%13: %mul.i.i = shl i64 %10, 5 %mul3.i.i = shl i64 %11, 3 %cmp32.i = icmp sgt i32 %5, 0, !llvm.access.group !12 %14 = sext i32 %4 to i64 %wide.trip.count.i = zext i32 %5 to i64 br label %pregion\_for\_entry.pregion\_for\_init.i pregion\_for\_entry.pregion\_for\_init.i:
%\_local\_id\_y.0 = phi i64 [ 0, %13 ], [ %47, %pregion\_for\_end.i ]
%add6.i.i = add nuw nsw i64 %\_local\_id\_y.0, %mul3.i.i, !llvm.access.group !12
%conv2.i = trunc i64 %add6.i.i to i32, !llvm.access.group !12
%mul.i = mul nsw i32 %conv2.i, %4, !llvm.access.group !12
%mul4.i = mul nsw i32 %conv2.i, %5 %15 = sext i32 %mul4.i to i64 br i1 false, label %scalar.ph, label %vector.ph F vector.ph: %broadcast.splatinsert = insertelement <8 x i64> undef, i64 %mul.i.i, i32 0 %broadcast.splat = shufflevector  $< 8 \times i64 >$  %broadcast.splatinsert,  $< 8 \times i64 >$  ... undef,  $< 8 \times i32 >$  zeroinitializer %broadcast.splatinsert1 = insertelement <8 x i32> undef, i32 %mul.i, i32 0 %broadcast.splat2 = shufflevector <8 x i32> %broadcast.splatinsert1, <8 x ... i32> undef, <8 x i32> zeroinitializer %broadcast.splatinsert3 = insertelement <8 x i1> undef, i1 %cmp32.i, i32 0 %broadcast.splat4 = shufflevector <8 x i1> %broadcast.splatinsert3, <8 x i1> ... undef, <8 x i32> zeroinitializer %broadcast.splatinsert8 = insertelement <8 x i64> undef, i64 %15, i32 0 %broadcast.splat9 = shufflevector <8 x i64> %broadcast.splatinsert8, <8 x ... i64> undef, <8 x i32> zeroinitializer %broadcast.splatinsert10 = insertelement <8 x float> undef, float %7, i32 0 %broadcast.splat11 = shufflevector <8 x float> %broadcast.splatinsert10, <8 ... x float> undef, <8 x i32> zeroinitializer %broadcast.splatinsert12 = insertelement <8 x i64> undef, i64 %14, i32 0 %broadcast.splat13 = shufflevector <8 x i64> %broadcast.splatinsert12, <8 x ... i64> undef, <8 x i32> zeroinitializer %broadcast.splatinsert15 = insertelement <8 x i64> undef, i64 ... %wide.trip.count.i, i32 0
%broadcast.splat16 = shufflevector <8 x i64> %broadcast.splatinsert15, <8 x
... i64> undef, <8 x i32> zeroinitializer br label %vector.body vector.body: %index = phi i64 [ 0, %vector.ph ], [ %index.next, %for.end.r\_exit.i18 ] %vec.ind = phi <8 x i64> [ <i64 0, i64 1, i64 2, i64 3, i64 4, i64 5, i64 6, ... i64 7>, %vector.ph ], [ %vec.ind.next, %for.end.r\_exit.i18 ] %16 = add nuw nsw <8 x i64> %vec.ind, %broadcast.splat, !llvm.access.group %17 = trunc <8 x i64> %16 to <8 x i32>, !llvm.access.group !12 %18 = add nsw <8 x i32> %broadcast.splat2, %17, !llvm.access.group !12 %19 = sext < 8 x i 32 > %18 to < 8 x i 64 > , !llvm.access.group ! 12%20 = getelementptr inbounds float, float\* %0, <8 x i64> %19, ...!llvm.access.group!12 call void @llvm.masked.scatter.v8f32.v8p0f32(<8 x float> zeroinitializer, <8 ... x float\*> %20, i32 4, <8 x i1> <i1 true, i1 true ... i1 true, i1 true, i1 true>), !tbaa !15, !llvm.access.group !12 %21 = extractelement <8 x i1> %broadcast.splat4, i32 0 br i1 %21, label %for.body.lr.ph.i5, label %for.end.r exit.i18 for.body.lr.ph.i5:  $\%22 = sh\hat{1} < 8 \times i64 > \%16$ , < i64 32, ... i64 32, i64 32>, !llvm.access.group !12  $%23 = ashr exact < 8 \times i64 > %22, < i64 32, i64 32,$ ... 32, i64 32, i64 32>, !llvm.access.group !12 br label %for.body.i6 for.body.i6:  $\text{wec.phi} = \text{phi} < 8 \times i64 > [\%31, \% \text{for.body.} i6], [zeroinitializer,]$ ... %for.body.lr.ph.i5 ] %vec.phi7 = phi <8 x float> [ %30, %for.body.i6 ], [ zeroinitializer, .. %for.body.lr.ph.i5 ] %24 = add nsw <8 x i64> %vec.phi, %broadcast.splat9, !llvm.access.group !12 %25 = getelementptr inbounds float, float\* %1, <8 x i64> %24, ...!llvm.access.group!12 %wide.masked.gather = call <8 x float> @llvm.masked.gather.v8f32.v8p0f32(<8 ... x float\*> %25, i32 4, <8 x i1> <i1 true, i1 true, ... i1 true, i1 true, i1 true>, <8 x float> undef), !tbaa !15, !llvm.access.group %26 = fmul <8 x float> %wide.masked.gather, %broadcast.splat11, ...!llvm.access.group!12 %27 = mul nsw <8 x i64> %vec.phi, %broadcast.splat13, !llvm.access.group !12 %28 = add nsw <8 x i64> %27, %23, !llvm.access.group !12 %29 = getelementptr inbounds float, float\* %2, <8 x i64 > %28,...!llvm.access.group!12
%wide.masked.gather14 = call <8 x float> ... @llvm.masked.gather.v8f32.v8p0f32(<8 x float\*> %29, i32 4, <8 x i1> <i1 true, i1 ... undef), !tbaa !15, !llvm.access.group !12
%30 = call <8 x float> @llvm.fmuladd.v8f32(<8 x float> %26, <8 x float> "30 = can <8 x noat> @nvm.imuladd.v8i32(<8 x noat> %26, <8 x noat> ... %wide.masked.gather14, <8 x float> %vec.phi7), !llvm.access.group !12 call void @llvm.masked.scatter.v8f32.v8p0f32(<8 x float> %30, <8 x float\*> ... %20, i32 4, <8 x i1> <i1 true, i1 t  $\%32 = icmp eq < 8 \times i64 > \%31$ , %broadcast.splat16, !llvm.access.group !12  $%33 = \text{extractelement} < 8 \times i1 > %32, i32 0$ br i1 %33, label %for.end.r exit.i.loopexit17, label %for.body.i6 for.end.r\_exit.i.loopexit17: br label %for.end.r\_exit.i18 for.end.r exit.i18:  $%34 = \bar{a}dd \text{ nuw nsw } < 8 \text{ x } i64 > \text{ wec.ind}, < i64 1, i64 1$ ... i64 1, i64 1, i64 1> %35 = icmp eq <8 x i64> %34, <i64 32, i64 32,  $%36 = \text{extractelement} < 8 \times i1 > %35, i32 0$ %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> %37 = icmp eq i64 %index.next, 32 br i1 %37, label %middle.block, label %vector.body, !llvm.loop !19 middle.block: %cmp.n = icmp eq i64 32, 32 br i1 %cmp.n, label %pregion\_for\_end.i, label %scalar.ph scalar.ph: %bc.resume.val = phi i64 [ 32, %middle.block ], [ 0, ... %pregion for entry.pregion for init.i ] br label %pregion for entry.entry.i pregion\_for\_entry.entry.i:
 % local id x.0 = phi i64 [ %bc.resume.val, %scalar.ph ], [ %46, ... %for.end.r exit.i ] %add1.i.i = add nuw nsw i64 %\_local\_id\_x.0, %mul.i.i, !llvm.access.group !12 %conv.i = trunc i64 %add1.i.i to i32, !llvm.access.group !12 %add.i = add nsw i32 %mul.i, %conv.i, !llvm.access.group !12 %idxprom.i = sext i32 %add.i to i64, !llvm.access.group !12 %arrayidx.i = getelementptr inbounds float, float\* %0, i64 %idxprom.i, ...!llvm.access.group!12 store float 0.000000e+00, float\* %arrayidx.i, align 4, !tbaa!15, ...!llvm.access.group!12 br i1 %cmp32.i, label %for.body.lr.ph.i, label %for.end.r exit.i, ...!llvm.access.group!12 for.body.lr.ph.i: %sext.i = shl i64 %add1.i.i, 32, !llvm.access.group !12 %38 = ashr exact i64 %sext.i, 32, !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 ] %39 = phi float [ %45, %for.body.i ], [ 0.000000e+00, %for.body.lr.ph.i ] %40 = add nsw i64 %indvars.iv.next.i2, %15, !llvm.access.group !12 %arrayidx7.i = getelementptr inbounds float, float\* %1, i64 %40, ...!llvm.access.group!12 %41 = load float, float\* %arrayidx7.i, align 4, !tbaa !15, ...!llvm.access.group!12
%mul8.i = fmul float %41, %7, !llvm.access.group!12
%42 = mul nsw i64 %indvars.iv.next.i2, %14, !llvm.access.group!12 %43 = add nsw i64 %42, %38, !llvm.access.group !12 %arrayidx12.i = getelementptr inbounds float, float\* %2, i64 %43, ...!llvm.access.group!12
%44 = load float, float\* %arrayidx12.i, align 4, !tbaa!15, ...!llvm.access.group!12
%45 = call float @llvm.fmuladd.f32(float %mul8.i, float %44, float %39) #5, ...!llvm.access.group!12 store float %45, float\* %arrayidx.i, align 4, !tbaa!15, !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 %for.end.r\_exit.i.loopexit, label %for.body.i, ...!llvm.loop!22,!llvm.access.group!12 for.end.r exit.i.loopexit: br label %for.end.r exit.i for.end.r exit.i:  $%46 = \bar{a}dd \text{ nuw nsw } i64 \% \text{ local } id x.0, 1$ %exitcond.not = icmp eq  $i\overline{6}4$  % $\overline{4}\overline{6}$ ,  $\overline{3}2$ br i1 %exitcond.not, label %pregion\_for\_end.i, label ... %pregion\_for\_entry.entry.i, !llvm.loop !24 pregion for end.i:  $^{1}\%47 = \overline{add}$  nuw nsw i64 % local id y.0, 1 %exitcond3.not = icmp eq  $\overline{i}64 \% \overline{47}$ , 8br i1 %exitcond3.not, label %mm2 kernel1.exit, label ... %pregion for entry.pregion for init.i, !llvm.loop !25 mm2 kernel1.exit: ret void CFG for 'pocl kernel mm2 kernel1' function