%9: %10 = sext i 32 % 3 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 %4, 0, !llvm.access.group !12 %wide.trip.count.i = zext i32 %4 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.splatinsert6 = insertelement <8 x i32> undef, i32 %4, i32 0 %broadcast.splat7 = shufflevector <8 x i32> %broadcast.splatinsert6, <8 x ... i32> 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 $< 8 \times i64 > [< i64 0, i64 1, i64 2, i64 3, i64 4, i64 5, i64 6,$... 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 x i64> %14, <i64 32, i64 32, i64 32, i64 32, i64 32, i64 32, i64 32, ... i64 32, i64 32>, !llvm.access.group !12 %16 = ashr exact <8 x i64> %15, <i64 32, i64 32, i64 32, i64 32, i64 32, i64 32, i64 ... 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.lr.ph.i5, label %if.end.r exit.i14 for.body.lr.ph.i5: %18 = trunc <8 x i64> %14 to <8 x i32>, !llvm.access.group !12 %19 = mul nsw <8 x i32> %broadcast.splat7, %18, !llvm.access.group !12 $\%20 = \text{sext} < 8 \times \text{i}32 > \%19 \text{ to } < 8 \times \text{i}64 > \text{, !llvm.access.group !}12$ br label %for.body.i8 for.body.i8: %vec.phi = phi <8 x i64> [%25, %for.body.i8], [zeroinitializer, ... %for.body.lr.ph.i5] %vec.phi9 = phi <8 x float> [%24, %for.body.i8], [zeroinitializer, ... %for.body.lr.ph.i5] %21 = add nsw <8 x i64> %vec.phi, %20, !llvm.access.group !12 $\%22 = \text{getelementptr inbounds float, float* } \%0, <8 \times i64 > \%21,$...!llvm.access.group!12 %wide.masked.gather = call <8 x float> @llvm.masked.gather.v8f32.v8p0f32(<8 ... x float*> %22, i32 4, <8 x i1> <i1 true, i1 ... i1 true, i1 true, i1 true, <8 x float> undef), !tbaa !14, !llvm.access.group ...!12 %23 = 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*> %23, i32 4, <8 x i1> <i1 true, ... i1 true, i7 true, i1 true, ... undef), !tbaa !14, !llvm.access.group !12 %24 = call <8 x float> @llvm.fmuladd.v8f32(<8 x float> %wide.masked.gather, ... <8 x float> %wide.masked.gather10, <8 x float> %vec.phi9), !llvm.access.group call void @llvm.masked.scatter.v8f32.v8p0f32(<8 x float> %24, <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 %25 = 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 $\%26 = icmp \ eq < 8 \ x \ i64 > \%25$, %broadcast.splat12, !llvm.access.group !12 $%27 = \text{extractelement} < 8 \times i1 > %26, i32 0$ br i1 %27, label %if.end.r exit.i14.loopexit, label %for.body.i8 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> %28 = icmp eq i64 %index.next, %n.vec br i1 %28, label %middle.block, label %vector.body, !llvm.loop !18 middle.block: %cmp.n = icmp eq i64 %umax, %n.vec br i1 %cmp.n, label %bicgKernel1.exit, label ... %pregion for entry.entry.i.preheader pregion for entry.entry.i.preheader: $\begin{call} \begin{call} \be$ br label % pregion for entry entry i pregion_for entry.entry.i: %_local_id_x.0 = phi i64 [%35, %if.end.r_exit.i], [%_local_id_x.0.ph, ... %pregion_for_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.lr.ph.i, label %if.end.r_exit.i, ..!llvm.access.group!12 F for.body.lr.ph.i: %conv.i = trunc i64 %add1.i.i to i32, !llvm.access.group !12 %mul.i = mul nsw i32 %conv.i, %4, !llvm.access.group !12 %29 = sext i32 %mul.i to i64, !llvm.access.group !12 br label %for.body.i, !llvm.access.group !12 for.body.i: %indvars.iv.next.i2 = phi i64 [%indvars.iv.next.i, %for.body.i], [0, .. %for.body.lr.ph.i] %30 = phi float [%34, %for.body.i], [0.000000e+00, %for.body.lr.ph.i] %31 = add nsw i64 %indvars.iv.next.i2, %29, !llvm.access.group !12 %arrayidx5.i = getelementptr inbounds float, float* %0, i64 %31, ...!llvm.access.group!12
%32 = 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 %33 = load float, float* %arrayidx7.i, align 4, !tbaa !14, ...!llvm.access.group!12
%34 = tail call float @llvm.fmuladd.f32(float %32, float %33, float %30) #5, ...!llvm.access.group!12 store float %34, float* %arrayidx.i, align 4, !tbaa !14, !llvm.access.group %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: %35 = add nuw i64 % local id x.0, 1%exitcond.not = icmp eq $i6\overline{4}$ %35, %umax br i1 %exitcond.not, label %bicgKernel1.exit.loopexit, label ... %pregion_for_entry.entry.i, !llvm.loop !23 F bicgKernel1.exit.loopexit: br label %bicgKernel1.exit bicgKernel1.exit:

ret void

CFG for ' pocl kernel bicgKernel1' function