%9: %mul.i.i = shl i64 %6, 8 %cmp221.i = icmp sgt i32 %4, 0 %wide.trip.count.i = zext i32 %4 to i64 br i1 %cmp221.i, label %pregion for entry.entry.i.us.preheader, label ... %vector.ph Τ F pregion for entry.entry.i.us.preheader: br label %pregion for entry.entry.i.us vector.ph: %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.splatinsert10 = insertelement <8 x i32> undef, i32 %3, i32 0 %broadcast.splat11 = shufflevector <8 x i32> %broadcast.splatinsert10, <8 x ... i32> undef, <8 x i32> zeroinitializer %broadcast.splatinsert12 = insertelement <8 x i32> undef, i32 %3, i32 0 %broadcast.splat13 = shufflevector <8 x i32> %broadcast.splatinsert12, <8 x ... i32> undef, <8 x i32> zeroinitializer %10 = or <8 x i64> %broadcast.splat, <i64 0, i64 1, i64 2, i64 3, i64 4, i64 ... 5, i64 6, i64 7> %11 = trunc <8 x i64> %10 to <8 x i32> %12 = trunc i64 %mul.i.i to i32 %13 = or i32 %12.8%14 = insertelement <8 x i32> undef, i32 %13, i64 0 $%15 = \text{shufflevector} < 8 \times i32 > \%14, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ %16 = or < 8 x i32 > %15, < i32 0, i32 1, i32 2, i32 3, i32 4, i32 5, i32 6,... i32 7> %17 = icmp sqt <8 x i32> %broadcast.splat11, %11 %18 = icmp sgt <8 x i32> %broadcast.splat13, %16 $%19 = \text{extractelement} < 8 \times \text{i}64 > \%10, \text{i}32 \text{ 0}$ %20 = shl i64 %19, 32%21 = ashr exact i64 %20, 32%22 = getelementptr inbounds float, float* %2, i64 %21 %23 = bitcast float* %22 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <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> zeroinitializer, <8 x ... float>* %25, i32 4, <8 x i1> %18), !tbaa !12, !llvm.access.group !16 %26 = or <8 x i64> %broadcast.splat, <i64 16, i64 17, i64 18, i64 19, i64 ... 20, i64 21, i64 22, i64 23> %27 = trunc <8 x i64> %26 to <8 x i32> %28 = trunc i64 %mul.i.i to i32 %29 = or i32 %28, 8%30 = insertelement <8 x i32> undef, i32 %29, i64 0 $%31 = \text{shufflevector} < 8 \times i32 > \%30, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ $\%32 = \text{or} < 8 \times i32 > \%31$, $< i32\ 16$, $i32\ 17$, $i32\ 18$, $i32\ 19$, $i32\ 20$, $i32\ 21$, i32... 22, i32 23> %33 = icmp sgt <8 x i32> %broadcast.splat11, %27 %34 = icmp sgt <8 x i32> %broadcast.splat13, %32 $%35 = \text{extractelement} < 8 \times i64 > \%26, i32 0$ %36 = shl i64 %35, 32%37 = ashr exact i64 %36, 32%38 = getelementptr inbounds float, float* %2, i64 %37 %39 = bitcast float* %38 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %39, i32 4, <8 x i1> %33), !tbaa !12, !llvm.access.group !16 %40 = getelementptr inbounds float, float* %38, i64 8 %41 = bitcast float* %40 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %41, i32 4, <8 x i1> %34), !tbaa !12, !llvm.access.group !16 %42 = or <8 x i64> %broadcast.splat, <i64 32, i64 33, i64 34, i64 35, i64 ... 36, i64 37, i64 38, i64 39> %43 = trunc <8 x i64> %42 to <8 x i32> %44 = trunc i64 %mul.i.i to i32 %45 = or i32 %44, 8%46 = insertelement <8 x i32> undef, i32 %45, i64 0 $\%47 = \text{shufflevector} < 8 \times i32 > \%46, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ $\%48 = \text{or} < 8 \times i32 > \%47$, < i32 32, i32 33, i32 34, i32 35, i32 36, i32 37, i32 39... 38, i32 39> %49 = icmp sgt <8 x i32> %broadcast.splat11, %43 %50 = icmp sgt <8 x i32> %broadcast.splat13, %48 $%51 = \text{extractelement} < 8 \times i64 > %42, i32 0$ %52 = shl i64 %51, 32%53 = ashr exact i64 %52, 32%54 = getelementptr inbounds float, float* %2, i64 %53 %55 = bitcast float* %54 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %55, i32 4, <8 x i1> %49), !tbaa !12, !llvm.access.group !16 %56 = getelementptr inbounds float, float* %54, i64 8 %57 = bitcast float* %56 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %57, i32 4, <8 x i1> %50), !tbaa !12, !llvm.access.group !16 %58 = or <8 x i64> %broadcast.splat, <i64 48, i64 49, i64 50, i64 51, i64 ... 52, i64 53, i64 54, i64 55> $%59 = \text{trunc} < 8 \times i64 > \%58 \text{ to } < 8 \times i32 >$ %60 = trunc i64 %mul.i.i to i32 %61 = or i32 %60, 8%62 = insertelement <8 x i32> undef, i32 %61, i64 0 $\%63 = \text{shufflevector} < 8 \times i32 > \%62, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ $\%64 = \text{or} < 8 \times i32 > \%63$, < i32.48, i32.49, i32.50, i32.51, i32.52, i32.53, i32.53... 54, i32 55> %65 = icmp sgt <8 x i32> %broadcast.splat11, %59 %66 = icmp sgt <8 x i32> %broadcast.splat13, %64 $\%67 = \text{extractelement} < 8 \times i64 > \%58, i32 0$ %68 = shl i64 %67, 32%69 = ashr exact i64 %68, 32%70 = getelementptr inbounds float, float* %2, i64 %69 %71 = bitcast float* %70 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %71, i32 4, <8 x i1> %65), !tbaa !12, !llvm.access.group !16 %72 = getelementptr inbounds float, float* %70, i64 8 %73 = bitcast float* %72 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %73, i32 4, <8 x i1> %66), !tbaa !12, !llvm.access.group !16 $\%74 = \text{or} < 8 \times \text{i}64 > \%\text{broadcast.splat}, < \text{i}64 64, i64 65, i64 66, i64 67, i64$... 68, i64 69, i64 70, i64 71> $\%75 = \text{trunc} < 8 \times i64 > \%74 \text{ to } < 8 \times i32 >$ %76 = trunc i64 %mul.i.i to i32 %77 = or i32 %76, 8%78 = insertelement <8 x i32> undef, i32 %77, i64 0 $\%79 = \text{shufflevector} < 8 \times i32 > \%78, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ $\%80 = \text{or} < 8 \times 32 > \%79$, $< 32 \times 64$, 32×65 , 32×66 , 32×67 , 32×68 , 32×69 , ... 70, i32 71> %81 = icmp sgt <8 x i32> %broadcast.splat11, %75 %82 = icmp sgt <8 x i32> %broadcast.splat13, %80 $\%83 = \text{extractelement} < 8 \times 164 > \%74, 132 0$ %84 = shl i64 %83, 32%85 = ashr exact i64 %84, 32%86 = getelementptr inbounds float, float* %2, i64 %85 %87 = bitcast float* %86 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %87, i32 4, <8 x i1> %81), !tbaa !12, !llvm.access.group !16 %88 = getelementptr inbounds float, float* %86, i64 8 %89 = bitcast float* %88 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %89, i32 4, <8 x i1> %82), !tbaa !12, !llvm.access.group !16 %90 = or <8 x i64> %broadcast.splat, <i64 80, i64 81, i64 82, i64 83, i64 ... 84, i64 85, i64 86, i64 87> %91 = trunc < 8 x i64 > %90 to < 8 x i32 >%92 = trunc i64 %mul.i.i to i32 %93 = or i32 %92, 8%94 = insertelement <8 x i32> undef, i32 %93, i64 0 $\%95 = \text{shufflevector} < 8 \times i32 > \%94, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ $\%96 = \text{or} < 8 \times 32 > \%95$, $< 32 \times 80$, 32×81 , 32×82 , 32×83 , 32×84 , 32×85 , 32×85 , 32×86 ... 86, i32 87> %97 = icmp sgt <8 x i32> %broadcast.splat11, %91 %98 = icmp sgt <8 x i32> %broadcast.splat13, %96 $\%99 = \text{extractelement} < 8 \times \text{i}64 > \%90, \text{i}32 \text{ 0}$ %100 = shl i64 %99, 32%101 = ashr exact i64 %100, 32%102 = getelementptr inbounds float, float* %2, i64 %101 %103 = bitcast float* %102 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* \%103, i32 4, <8 x i1> \%97), !tbaa !12, !llvm.access.group !16 %104 = getelementptr inbounds float, float* %102, i64 8 %105 = bitcast float* %104 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %105, i32 4, <8 x i1> %98), !tbaa !12, !llvm.access.group !16 %106 = or <8 x i64> %broadcast.splat, <i64 96, i64 97, i64 98, i64 99, i64 ... 100, i64 101, i64 102, i64 103> %107 = trunc <8 x i64> %106 to <8 x i32> %108 = trunc i64 %mul.i.i to i32 %109 = or i32 %108, 8%110 = insertelement <8 x i32> undef, i32 %109, i64 0 %111 = shufflevector <8 x i32> %110, <8 x i32> undef, <8 x i32> ... zeroinitializer $%112 = \text{or} < 8 \times 32 > %111$, < 3296, 3297, 3298, 3299, 32100, 32101, ... i32 102, i32 103> %113 = icmp sgt <8 x i32> %broadcast.splat11, %107 %114 = icmp sgt <8 x i32> %broadcast.splat13, %112 $%115 = \text{extractelement} < 8 \times i64 > %106, i32 0$ %116 = shl i64 %115, 32 %117 = ashr exact i64 %116, 32 %118 = getelementptr inbounds float, float* %2, i64 %117 %119 = bitcast float* %118 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %119, i32 4, <8 x i1> %113), !tbaa !12, !llvm.access.group !16 %120 = getelementptr inbounds float, float* %118, i64 8 %121 = bitcast float* %120 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %121, i32 4, <8 x i1> %114), !tbaa !12, !llvm.access.group !16 %122 = or <8 x i64> %broadcast.splat, <i64 112, i64 113, i64 114, i64 115, ... i64 116, i64 117, i64 118, i64 119> %123 = trunc <8 x i64> %122 to <8 x i32> %124 = trunc i64 %mul.i.i to i32 %125 = or i32 %124, 8%126 = insertelement <8 x i32> undef, i32 %125, i64 0 %127 = shufflevector <8 x i32> %126, <8 x i32> undef, <8 x i32> ... zeroinitializer %128 = or < 8 x i32 > %127, < i32 112, i32 113, i32 114, i32 115, i32 116, i32... 117, i32 118, i32 119> %129 = icmp sgt <8 x i32> %broadcast.splat11, %123 %130 = icmp sgt <8 x i32> %broadcast.splat13, %128 $%131 = \text{extractelement} < 8 \times i64 > %122, i32 0$ %132 = shl i64 %131, 32%133 = ashr exact i64 %132, 32 %134 = getelementptr inbounds float, float* %2, i64 %133 %135 = bitcast float* %134 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x pregion for entry.entry.i.us: ... float>* %135, i32 4, <8 x i1> %129), !tbaa !12, !llvm.access.group !16 % local id x.0.us = phi i64 [%279, %if.end.i.us.1], [0, %136 = getelementptr inbounds float, float* %134, i64 8 ... %pregion for entry.entry.i.us.preheader] %137 = bitcast float* %136 to <8 x float>* %add1.i.i.us = add nuw nsw i64 %_local_id_x.0.us, %mul.i.i call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x $%conv.i.us = trunc i64 %add1.i.i.us to i3\overline{2}$... float>* %137, i32 4, <8 x i1> %130), !tbaa !12, !llvm.access.group !16 %cmp.i.us = icmp slt i32 %conv.i.us, %3 %138 = or <8 x i64> %broadcast.splat, <i64 128, i64 129, i64 130, i64 131, br i1 %cmp.i.us, label %if.then.i.us, label %if.end.i.us ... i64 132, i64 133, i64 134, i64 135> $%139 = \text{trunc} < 8 \times i64 > %138 \text{ to} < 8 \times i32 >$ %140 = trunc i64 %mul.i.i to i32 %141 = or i32 %140, 8%142 = insertelement <8 x i32> undef, i32 %141, i64 0 %143 = shufflevector <8 x i32> %142, <8 x i32> undef, <8 x i32> ... zeroinitializer %144 = or < 8 x i32 > %143, < i32 128, i32 129, i32 130, i32 131, i32 132, i32... 133, i32 134, i32 135> %145 = icmp sgt <8 x i32> %broadcast.splat11, %139 %146 = icmp sgt <8 x i32> %broadcast.splat13, %144 $%147 = \text{extractelement} < 8 \times i64 > %138, i32 0$ %148 = shl i64 %147, 32%149 = ashr exact i64 %148, 32 %150 = getelementptr inbounds float, float* %2, i64 %149 %151 = bitcast float* %150 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %151, i32 4, <8 x i1> %145), !tbaa !12, !llvm.access.group !16 %152 = getelementptr inbounds float, float* %150, i64 8 %153 = bitcast float* %152 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %153, i32 4, <8 x i1> %146), !tbaa !12, !llvm.access.group !16 %154 = or <8 x i64> %broadcast.splat, <i64 144, i64 145, i64 146, i64 147, ... i64 148, i64 149, i64 150, i64 151> %155 = trunc <8 x i64> %154 to <8 x i32> %156 = trunc i64 %mul.i.i to i32 %157 = or i32 %156, 8%158 = insertelement <8 x i32> undef, i32 %157, i64 0 %159 = shufflevector <8 x i32> %158, <8 x i32> undef, <8 x i32> ... zeroinitializer $%160 = \text{or} < 8 \times i32 > %159$, < i32 144, i32 145, i32 146, i32 147, i32 148, i32... 149, i32 150, i32 151> %161 = icmp sgt <8 x i32> %broadcast.splat11, %155 %162 = icmp sgt <8 x i32> %broadcast.splat13, %160 $%163 = \text{extractelement} < 8 \times i64 > %154, i32 0$ %164 = shl i64 %163, 32%165 = ashr exact i64 %164, 32%166 = getelementptr inbounds float, float* %2, i64 %165 %167 = bitcast float* %166 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %167, i32 4, <8 x i1> %161), !tbaa !12, !llvm.access.group !16 %168 = getelementptr inbounds float, float* %166, i64 8 %169 = bitcast float* %168 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %169, i32 4, <8 x i1> %162), !tbaa !12, !llvm.access.group !16 %170 = or <8 x i64> %broadcast.splat, <i64 160, i64 161, i64 162, i64 163, ... i64 164, i64 165, i64 166, i64 167> $%171 = \text{trunc} < 8 \times i64 > %170 \text{ to } < 8 \times i32 >$ %172 = trunc i64 %mul.i.i to i32 %173 = or i32 %172, 8%174 = insertelement <8 x i32> undef, i32 %173, i64 0 %175 = shufflevector <8 x i32> %174, <8 x i32> undef, <8 x i32> ... zeroinitializer %176 = or < 8 x i32 > %175, $< i32\ 160$, $i32\ 161$, $i32\ 162$, $i32\ 163$, $i32\ 164$, i32... 165, i32 166, i32 167> %177 = icmp sgt <8 x i32> %broadcast.splat11, %171 %178 = icmp sgt <8 x i32> %broadcast.splat13, %176 $%179 = \text{extractelement} < 8 \times i64 > %170, i32 0$ %180 = shl i64 %179, 32%181 = ashr exact i64 %180, 32%182 = getelementptr inbounds float, float* %2, i64 %181 %183 = bitcast float* %182 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %183, i32 4, <8 x i1> %177), !tbaa !12, !llvm.access.group !16 %184 = getelementptr inbounds float, float* %182, i64 8 %185 = bitcast float* %184 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %185, i32 4, <8 x i1> %178), !tbaa !12, !llvm.access.group !16 %186 = or <8 x i64> %broadcast.splat, <i64 176, i64 177, i64 178, i64 179, ... i64 180, i64 181, i64 182, i64 183> %187 = trunc <8 x i64> %186 to <8 x i32> %188 = trunc i64 %mul.i.i to i32 %189 = or i32 %188, 8%190 = insertelement <8 x i32> undef, i32 %189, i64 0 %191 = shufflevector <8 x i32> %190, <8 x i32> undef, <8 x i32> ... zeroinitializer $%192 = \text{or} < 8 \times i32 > %191$, < i32 176, i32 177, i32 178, i32 179, i32 180, i32... 181, i32 182, i32 183> %193 = icmp sgt <8 x i32> %broadcast.splat11, %187 %194 = icmp sgt <8 x i32> %broadcast.splat13, %192 $%195 = \text{extractelement} < 8 \times i64 > %186, i32 0$ %196 = shl i64 %195, 32%197 = ashr exact i64 %196, 32 %198 = getelementptr inbounds float, float* %2, i64 %197 %199 = bitcast float* %198 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %199, i32 4, <8 x i1> %193), !tbaa !12, !llvm.access.group !16 %200 = getelementptr inbounds float, float* %198, i64 8 %201 = bitcast float* %200 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %201, i32 4, <8 x i1> %194), !tbaa !12, !llvm.access.group !16 %202 = or <8 x i64> %broadcast.splat, <i64 192, i64 193, i64 194, i64 195, ... i64 196, i64 197, i64 198, i64 199> $%203 = trunc < 8 \times i64 > %202 \text{ to } < 8 \times i32 >$ %204 = trunc i64 %mul.i.i to i32 %205 = or i32 %204, 8%206 = insertelement <8 x i32> undef, i32 %205, i64 0 %207 = shufflevector <8 x i32> %206, <8 x i32> undef, <8 x i32> ... zeroinitializer %208 = or < 8 x i32 > %207, < i32 192, i32 193, i32 194, i32 195, i32 196, i32... 197, i32 198, i32 199> %209 = icmp sgt <8 x i32> %broadcast.splat11, %203 %210 = icmp sgt <8 x i32> %broadcast.splat13, %208 $%211 = \text{extractelement} < 8 \times \text{i} 64 > \%202, \text{i} 32 \text{ 0}$ %212 = shl i64 %211, 32 %213 = ashr exact i64 %212, 32 %214 = getelementptr inbounds float, float* %2, i64 %213 %215 = bitcast float* %214 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x $\,$... float>* %215, i32 4, <8 x i1> %209), !tbaa !12, !llvm.access.group !16 %216 = getelementptr inbounds float, float* %214, i64 8 %217 = bitcast float* %216 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %217, i32 4, <8 x i1> %210), !tbaa !12, !llvm.access.group !16 %218 = or <8 x i64> %broadcast.splat, <i64 208, i64 209, i64 210, i64 211, ... i64 212, i64 213, i64 214, i64 215> %219 = trunc <8 x i64> %218 to <8 x i32> %220 = trunc i64 %mul.i.i to i32 %221 = or i32 %220, 8%222 = insertelement <8 x i32> undef, i32 %221, i64 0 %223 = shufflevector <8 x i32> %222, <8 x i32> undef, <8 x i32> ... zeroinitializer $\%224 = \text{or} < 8 \times i32 > \%223$, $< i32\ 208$, $i32\ 209$, $i32\ 210$, $i32\ 211$, $i32\ 212$, i32... 213, i32 214, i32 215> %225 = icmp sgt <8 x i32> %broadcast.splat11, %219 %226 = icmp sgt <8 x i32> %broadcast.splat13, %224 $%227 = \text{extractelement} < 8 \times i64 > %218, i32 0$ %228 = shl i64 %227, 32%229 = ashr exact i64 %228, 32 %230 = getelementptr inbounds float, float* %2, i64 %229 %231 = bitcast float* %230 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %231, i32 4, <8 x i1> %225), !tbaa !12, !llvm.access.group !16 %232 = getelementptr inbounds float, float* %230, i64 8 %233 = bitcast float* %232 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %233, i32 4, <8 x i1> %226), !tbaa !12, !llvm.access.group !16 %234 = or <8 x i64> %broadcast.splat, <i64 224, i64 225, i64 226, i64 227, ... i64 228, i64 229, i64 230, i64 231> %235 = trunc <8 x i64> %234 to <8 x i32> %236 = trunc i64 %mul.i.i to i32 %237 = or i32 %236, 8%238 = insertelement <8 x i32> undef, i32 %237, i64 0 %239 = shufflevector <8 x i32> %238, <8 x i32> undef, <8 x i32> ... zeroinitializer $\%240 = \text{or} < 8 \times i32 > \%239$, $< i32\ 224$, $i32\ 225$, $i32\ 226$, $i32\ 227$, $i32\ 228$, i32... 229, i32 230, i32 231> %241 = icmp sgt <8 x i32> %broadcast.splat11, %235 %242 = icmp sgt <8 x i32> %broadcast.splat13, %240 $%243 = \text{extractelement} < 8 \times i64 > %234, i32 0$ %244 = shl i64 %243, 32 %245 = ashr exact i64 %244, 32%246 = getelementptr inbounds float, float* %2, i64 %245 %247 = bitcast float* %246 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %247, i32 4, <8 x i1> %241), !tbaa !12, !llvm.access.group !16 %248 = getelementptr inbounds float, float* %246, i64 8 %249 = bitcast float* %248 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %249, i32 4, <8 x i1> %242), !tbaa !12, !llvm.access.group !16 %250 = or <8 x i64> %broadcast.splat, <i64 240, i64 241, i64 242, i64 243, ... i64 244, i64 245, i64 246, i64 247> %251 = trunc <8 x i64> %250 to <8 x i32> %252 = trunc i64 %mul.i.i to i32 %253 = or i32 %252, 8%254 = insertelement <8 x i32> undef, i32 %253, i64 0 %255 = shufflevector <8 x i32> %254, <8 x i32> undef, <8 x i32> ... zeroinitializer %256 = or < 8 x i32 > %255, < i32 240, i32 241, i32 242, i32 243, i32 244, i32... 245, i32 246, i32 247> %257 = icmp sgt <8 x i32> %broadcast.splat11, %251 %258 = icmp sgt <8 x i32> %broadcast.splat13, %256 $%259 = \text{extractelement} < 8 \times i64 > %250, i32 0$ %260 = shl i64 %259, 32%261 = ashr exact i64 %260, 32 %262 = getelementptr inbounds float, float* %2, i64 %261 %263 = bitcast float* %262 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %263, i32 4, <8 x i1> %257), !tbaa !12, !llvm.access.group !16 %264 = getelementptr inbounds float, float* %262, i64 8 %265 = bitcast float* %264 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x ... float>* %265, i32 4, <8 x i1> %258), !tbaa !12, !llvm.access.group !16 br label %bicgKernel1.exit 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* %2, i64 %idxprom.i.us store float 0.000000e+00, float* %arrayidx.i.us, align 4, !tbaa !12, ...!llvm.access.group!16 %mul.i.us = mul nsw i32 %conv.i.us, %4 %266 = sext i32 %mul.i.us to i64 br label %for.body.i.us for.body.i.us: %indvars.iv.next.i2.us = phi i64 [%indvars.iv.next.i.us, %for.body.i.us], ... [0, %if.then.i.us] %267 = phi float [%271, %for.body.i.us], [0.000000e+00, %if.then.i.us] %268 = add nsw i64 %indvars.iv.next.i2.us, %266 %arrayidx5.i.us = getelementptr inbounds float, float* %0, i64 %268 %269 = load float, float* %arrayidx5.i.us, align 4, !tbaa !12 %arrayidx7.i.us = getelementptr inbounds float, float* %1, i64 ... %indvars.iv.next.i2.us %270 = load float, float* %arrayidx7.i.us, align 4, !tbaa !12 %271 = tail call float @llvm.fmuladd.f32(float %269, float %270, float %267) ... #2 store float %271, float* %arrayidx.i.us, align 4, !tbaa !12, ...!llvm.access.group!16 %indvars.iv.next.i.us = add nuw nsw i64 %indvars.iv.next.i2.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 %if.end.i.us.loopexit, label %for.body.i.us, ...!llvm.loop!18 F if.end.i.us.loopexit: br label %if.end.i.us if.end.i.us: %272 = or i64 % local id x.0.us, 1%add1.i.i.us.1 = add nuw nsw i64 %272, %mul.i.i %conv.i.us.1 = trunc i64 %add1.i.i.us.1 to i32 %cmp.i.us.1 = icmp slt i32 %conv.i.us.1, %3 br i1 %cmp.i.us.1, label %if.then.i.us.1, label %if.end.i.us.1 if.then.i.us.1: %sext.i.us.1 = shl i64 %add1.i.i.us.1, 32 %idxprom.i.us.1 = ashr exact i64 %sext.i.us.1, 32 %arrayidx.i.us.1 = getelementptr inbounds float, float* %2, i64 ... %idxprom.i.us.1 store float 0.000000e+00, float* %arrayidx.i.us.1, align 4, !tbaa !12, ...!llvm.access.group!16 %mul.i.us.1 = mul nsw i32 %conv.i.us.1, %4 %273 = sext i32 %mul.i.us.1 to i64 br label %for.body.i.us.1 for.body.i.us.1: %indvars.iv.next.i2.us.1 = phi i64 [%indvars.iv.next.i.us.1, ... %for.body.i.us.1], [0, %if.then.i.us.1] %274 = phi float [%278, %for.body.i.us.1], [0.000000e+00, %if.then.i.us.1 %275 = add nsw i64 %indvars.iv.next.i2.us.1, %273 %arrayidx5.i.us.1 = getelementptr inbounds float, float* %0, i64 %275 %276 = load float, float* %arrayidx5.i.us.1, align 4, !tbaa !12 %arrayidx7.i.us.1 = getelementptr inbounds float, float* %1, i64 ... %indvars.iv.next.i2.us.1 %277 = load float, float* %arrayidx7.i.us.1, align 4, !tbaa !12 %278 = tail call float @llvm.fmuladd.f32(float %276, float %277, float %274) ... #2 store float %278, float* %arrayidx.i.us.1, align 4, !tbaa !12, .. !llvm.access.group !16 %indvars.iv.next.i.us.1 = add nuw nsw i64 %indvars.iv.next.i2.us.1, 1 %exitcond.not.i.us.1 = icmp eq i64 %indvars.iv.next.i.us.1, .. %wide.trip.count.i br i1 %exitcond.not.i.us.1, label %if.end.i.us.1.loopexit, label ... %for.body.i.us.1, !llvm.loop !18 F if.end.i.us.1.loopexit: br label %if.end.i.us.1 if.end.i.us.1: %279 = add nuw nsw i64 % local id x.0.us, 2%exitcond.not.1 = icmp eq $\overline{i}64$ % $\overline{2}79$, 256 br i1 %exitcond.not.1, label %bicgKernel1.exit.loopexit, label ... %pregion for entry.entry.i.us, !llvm.loop!20 F bicgKernel1.exit.loopexit: br label %bicgKernel1.exit bicgKernel1.exit: ret void CFG for 'pocl kernel bicgKernel1' function