

```
%9:
%mul.i.i = shl i64 %6, 8
%cmp222.i = icmp sgt i32 %4, 0
%10 = sext i32 %3 to i64
%wide.trip.count.i = zext i32 %4 to i64
br i1 %cmp222.i, label %preion_for_entry.entry.i.us.preheader, label
... %preion_for_entry.entry.i.preheader
```

T	F
---	---

```
preion_for_entry.entry.i.us.preheader:
br label %preion_for_entry.entry.i.us
```

```
preion_for_entry.entry.i.preheader:
%div.i = fdiv float 0.000000e+00, %2
%broadcast.splatinsert = insertelement <8 x i64> undef, i64 %mul.i.i, i32 0
%broadcast.splat = shufflevector <8 x i64> %broadcast.splatinsert, <8 x i64>
... undef, <8 x i32> zeroinitializer
%broadcast.splatinsert13 = insertelement <8 x i32> undef, i32 %3, i32 0
%broadcast.splat14 = shufflevector <8 x i32> %broadcast.splatinsert13, <8 x
... i32> undef, <8 x i32> zeroinitializer
%broadcast.splatinsert15 = insertelement <8 x i32> undef, i32 %3, i32 0
%broadcast.splat16 = shufflevector <8 x i32> %broadcast.splatinsert15, <8 x
... i32> undef, <8 x i32> zeroinitializer
%broadcast.splatinsert17 = insertelement <8 x float> undef, float %div.i,
... i32 0
%broadcast.splat18 = shufflevector <8 x float> %broadcast.splatinsert17, <8
... x float> undef, <8 x i32> zeroinitializer
%broadcast.splatinsert19 = insertelement <8 x float> undef, float %div.i,
... i32 0
%broadcast.splat20 = shufflevector <8 x float> %broadcast.splatinsert19, <8
... x float> undef, <8 x i32> zeroinitializer
%.scalar = or i64 %mul.i.i, 8
%11 = insertelement <8 x i64> undef, i64 %.scalar, i64 0
%12 = shufflevector <8 x i64> %11, <8 x i64> undef, <8 x i32> zeroinitializer
br label %vector.body
```

```
preion_for_entry.entry.i.us:
%local_id_x.0.us = phi i64 [ %30, %if.end.r_exit.i.us ], [ 0,
... %preion_for_entry.entry.i.us.preheader ]
%add1.i.i.us = add nuw nsw i64 %local_id_x.0.us, %mul.i.i
%conv.i.us = trunc i64 %add1.i.i.us to i32
%cmp.i.us = icmp slt i32 %conv.i.us, %3
br i1 %cmp.i.us, label %if.then.i.us, label %if.end.r_exit.i.us
```

T	F
---	---

```
vector.body:
%index = phi i64 [ 0, %preion_for_entry.entry.i.preheader ], [ %index.next,
... %vector.body ]
%vec.ind = phi <8 x i64> [ <i64 0, i64 1, i64 2, i64 3, i64 4, i64 5, i64 6,
... i64 7>, %preion_for_entry.entry.i.preheader ], [ %vec.ind.next, %vector.body
... ]
%13 = add nuw nsw <8 x i64> %vec.ind, %broadcast.splat
%14 = add <8 x i64> %12, %vec.ind
%15 = trunc <8 x i64> %13 to <8 x i32>
%16 = trunc <8 x i64> %14 to <8 x i32>
%17 = icmp sgt <8 x i32> %broadcast.splat14, %15
%18 = icmp sgt <8 x i32> %broadcast.splat16, %16
%19 = extractelement <8 x i64> %13, i32 0
%20 = shl i64 %19, 32
%21 = ashr exact i64 %20, 32
%22 = getelementptr inbounds float, float* %0, i64 %21
%23 = bitcast float* %22 to <8 x float>*
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18,
... <8 x float>* %23, i32 4, <8 x i1> %17), !tbaa !12, !llvm.access.group !16
%24 = getelementptr inbounds float, float* %22, i64 8
%25 = bitcast float* %24 to <8 x float>*
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat20,
... <8 x float>* %25, i32 4, <8 x i1> %18), !tbaa !12, !llvm.access.group !16
%index.next = add i64 %index, 16
%vec.ind.next = add <8 x i64> %vec.ind, <i64 16, i64 16, i64 16, i64 16, i64
... 16, i64 16, i64 16, i64 16>
%26 = icmp eq i64 %index.next, 256
br i1 %26, label %mean_kernel.exit.loopexit22, label %vector.body,
... !llvm.loop !18
```

T	F
---	---

```
if.then.i.us:
%sext.i.us = shl i64 %add1.i.i.us, 32
%idxprom.i.us = ashr exact i64 %sext.i.us, 32
%arrayidx.i.us = getelementptr inbounds float, float* %0, i64 %idxprom.i.us
store float 0.000000e+00, float* %arrayidx.i.us, align 4, !tbaa !12,
... !llvm.access.group !16
br label %for.body.i.us
```

```
for.body.i.us:
%indvars.iv.next.i5.us = phi i64 [ %indvars.iv.next.i.us, %for.body.i.us ],
... [ 0, %if.then.i.us ]
%add8.i2.us = phi float [ %add8.i.us, %for.body.i.us ], [ 0.000000e+00,
... %if.then.i.us ]
%27 = mul nsw i64 %indvars.iv.next.i5.us, %10
%28 = add nsw i64 %27, %idxprom.i.us
%arrayidx5.i.us = getelementptr inbounds float, float* %1, i64 %28
%29 = load float, float* %arrayidx5.i.us, align 4, !tbaa !12
%add8.i.us = fadd float %add8.i2.us, %29
store float %add8.i.us, float* %arrayidx.i.us, align 4, !tbaa !12,
... !llvm.access.group !16
%indvars.iv.next.i.us = add nuw nsw i64 %indvars.iv.next.i5.us, 1
%exitcond.not.i.us = icmp eq i64 %indvars.iv.next.i.us, %wide.trip.count.i
br i1 %exitcond.not.i.us, label %for.end.loopexit.i.us, label
... %for.body.i.us, !llvm.loop !21
```

T	F
---	---

```
for.end.loopexit.i.us:
%add8.i.us.lcssa = phi float [ %add8.i.us, %for.body.i.us ]
%div.i.us = fdiv float %add8.i.us.lcssa, %2, !fpmath !23
store float %div.i.us, float* %arrayidx.i.us, align 4, !tbaa !12,
... !llvm.access.group !16
br label %if.end.r_exit.i.us
```

```
if.end.r_exit.i.us:
%30 = add nuw nsw i64 %local_id_x.0.us, 1
%exitcond.not = icmp eq i64 %30, 256
br i1 %exitcond.not, label %mean_kernel.exit.loopexit, label
... %preion_for_entry.entry.i.us, !llvm.loop !24
```

T	F
---	---

```
mean_kernel.exit.loopexit:
br label %mean_kernel.exit
```

```
mean_kernel.exit:
ret void
```

CFG for '_pocl_kernel_mean_kernel' function