OpenMP Fortran

1 Code

```
>>>
```

```
!$    TID = OMP_GET_THREAD_NUM();
      print *, 'In parallel region - Thread ID ',TID
!$OMP END PARALLEL
      end program main
```

>>>

```
      real(kind=8)   :: dummy
!-----------------------------------------------------------------------
! You can use this instead of the call below if your system supports it
!      call system("sleep 2")
!-----------------------------------------------------------------------
      call simulate_some_work(tid,dummy)
      return
      end subroutine work_while_wait
      subroutine do_work(tid)
      implicit none
      integer(kind=4):: tid
      real(kind=8)   :: dummy
      print 9010, tid
!-----------------------------------------------------------------------
! You can use this instead of the call below if your system supports it
!      call system("sleep 5")
!-----------------------------------------------------------------------
      call simulate_some_work(tid,dummy)
      print 9020, tid
      return
 9010 format(3X,'Thread ',I3,' has the lock and performs the work')
 9020 format(3X,'Thread ',I3,' has completed the work')
      end subroutine do_work
      subroutine simulate_some_work(tid,x)
      implicit none
      integer(kind=4):: tid
      real(kind=8)   :: x
      integer(kind=4), parameter:: incr = 10000000
      integer(kind=4)           :: i
      x = 1.0
      do i = 1+tid*incr, (tid+1)*incr
         x = cos(x/i) * sin(x/i)
      end do
      end subroutine simulate_some_work

>>>
```

```
      end do
      call mxv(n, a, b, c)
      print *,'After MXV:'
      print *,'Result array a is   :',a(1:n)
      print *,'Reference solution  :',ref(1:n)
      print *,'Difference a - ref  :',a(1:n)-ref(1:n)
      if (allocated(a)) deallocate(a)
      if (allocated(b)) deallocate(b)
      if (allocated(c)) deallocate(c)
      if (allocated(c)) deallocate(indx)
      end program array_reduction
      subroutine mxv(n, a, b, c)
      implicit none
      integer:: n, a(1:n), b(1:n,1:n), c(1:n)
      integer:: j
!$OMP PARALLEL WORKSHARE
      a(1:n) = 0.0
!$OMP END PARALLEL WORKSHARE
!$OMP PARALLEL DO DEFAULT(NONE) PRIVATE(j) SHARED(n,b,c) &
!$OMP          REDUCTION(+:a)
      do j = 1, n
         a(1:n) = a(1:n) + b(1:n,j)*c(j)
      end do
!$OMP END PARALLEL DO
      return
      end
```