt > 0.45*ci) then
         write (*,*) 'Warning: Courant condition may be exceeded!'
      endif
!
! * * * start main iteration loop * * *
!
  500 if (nloop <= ntime) go to 2000
!     write (*,*) 'ntime = ', ntime
!
! deposit current with OpenMP: updates ppart, cue, ncl, ihole, irc
      call dtimer(dtime,itime,-1)
      cue = 0.0
      if (relativity==1) then
         call GRJPPOSTF2L(ppart,cue,kpic,ncl,ihole,qme,dth,ci,nppmx0,   &
     &idimp,nx,ny,mx,my,nxe,nye,mx1,mxy1,ntmax,irc)
      else
         call GJPPOSTF2L(ppart,cue,kpic,ncl,ihole,qme,dth,nppmx0,idimp, &
     &nx,ny,mx,my,nxe,nye,mx1,mxy1,ntmax,irc)
      endif
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tdjpost = tdjpost + time
      if (irc /= 0) then
         if (relativity==1) then
            write (*,*) 'GRJPPOSTF2L error: irc=', irc
         else
            write (*,*) 'GJPPOSTF2L error: irc=', irc
         endif
         stop
      endif
!
! reorder particles by cell with OpenMP:
! updates ppart, ppbuff, kpic, ncl, and irc
      call dtimer(dtime,itime,-1)
      call PPORDERF2L(ppart,ppbuff,kpic,ncl,ihole,idimp,nppmx0,mx1,my1, &
     &npbmx,ntmax,irc)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tsort = tsort + time
      if (irc /= 0) then
         write (*,*) 'current PPORDERF2L error: ntmax, irc=', ntmax, irc
         stop
```

```
      endif
!
! deposit charge with OpenMP: updates qe
      call dtimer(dtime,itime,-1)
      qe = 0.0
      call GPPOST2L(ppart,qe,kpic,qme,nppmx0,idimp,mx,my,nxe,nye,mx1,   &
     &mxy1)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tdpost = tdpost + time
!
! add guard cells with OpenMP: updates cue, qe
      call dtimer(dtime,itime,-1)
      call ACGUARD2L(cue,nx,ny,nxe,nye)
      call AGUARD2L(qe,nx,ny,nxe,nye)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tguard = tguard + time
!
! transform charge to fourier space with OpenMP: updates qe
      call dtimer(dtime,itime,-1)
      isign = -1
      call WFFT2RMX(qe,isign,mixup,sct,indx,indy,nxeh,nye,nxhy,nxyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft = tfft + time
!
! transform current to fourier space with OpenMP: updates cue
      call dtimer(dtime,itime,-1)
      isign = -1
      call WFFT2RM3(cue,isign,mixup,sct,indx,indy,nxeh,nye,nxhy,nxyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft = tfft + time
!
! take transverse part of current with OpenMP: updates cue
      call dtimer(dtime,itime,-1)
      call MCUPERP2(cue,nx,ny,nxeh,nye)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfield = tfield + time
!
! calculate electromagnetic fields in fourier space with OpenMP:
! updates exyz, bxyz
      call dtimer(dtime,itime,-1)
      if (ntime==0) then
         call MIBPOIS23(cue,bxyz,ffc,ci,wm,nx,ny,nxeh,nye,nxh,nyh)
         wf = 0.0
         dth = 0.5*dt
      else
         call MMAXWEL2(exyz,bxyz,cue,ffc,ci,dt,wf,wm,nx,ny,nxeh,nye,nxh,&
     &nyh)
      endif
      call dtimer(dtime,itime,1)
```

```
      time = real(dtime)
      tfield = tfield + time
!
! calculate force/charge in fourier space OpenMP: updates fxyze
      call dtimer(dtime,itime,-1)
      isign = -1
      call MPOIS23(qe,fxyze,isign,ffc,ax,ay,affp,we,nx,ny,nxeh,nye,nxh, &
     &nyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfield = tfield + time
!
! add longitudinal and transverse electric fields with OpenMP:
! updates fxyze
      call dtimer(dtime,itime,-1)
      isign = 1
      call MEMFIELD2(fxyze,exyz,ffc,isign,nx,ny,nxeh,nye,nxh,nyh)
! copy magnetic field with OpenMP: updates bxyze
      isign = -1
      call MEMFIELD2(bxyze,bxyz,ffc,isign,nx,ny,nxeh,nye,nxh,nyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfield = tfield + time
!
! transform electric force to real space with OpenMP: updates fxyze
      call dtimer(dtime,itime,-1)
      isign = 1
      call WFFT2RM3(fxyze,isign,mixup,sct,indx,indy,nxeh,nye,nxhy,nxyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft = tfft + time
!
! transform magnetic force to real space with OpenMP: updates bxyze
      call dtimer(dtime,itime,-1)
      isign = 1
      call WFFT2RM3(bxyze,isign,mixup,sct,indx,indy,nxeh,nye,nxhy,nxyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft = tfft + time
!
! copy guard cells with OpenMP: updates fxyze, bxyze
      call dtimer(dtime,itime,-1)
      call BGUARD2L(fxyze,nx,ny,nxe,nye)
      call BGUARD2L(bxyze,nx,ny,nxe,nye)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tguard = tguard + time
!
! push particles with OpenMP: updates ppart, ncl, ihole, wke, irc
      wke = 0.0
      call dtimer(dtime,itime,-1)
      if (relativity==1) then
         call GRBPPUSHF23L(ppart,fxyze,bxyze,kpic,ncl,ihole,qbme,dt,dth,&
     &ci,wke,idimp,nppmx0,nx,ny,mx,my,nxe,nye,mx1,mxy1,ntmax,irc)
```

```
      else
         call GBPPUSHF23L(ppart,fxyze,bxyze,kpic,ncl,ihole,qbme,dt,dth, &
     &wke,idimp,nppmx0,nx,ny,mx,my,nxe,nye,mx1,mxy1,ntmax,irc)
      endif
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tpush = tpush + time
      if (irc /= 0) then
         if (relativity==1) then
            write (*,*) 'GRBPPUSHF23L error: irc=', irc
         else
            write (*,*) 'GBPPUSHF23L error: irc=', irc
         endif
         stop
      endif
!
! reorder particles by cell with OpenMP:
! updates ppart, ppbuff, kpic, ncl, and irc
      call dtimer(dtime,itime,-1)
      call PPORDERF2L(ppart,ppbuff,kpic,ncl,ihole,idimp,nppmx0,mx1,my1, &
     &npbmx,ntmax,irc)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tsort = tsort + time
      if (irc /= 0) then
         write (*,*) 'push PPORDERF2L error: ntmax, irc=', ntmax, irc
         stop
      endif
!
      if (ntime==0) then
         wt = we + wf + wm
         write (*,*) 'Initial Total Field, Kinetic and Total Energies:'
         write (*,'(3e14.7)') wt, wke, wke + wt
         write (*,*) 'Initial Electrostatic, Transverse Electric and Mag&
     &netic Field Energies:'
         write (*,'(3e14.7)') we, wf, wm
      endif
      ntime = ntime + 1
      go to 500
 2000 continue
!
! * * * end main iteration loop * * *
!
      write (*,*) 'ntime, relativity = ', ntime, relativity
      wt = we + wf + wm
      write (*,*) 'Final Total Field, Kinetic and Total Energies:'
      write (*,'(3e14.7)') wt, wke, wke + wt
      write (*,*) 'Final Electrostatic, Transverse Electric and Magnetic&
     & Field Energies:'
      write (*,'(3e14.7)') we, wf, wm
!
      write (*,*)
      write (*,*) 'deposit time = ', tdpost
      write (*,*) 'current deposit time = ', tdjpost
```

```
      tdpost = tdpost + tdjpost
      write (*,*) 'total deposit time = ', tdpost
      write (*,*) 'guard time = ', tguard
      write (*,*) 'solver time = ', tfield
      write (*,*) 'fft time = ', tfft
      write (*,*) 'push time = ', tpush
      write (*,*) 'sort time = ', tsort
      tfield = tfield + tguard + tfft
      write (*,*) 'total solver time = ', tfield
      time = tdpost + tpush + tsort
      write (*,*) 'total particle time = ', time
      wt = time + tfield
      write (*,*) 'total time = ', wt
      write (*,*)
!
      wt = 1.0e+09/(real(nloop)*real(np))
      write (*,*) 'Push Time (nsec) = ', tpush*wt
      write (*,*) 'Deposit Time (nsec) = ', tdpost*wt
      write (*,*) 'Sort Time (nsec) = ', tsort*wt
      write (*,*) 'Total Particle Time (nsec) = ', time*wt
!
      stop
      end
```