%9: %10 = sext i 32 % 4 to i 64%11 = icmp slt i64 %10, 256%12 = select i1 %11, i64 %10, i64 256 %mul.i.i = shl i64 %6, 8 %cmp221.i = icmp sgt i32 %3, 0, !llvm.access.group !12 %wide.trip.count.i = zext i32 %3 to i64 %13 = icmp ugt i64 %12, 1%umax = select i1 %13, i64 %12, i64 1 %min.iters.check = icmp ult i64 %umax, 8 br i1 %min.iters.check, label %pregion for entry.entry.i.preheader, label ... %vector.ph Τ vector.ph: %n.vec = and i64 %umax, -8%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.splatinsert8 = insertelement <8 x i64> undef, i64 %10, i32 0 %broadcast.splat9 = shufflevector <8 x i64> %broadcast.splatinsert8, <8 x ... i64> undef, <8 x i32> zeroinitializer %broadcast.splatinsert11 = insertelement <8 x i64> undef. i64 ... %wide.trip.count.i, i32 0 %broadcast.splat12 = shufflevector <8 x i64> %broadcast.splatinsert11, <8 x ... i64> undef. <8 x i32> zeroinitializer br label %vector.body vector.body: %index = phi i64 [ 0, %vector.ph ], [ %index.next, %if.end.r exit.i14 ] % vec.ind = phi  $< 8 \times 164 > [< 164 \ 0, 164 \ 1, 164 \ 2, 164 \ 3, 164 \ 4, 164 \ 5, 164 \ 6, 164 \$ ... i64 7>, %vector.ph], [%vec.ind.next, %if.end.r exit.i14] %14 = add <8 x i64> %vec.ind, %broadcast.splat, !llvm.access.group !12  $%15 = shl < 8 \times i64 > %14$ , < i64 32, i64... i64 32, i64 32>, !llvm.access.group !12  $\%16 = ashr exact < 8 \times i64 > \%15$ , < i64 32, i64 32... 32, i64 32, i64 32>, !llvm.access.group !12 %17 = getelementptr inbounds float, float\* %2, <8 x i64> %16, ...!llvm.access.group!12 call void @llvm.masked.scatter.v8f32.v8p0f32(<8 x float> zeroinitializer, <8 ... x float\*> %17, i32 4, <8 x i1> <i1 true, i1 true, i1 true, i1 true, i1 true, ... i1 true, i1 true, i1 true>), !tbaa !14, !llvm.access.group !12 br i1 %cmp221.i, label %for.body.i6.preheader, label %if.end.r exit.i14 Τ for.body.i6.preheader: br label %for.body.i6 for.body.i6:  $\text{%vec.phi} = \text{phi} < 8 \times i64 > [\%23, \%for.body.i6], [zeroinitializer,]$ .. %for.body.i6.preheader ] %vec.phi7 = phi <8 x float> [ %22, %for.body.i6 ], [ zeroinitializer, ... %for.body.i6.preheader ] %18 = mul nsw <8 x i64> %vec.phi, %broadcast.splat9, !llvm.access.group !12 %19 = add nsw <8 x i64> %18, %16, !llvm.access.group !12 %20 = getelementptr inbounds float, float\* %0,  $<8 \times i64 > \%19$ , ...!llvm.access.group!12 %wide.masked.gather = call <8 x float> @llvm.masked.gather.v8f32.v8p0f32(<8 ... x float\*> %20, i32 4, <8 x i1> <i1 true, i1 ... i1 true, i1 true, i1 true>, <8 x float> undef), !tbaa !14, !llvm.access.group %21 = getelementptr inbounds float, float\* %1, <8 x i64> %vec.phi, ...!llvm.access.group!12 %wide.masked.gather10 = call <8 x float> ... @llvm.masked.gather.v8f32.v8p0f32(<8 x float\*> %21, i32 4, <8 x i1> <i1 true, ... i1 true, i7 true, i1 true, ... undef), !tbaa !14, !llvm.access.group !12 %22 = call <8 x float> @llvm.fmuladd.v8f32(<8 x float> %wide.masked.gather, ... <8 x float> %wide.masked.gather10, <8 x float> %vec.phi7), !llvm.access.group ... !12 call void @llvm.masked.scatter.v8f32.v8p0f32(<8 x float> %22, <8 x float\*> ... %17, i32 4, <8 x i1> <i1 true, i1 true, i1 true, i1 true, i1 true, i1 true, i1 true, ... i1 true, i1 true>), !tbaa !14, !llvm.access.group !12 %23 = add nuw nsw <8 x i64> %vec.phi, <i64 1, i64 1, i64 1, i64 1, i64 1, ... i64 1, i64 1, i64 1>, !llvm.access.group !12 %24 = icmp eq <8 x i64> %23, %broadcast.splat12, !llvm.access.group !12 %25 = extractelement < 8 x i1 > %24, i32 0br i1 %25, label %if.end.r exit.i14.loopexit, label %for.body.i6 F if.end.r exit.i14.loopexit: br label %if.end.r exit.i14 if.end.r exit.i14: %index.next = add i64 %index, 8 %vec.ind.next = add <8 x i64> %vec.ind, <i64 8, i64 8, i64 8, i64 8, i64 8, ... i64 8, i64 8, i64 8> %26 = icmp eq i64 %index.next, %n.vec br i1 %26, label %middle.block, label %vector.body, !llvm.loop !18 F middle.block: %cmp.n = icmp eq i64 %umax, %n.vec br i1 %cmp.n, label %bicgKernel2.exit, label ... %pregion for entry.entry.i.preheader pregion for entry.entry.i.preheader: %\_local\_id\_x.0.ph = phi i64 [ 0, %9 ], [ %n.vec, %middle.block ] br label %pregion for entry.entry.i pregion for entry.entry.i: % local id x.0 = phi i64 [ %33, %if.end.r\_exit.i ], [ %\_local\_id\_x.0.ph, ... %pregion for entry.entry.i.preheader ] %add1.i.i = add i64 % local\_id\_x.0, %mul.i.i, !llvm.access.group !12 %sext.i = shl i64 %add1.i.i, 32, !llvm.access.group !12 %idxprom.i = ashr exact i64 %sext.i, 32, !llvm.access.group !12 %arrayidx.i = getelementptr inbounds float, float\* %2, i64 %idxprom.i, ...!llvm.access.group!12 store float 0.000000e+00, float\* %arrayidx.i, align 4, !tbaa !14, ...!llvm.access.group!12 br i1 %cmp221.i, label %for.body.i.preheader, label %if.end.r exit.i, ...!llvm.access.group!12 for.body.i.preheader: br label %for.body.i for.body.i: %indvars.iv.next.i2 = phi i64 [ %indvars.iv.next.i, %for.body.i ], [ 0, .. %for.body.i.preheader ] %27 = phi float [ %32, %for.body.i ], [ 0.000000e+00, %for.body.i.preheader ] %28 = mul nsw i64 %indvars.iv.next.i2, %10, !llvm.access.group !12 %29 = add nsw i64 %28, %idxprom.i, !llvm.access.group !12 %arrayidx5.i = getelementptr inbounds float, float\* %0, i64 %29, ...!llvm.access.group!12
%30 = load float, float\* %arrayidx5.i, align 4, !tbaa!14, ..!llvm.access.group!12 %arrayidx7.i = getelementptr inbounds float, float\* %1, i64 ... %indvars.iv.next.i2, !llvm.access.group !12 %31 = load float, float\* %arrayidx7.i, align 4, !tbaa !14, ...!llvm.access.group!12
%32 = tail call float @llvm.fmuladd.f32(float %30, float %31, float %27) #5, ..!llvm.access.group!12 store float %32, float\* %arrayidx.i, align 4, !tbaa !14, !llvm.access.group .. !12 %indvars.iv.next.i = add nuw nsw i64 %indvars.iv.next.i2, 1, .. !llvm.access.group !12 %exitcond.not.i = icmp eq i64 %indvars.iv.next.i, %wide.trip.count.i, .. !llvm.access.group !12 br i1 %exitcond.not.i, label %if.end.r\_exit.i.loopexit, label %for.body.i, ...!llvm.loop!21,!llvm.access.group!12 F if.end.r exit.i.loopexit: br label %if.end.r exit.i if.end.r\_exit.i: %33 = add nuw i64 %\_local\_id\_x.0, 1 %exitcond.not = icmp eq  $i6\overline{4}$  %33, %umax br i1 %exitcond.not, label %bicgKernel2.exit.loopexit, label ... %pregion for entry.entry.i, !llvm.loop !23 bicgKernel2.exit.loopexit: br label %bicgKernel2.exit bicgKernel2.exit: ret void