# Barriers

- Points in program at which all threads have to arrive before any can proceed
- Provide *sequence control*

# Centralized Barrier

Shared variable: count = 0

Code for barrier for a given thread:

```
FetchAndAdd(count, 1)
while (count != numThreads) ;
```

Problem: ?

# Centralized Barrier

Shared variable: count = 0

Code for barrier for a given thread:

```
FetchAndAdd(count, 1)
while (count != numThreads) ;
```

Problem: reset

# Centralized Barrier, Suggested by Past Students

Shared variable: count = 0

Code for barrier for a given thread:

```
FetchAndAdd(count, 1)
while (count mod numThreads != 0) ;
```

Problem?

# Centralized Barrier, Suggested by Past Students, Version 2.0

Shared variable: count = 0

Code for barrier for a given thread:

```
if  (FetchAndAdd(count, 1) mod numThreads == 0)
   count = 0
else
   while (count != 0) ;
```

Problem?

# Centralized Barrier (handles reset)

Shared variables: countEven = countOdd = nB = 0

Code for barrier for a given thread:

```
if  (nB mod 2 == 0) {
   if  (FetchAndAdd(countEven, 1) == numThreads) {
      nB = nB + 1
      countEven = 0
   }
   else
      while (countEven != 0) ;
 else {
     // same code, but with countOdd
}
```

# Symmetric Barrier, 2 threads (not quite correct)

arrive[0] == arrive[1] == 0 initially

Thread 0's code

```
arrive[0] = 1
while (arrive[1] != 1)
   ;
arrive[1] = 0
```

Thread 1's code

```
arrive[1] = 1
while (arrive[0] != 1)
   ;
arrive[0] = 0
```

# Symmetric Barrier, 2 threads (correct)

arrive[0] == arrive[1] == 0 initially

Thread 0's code

```
while (arrive[0] != 0)
   ;
arrive[0] = 1
while (arrive[1] != 1)
   ;
arrive[1] = 0
```

Thread 1's code

```
while (arrive[1] != 0)
   ;
arrive[1] = 1
while (arrive[0] != 1)
   ;
arrive[0] = 0
```

# Symmetric Barrier, $2^p$ threads

- Conceptually, just glue multiple two-thread barriers together
  - Problem:  flags meant for one thread might be seen by another thread

# Dissemination Barrier

int arrive[0:P-1] = {0, 0, …, 0}

Thread i's code:

```
for j = 1 to ceiling(log(P)) {
   while (arrive[i] != 0)  ;
   arrive[i] = j
   lookAt = (i + 2^(j-1)) mod P
   while (arrive[lookAt] != j)  ;
   arrive[lookAt] = 0
}
```