A Tale of Two Assembly Codes*

Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

* With apologies to Charles Dickens


```
int  NumInThreadTeam;
int  NumAtBarrier;
int  NumGone;

void  InitBarrier( int n )
{
      NumInThreadTeam = n;
      NumAtBarrier = 0;
}

void WaitBarrier( )
{
       omp_set_lock( &Lock );
       {
              NumAtBarrier++;
              if( NumAtBarrier == NumInThreadTeam )
              {
                     NumGone = 0;
                     NumAtBarrier = 0;
                     // let all other threads return before this one unlocks:
                     while( NumGone != NumInThreadTeam-1 );
                     omp_unset_lock( &Lock );
                     return;
              }
       }
       omp_unset_lock( &Lock );

       while( NumAtBarrier != 0 );
       #pragma omp atomic
       NumGone++;
}
```

Oregon St
Universit
Computer Gr

Assembly Code with -O3

```
NumGone = 0;
NumAtBarrier = 0;
// let all other threads return before this one unlocks:
while( NumGone != NumInThreadTeam-1 );
```

```
        subq  $8, %rsp
        .cfi_def_cfa_offset 16
        movl  $Lock, %edi
        call    omp_set_lock
        movl  NumAtBarrier(%rip), %eax
        addl  $1, %eax
        cmpl  NumInThreadTeam(%rip), %eax
        movl  %eax, NumAtBarrier(%rip)
        je      .L145
        movl  $Lock, %edi
        call    omp_unset_lock
        movl  NumAtBarrier(%rip), %eax
        testl   %eax, %eax
        je      .L139
.L140:
        jmp   .L140
.L139:
        lock addl     $1, NumGone(%rip)
        addq  $8, %rsp
        ret
.L145:
        cmpl  $1, %eax
        movl  $0, NumGone(%rip)
        movl  $0, NumAtBarrier(%rip)
        je      .L146
.L142:
        jmp   .L142
.L146:
        movl  $Lock, %edi
        addq  $8, %rsp
        .cfi_def_cfa_offset 8
        jmp   omp_unset_lock
```

Oregon State University
Computer Graphics

From the Parallel Background Notes:

Tip #4 -- Sending a Message to the Optimizer: The *volatile* Keyword

The *volatile* keyword is used to let the compiler know that another thread might be changing a variable "in the background", so don't make any assumptions about what can be optimized away.

```
int val = 0;
    . . .
while(  val != 0  ) ;
```

A good compiler optimizer will *eliminate* this code because it *"knows"* that, for all time, *val* == 0

```
volatile int val = 0;
    . . .
while(  val != 0  ) ;
```

The ***volatile*** keyword tells the compiler optimizer that it cannot count on *val* being == 0 here

Oregon State University
Computer Graphics

Assembly Code with -O3

5

```
NumGone = 0;
NumAtBarrier = 0;
// let all other threads return before this one unlocks:
while( NumGone != NumInThreadTeam-1 );
```

```
        subq  $8, %rsp
        .cfi_def_cfa_offset 16
        movl  $Lock, %edi
        call  omp_set_lock
        movl  NumAtBarrier(%rip), %eax
        addl  $1, %eax
        cmpl  NumInThreadTeam(%rip), %eax
        movl  %eax, NumAtBarrier(%rip)
        je    .L145
        movl  $Lock, %edi
        call  omp_unset_lock
        movl  NumAtBarrier(%rip), %eax
        testl %eax, %eax
        je    .L139
.L140:
        jmp   .L140
.L139:
        lock addl   $1, NumGone(%rip)
        addq  $8, %rsp
        ret
.L145:
        cmpl  $1, %eax
        movl  $0, NumGone(%rip)
        movl  $0, NumAtBarrier(%rip)
        je    .L146
.L142:
        jmp   .L142
.L146:
        movl  $Lock, %edi
        addq  $8, %rsp
        .cfi_def_cfa_offset 8
        jmp   omp_unset_lock
```

Oregon State University
Computer Graphics

6

```
volatile int  NumInThreadTeam;
volatile int  NumAtBarrier;
volatile int  NumGone;

void  InitBarrier( int n )
{
      NumInThreadTeam = n;
      NumAtBarrier = 0;
}

void  WaitBarrier( )
{
       omp_set_lock( &Lock );
       {
              NumAtBarrier++;
              if( NumAtBarrier == NumInThreadTeam )
              {
                     NumGone = 0;
                     NumAtBarrier = 0;
                     // let all other threads return before this one unlocks:
                     while( NumGone != NumInThreadTeam-1 );
                     omp_unset_lock( &Lock );
                     return;
              }
       }
       omp_unset_lock( &Lock );

       while( NumAtBarrier != 0 );
       #pragma omp atomic
       NumGone++;
}
```

Oregon St
Universit
Computer Gra

Assembly Code with *volatile* and –O3

7

```
.L142:
        jmp   .L142
```

```
        subq  $8, %rsp
        movl  $Lock, %edi
        call  omp_set_lock
        movl  NumAtBarrier(%rip), %eax
        addl  $1, %eax
        movl  %eax, NumAtBarrier(%rip)
        movl  NumAtBarrier(%rip), %edx
        movl  NumInThreadTeam(%rip), %eax
        cmpl  %eax, %edx
        je    .L128
        movl  $Lock, %edi
        call  omp_unset_lock
.L126:
        movl  NumAtBarrier(%rip), %eax
        testl %eax, %eax
        jne   .L126
        lock addl   $1, NumGone(%rip)
        addq  $8, %rsp
        ret
.L128:
        movl  $0, NumGone(%rip)
        movl  $0, NumAtBarrier(%rip)
.L124:
        movl  NumInThreadTeam(%rip), %edx
        movl  NumGone(%rip), %eax
        subl  $1, %edx
        cmpl  %eax, %edx
        jne   .L124
        movl  $Lock, %edi
        addq  $8, %rsp
        jmp   omp_unset_lock
```

Oregon State University
Computer Graphics

Moral of the Story

8

I should read my own notes more often… ☺

Oregon State University
Computer Graphics