```
%0:
%tmp2.i1 = shl i64 %6, 8
%tmp2.i1 = sub nsw i32 %3, 0
%10 = sext i32 %4 to i64
%11 = sext i32 %5 to i64
br i1 %tmp2.i1, label %pregion_for_entry.entry.i.no_preheader, label
  %vector.ph
```

```
pregion_for_entry.entry.i.no_preheader:
  br label %pregion_for_entry.entry.i.i
```

```
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> zerominitializer ;
%broadcast.splatinsert111 = insertelement <8 x i32> undef, i32 %4, i32 0 ;
%broadcast.splat112 = shufflevector <8 x i32> %broadcast.splatinsert111, <8 x
  i32> undef, <8 x i32> zerominitializer ;
%broadcast.splatinsert113 = insertelement <8 x i32> undef, i32 %4, i32 0 ;
%broadcast.splat114 = shufflevector <8 x i32> %broadcast.splatinsert113, <8 x
  i32> undef, <8 x i32> zerominitializer ;
%73 = or <8 x i64> %broadcast.splat, <64 0, 64 1, 64 2, 64 3, 64 4, 64
  5, 64 6, 64 7> ;
%12 = trunc <8 x i64> %11 to <8 x i32>, !llvm.access.group 112 ;
%13 = trunc i64 %10 to i32 ;
%14 = or i32 %13, 8 ;
%15 = insertelement <8 x i32> undef, i32 %14, i64 0 ;
%16 = shufflevector <8 x i32> %15, <8 x i32> undef, <8 x i32> zerominitializer ;
%17 = or <8 x i32> %16, <32 0, 32 1, 32 2, 32 3, 32 4, 32 5, 32 6,
  32 7> ;
%18 = icmp sgt <8 x i32> %broadcast.splat112, %12, !llvm.access.group 112 ;
%19 = icmp sgt <8 x i32> %broadcast.splat114, %17, !llvm.access.group 112 ;
%20 = extractelement <8 x i64> %11, i32 0 ;
%21 = add i64 %20, 1 ;
%22 = add nsw i64 %21, 32, !llvm.access.group 112 ;
%23 = getelementptr inbounds float, float* %2, i64 %22, !llvm.access.group
  112 ;
%24 = bitcast float* %23 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %24, i32 4, <8 x i1> %18), !llvm.access.group 112 ;
%25 = getelementptr inbounds float, float* %23, i64 8 ;
%26 = bitcast float* %25 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %26, i32 4, <8 x i1> %19), !llvm.access.group 112 ;
%27 = or <8 x i64> %broadcast.splat, <64 16, 64 17, 64 18, 64 19, 64
  20, 64 21, 64 22, 64 23> ;
%28 = trunc <8 x i64> %27 to <8 x i32>, !llvm.access.group 112 ;
%29 = trunc i64 %mul.i.i to i32 ;
%30 = add i32 %29, 8 ;
%31 = insertelement <8 x i32> undef, i32 %30, i64 0 ;
%32 = shufflevector <8 x i32> %31, <8 x i32> undef, <8 x i32> zerominitializer ;
%33 = or <8 x i32> %32, <32 16, 32 17, 32 18, 32 19, 32 20, 32 21, 32
  22, 32 23> ;
%34 = icmp sgt <8 x i32> %broadcast.splat112, %28, !llvm.access.group 112 ;
%35 = icmp sgt <8 x i32> %broadcast.splat114, %33, !llvm.access.group 112 ;
%36 = extractelement <8 x i64> %27, i32 0 ;
%37 = add i64 %36, 1 ;
%38 = add nsw i64 %37, 32, !llvm.access.group 112 ;
%39 = getelementptr inbounds float, float* %2, i64 %38, !llvm.access.group
  112 ;
%40 = bitcast float* %39 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %40, i32 4, <8 x i1> %34), !llvm.access.group 112 ;
%41 = getelementptr inbounds float, float* %39, i64 8 ;
%42 = bitcast float* %41 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %42, i32 4, <8 x i1> %35), !llvm.access.group 112 ;
%43 = or <8 x i64> %broadcast.splat, <64 32, 64 33, 64 34, 64 35, 64
  36, 64 37, 64 38, 64 39> ;
%44 = trunc <8 x i64> %43 to <8 x i32>, !llvm.access.group 112 ;
%45 = trunc i64 %mul.i.i to i32 ;
%46 = add i32 %45, 8 ;
%47 = insertelement <8 x i32> undef, i32 %46, i64 0 ;
%48 = shufflevector <8 x i32> %47, <8 x i32> undef, <8 x i32> zerominitializer ;
%49 = or <8 x i32> %48, <32 32, 32 33, 32 34, 32 35, 32 36, 32 37, 32
  38, 32 39> ;
%50 = icmp sgt <8 x i32> %broadcast.splat112, %44, !llvm.access.group 112 ;
%51 = icmp sgt <8 x i32> %broadcast.splat114, %49, !llvm.access.group 112 ;
%52 = extractelement <8 x i64> %43, i32 0 ;
%53 = add nsw %52, 32, !llvm.access.group 112 ;
%54 = sext i64 %53, 32, !llvm.access.group 112 ;
%55 = getelementptr inbounds float, float* %2, i64 %54, !llvm.access.group
  112 ;
%56 = bitcast float* %55 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %56, i32 4, <8 x i1> %50), !llvm.access.group 112 ;
%57 = getelementptr inbounds float, float* %55, i64 8 ;
%58 = bitcast float* %57 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %58, i32 4, <8 x i1> %51), !llvm.access.group 112 ;
%59 = or <8 x i64> %broadcast.splat, <64 48, 64 49, 64 50, 64 51, 64
  52, 64 53, 64 54, 64 55> ;
%60 = trunc <8 x i64> %59 to <8 x i32>, !llvm.access.group 112 ;
%61 = trunc i64 %mul.i.i to i32 ;
%62 = add i32 %61, 8 ;
%63 = insertelement <8 x i32> undef, i32 %62, i64 0 ;
%64 = shufflevector <8 x i32> %63, <8 x i32> undef, <8 x i32> zerominitializer ;
%65 = or <8 x i32> %64, <32 48, 32 49, 32 50, 32 51, 32 52, 32 53,
  32 54, 32 55> ;
%66 = icmp sgt <8 x i32> %broadcast.splat112, %60, !llvm.access.group 112 ;
%67 = icmp sgt <8 x i32> %broadcast.splat114, %65, !llvm.access.group 112 ;
%68 = extractelement <8 x i64> %59, i32 0 ;
%69 = add %68, 32, !llvm.access.group 112 ;
%70 = sext i64 %69, 32, !llvm.access.group 112 ;
%71 = getelementptr inbounds float, float* %2, i64 %70, !llvm.access.group
  112 ;
%72 = bitcast float* %71 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %72, i32 4, <8 x i1> %66), !llvm.114, !llvm.access.group 112 ;
%73 = getelementptr inbounds float, float* %71, i64 8 ;
%74 = bitcast float* %73 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %74, i32 4, <8 x i1> %67), !llvm.114, !llvm.access.group 112 ;
%75 = or <8 x i64> %broadcast.splat, <64 64, 64 65, 64 66, 64 67, 64
  68, 64 69, 64 70, 64 71> ;
%76 = trunc <8 x i64> %75 to <8 x i32>, !llvm.access.group 112 ;
%77 = trunc i64 %mul.i.i to i32 ;
%78 = add i32 %77, 8 ;
%79 = insertelement <8 x i32> undef, i32 %78, i64 0 ;
%80 = shufflevector <8 x i32> %79, <8 x i32> undef, <8 x i32> zerominitializer ;
%81 = or <8 x i32> %80, <32 64, 32 65, 32 66, 32 67, 32 68, 32 69, 32
  70, 32 71> ;
%82 = icmp sgt <8 x i32> %broadcast.splat112, %76, !llvm.access.group 112 ;
%83 = icmp sgt <8 x i32> %broadcast.splat114, %81, !llvm.access.group 112 ;
%84 = extractelement <8 x i64> %75, i32 0 ;
%85 = add %84, 32, !llvm.access.group 112 ;
%86 = sext i64 %85, 32, !llvm.access.group 112 ;
%87 = getelementptr inbounds float, float* %2, i64 %86, !llvm.access.group
  112 ;
%88 = bitcast float* %87 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %88, i32 4, <8 x i1> %82), !llvm.114, !llvm.access.group 112 ;
%89 = getelementptr inbounds float, float* %87, i64 8 ;
%90 = bitcast float* %89 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %90, i32 4, <8 x i1> %83), !llvm.114, !llvm.access.group 112 ;
%91 = or <8 x i64> %broadcast.splat, <64 80, 64 81, 64 82, 64 83, 64
  84, 64 85, 64 86, 64 87> ;
%92 = trunc <8 x i64> %91 to <8 x i32>, !llvm.access.group 112 ;
%93 = trunc i64 %mul.i.i to i32 ;
%94 = or i32 %93, 8 ;
%95 = insertelement <8 x i32> undef, i32 %94, i64 0 ;
%96 = shufflevector <8 x i32> %95, <8 x i32> undef, <8 x i32> zerominitializer ;
%97 = or <8 x i32> %96, <32 80, 32 81, 32 82, 32 83, 32 84, 32 85, 32
  86, 32 87> ;
%98 = icmp sgt <8 x i32> %broadcast.splat112, %92, !llvm.access.group 112 ;
%99 = icmp sgt <8 x i32> %broadcast.splat114, %97, !llvm.access.group 112 ;
%100 = extractelement <8 x i64> %91, i32 0 ;
%101 = add i64 %100, 32, !llvm.access.group 112 ;
%102 = sext i64 %101, 32, !llvm.access.group 112 ;
%103 = getelementptr inbounds float, float* %2, i64 %102, !llvm.access.group
  112 ;
%104 = bitcast float* %103 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %104, i32 4, <8 x i1> %98), !llvm.114, !llvm.access.group 112 ;
%105 = getelementptr inbounds float, float* %103, i64 8 ;
%106 = bitcast float* %105 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %106, i32 4, <8 x i1> %99), !llvm.114, !llvm.access.group 112 ;
%107 = or <8 x i64> %broadcast.splat, <64 96, 64 97, 64 98, 64 99, 64
  100, 64 101, 64 102, 64 103> ;
%108 = trunc <8 x i64> %107 to <8 x i32>, !llvm.access.group 112 ;
%109 = trunc i64 %mul.i.i to i32 ;
%110 = or i32 %109, 8 ;
%111 = insertelement <8 x i32> undef, i32 %110, i64 0 ;
%112 = shufflevector <8 x i32> %111, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%113 = or <8 x i32> %112, <32 96, 32 97, 32 98, 32 99, 32 100, 32 101,
  32 102, 32 103> ;
%114 = icmp sgt <8 x i32> %broadcast.splat112, %108, !llvm.access.group 112 ;
%115 = icmp sgt <8 x i32> %broadcast.splat114, %113, !llvm.access.group 112 ;
%116 = extractelement <8 x i64> %107, i32 0 ;
%117 = add i64 %116, 32, !llvm.access.group 112 ;
%118 = sext i64 %117, 32, !llvm.access.group 112 ;
%119 = getelementptr inbounds float, float* %2, i64 %118, !llvm.access
  group 112 ;
%120 = bitcast float* %119 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %120, i32 4, <8 x i1> %114), !llvm.114, !llvm.access.group 112 ;
%121 = getelementptr inbounds float, float* %119, i64 8 ;
%122 = bitcast float* %121 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %122, i32 4, <8 x i1> %115), !llvm.114, !llvm.access.group 112 ;
%123 = or <8 x i64> %broadcast.splat, <64 112, 64 113, 64 114, 64 115,
  64 116, 64 117, 64 118, 64 119> ;
%124 = trunc <8 x i64> %123 to <8 x i32>, !llvm.access.group 112 ;
%125 = trunc i64 %mul.i.i to i32 ;
%126 = add i32 %125, 8 ;
%127 = insertelement <8 x i32> undef, i32 %126, i64 0 ;
%128 = shufflevector <8 x i32> %127, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%129 = or <8 x i32> %128, <32 112, 32 113, 32 114, 32 115, 32 116, 32
  117, 32 118, 32 119> ;
%130 = icmp sgt <8 x i32> %broadcast.splat112, %124, !llvm.access.group 112 ;
%131 = icmp sgt <8 x i32> %broadcast.splat114, %129, !llvm.access.group 112 ;
%132 = extractelement <8 x i64> %123, i32 0 ;
%133 = add i64 %132, 32, !llvm.access.group 112 ;
%134 = sext i64 %133, 32, !llvm.access.group 112 ;
%135 = getelementptr inbounds float, float* %2, i64 %134, !llvm.access
  group 112 ;
%136 = bitcast float* %135 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %136, i32 4, <8 x i1> %130), !llvm.114, !llvm.access.group 112 ;
%137 = getelementptr inbounds float, float* %135, i64 8 ;
%138 = bitcast float* %137 to <8 x float>* ;
%140 = trunc <8 x i64> %139 to <8 x i32>, !llvm.access.group 112 ;
%141 = trunc i64 %mul.i.i to i32 ;
%142 = or i32 %141, 8 ;
%143 = insertelement <8 x i32> undef, i32 %142, i64 0 ;
%144 = shufflevector <8 x i32> %143, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%145 = or <8 x i32> %144, <32 128, 32 129, 32 130, 32 131, 32 132, 32
  133, 32 134, 32 135> ;
%146 = icmp sgt <8 x i32> %broadcast.splat112, %140, !llvm.access.group 112 ;
%147 = icmp sgt <8 x i32> %broadcast.splat114, %145, !llvm.access.group 112 ;
%148 = extractelement <8 x i64> %139, i32 0 ;
%149 = add i64 %148, 32, !llvm.access.group 112 ;
%150 = sext i64 %149, 32, !llvm.access.group 112 ;
%151 = getelementptr inbounds float, float* %2, i64 %150, !llvm.access
  group 112 ;
%152 = bitcast float* %151 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %152, i32 4, <8 x i1> %146), !llvm.114, !llvm.access.group 112 ;
%153 = getelementptr inbounds float, float* %151, i64 8 ;
%154 = bitcast float* %153 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %154, i32 4, <8 x i1> %147), !llvm.114, !llvm.access.group 112 ;
%155 = or <8 x i64> %broadcast.splat, <64 144, 64 145, 64 146, 64 147,
  64 148, 64 149, 64 150, 64 151> ;
%156 = trunc <8 x i64> %155 to <8 x i32>, !llvm.access.group 112 ;
%157 = trunc i64 %mul.i.i to i32 ;
%158 = or i32 %157, 8 ;
%159 = insertelement <8 x i32> undef, i32 %158, i64 0 ;
%160 = shufflevector <8 x i32> %159, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%161 = or <8 x i32> %160, <32 144, 32 145, 32 146, 32 147, 32 148, 32
  149, 32 150, 32 151> ;
%162 = icmp sgt <8 x i32> %broadcast.splat112, %156, !llvm.access.group 112 ;
%163 = icmp sgt <8 x i32> %broadcast.splat114, %161, !llvm.access.group 112 ;
%164 = extractelement <8 x i64> %155, i32 0 ;
%165 = add i64 %164, 32, !llvm.access.group 112 ;
%166 = sext i64 %165, 32, !llvm.access.group 112 ;
%167 = getelementptr inbounds float, float* %2, i64 %166, !llvm.access
  group 112 ;
%168 = bitcast float* %167 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %168, i32 4, <8 x i1> %162), !llvm.114, !llvm.access.group 112 ;
%169 = getelementptr inbounds float, float* %167, i64 8 ;
%170 = bitcast float* %169 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %170, i32 4, <8 x i1> %163), !llvm.114, !llvm.access.group 112 ;
%171 = or <8 x i64> %broadcast.splat, <64 160, 64 161, 64 162, 64 163,
  64 164, 64 165, 64 166, 64 167> ;
%172 = trunc <8 x i64> %171 to <8 x i32>, !llvm.access.group 112 ;
%173 = trunc i64 %mul.i.i to i32 ;
%174 = or i32 %173, 8 ;
%175 = insertelement <8 x i32> undef, i32 %174, i64 0 ;
%176 = shufflevector <8 x i32> %175, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%177 = or <8 x i32> %176, <32 160, 32 161, 32 162, 32 163, 32 164, 32
  165, 32 166, 32 167> ;
%178 = icmp sgt <8 x i32> %broadcast.splat112, %172, !llvm.access.group 112 ;
%179 = icmp sgt <8 x i32> %broadcast.splat114, %177, !llvm.access.group 112 ;
%180 = extractelement <8 x i64> %171, i32 0 ;
%181 = add i64 %180, 32, !llvm.access.group 112 ;
%182 = sext i64 %181, 32, !llvm.access.group 112 ;
%183 = getelementptr inbounds float, float* %2, i64 %182, !llvm.access
  group 112 ;
%184 = bitcast float* %183 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %184, i32 4, <8 x i1> %178), !llvm.114, !llvm.access.group 112 ;
%185 = getelementptr inbounds float, float* %183, i64 8 ;
%186 = bitcast float* %185 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %186, i32 4, <8 x i1> %179), !llvm.114, !llvm.access.group 112 ;
%187 = or <8 x i64> %broadcast.splat, <64 176, 64 177, 64 178, 64 179,
  64 180, 64 181, 64 182, 64 183> ;
%188 = trunc <8 x i64> %187 to <8 x i32>, !llvm.access.group 112 ;
%189 = trunc i64 %mul.i.i to i32 ;
%190 = or i32 %189, 8 ;
%191 = insertelement <8 x i32> undef, i32 %190, i64 0 ;
%192 = shufflevector <8 x i32> %191, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%193 = or <8 x i32> %192, <32 176, 32 177, 32 178, 32 179, 32 180, 32
  181, 32 182, 32 183> ;
%194 = icmp sgt <8 x i32> %broadcast.splat112, %188, !llvm.access.group 112 ;
%195 = icmp sgt <8 x i32> %broadcast.splat114, %193, !llvm.access.group 112 ;
%196 = extractelement <8 x i64> %187, i32 0 ;
%197 = add i64 %196, 32, !llvm.access.group 112 ;
%198 = sext i64 %197, 32, !llvm.access.group 112 ;
%199 = getelementptr inbounds float, float* %2, i64 %198, !llvm.access
  group 112 ;
%200 = bitcast float* %199 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %200, i32 4, <8 x i1> %194), !llvm.114, !llvm.access.group 112 ;
%201 = getelementptr inbounds float, float* %199, i64 8 ;
%202 = bitcast float* %201 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %202, i32 4, <8 x i1> %195), !llvm.114, !llvm.access.group 112 ;
%203 = or <8 x i64> %broadcast.splat, <64 192, 64 193, 64 194, 64 195,
  64 196, 64 197, 64 198, 64 199> ;
%204 = trunc <8 x i64> %203 to <8 x i32>, !llvm.access.group 112 ;
%205 = trunc i64 %mul.i.i to i32 ;
%206 = or i32 %205, 8 ;
%207 = insertelement <8 x i32> undef, i32 %206, i64 0 ;
%208 = shufflevector <8 x i32> %207, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%209 = or <8 x i32> %208, <32 192, 32 193, 32 194, 32 195, 32 196, 32
  197, 32 198, 32 199> ;
%210 = icmp sgt <8 x i32> %broadcast.splat112, %204, !llvm.access.group 112 ;
%211 = icmp sgt <8 x i32> %broadcast.splat114, %209, !llvm.access.group 112 ;
%212 = extractelement <8 x i64> %203, i32 0 ;
%213 = add i64 %212, 32, !llvm.access.group 112 ;
%214 = sext i64 %213, 32, !llvm.access.group 112 ;
%215 = getelementptr inbounds float, float* %2, i64 %214, !llvm.access
  group 112 ;
%216 = bitcast float* %215 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %216, i32 4, <8 x i1> %210), !llvm.114, !llvm.access.group 112 ;
%217 = getelementptr inbounds float, float* %215, i64 8 ;
%218 = bitcast float* %217 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %218, i32 4, <8 x i1> %211), !llvm.114, !llvm.access.group 112 ;
%219 = or <8 x i64> %broadcast.splat, <64 208, 64 209, 64 210, 64 211,
  64 212, 64 213, 64 214, 64 215> ;
%220 = trunc <8 x i64> %219 to <8 x i32>, !llvm.access.group 112 ;
%221 = trunc i64 %mul.i.i to i32 ;
%222 = or i32 %221, 8 ;
%223 = insertelement <8 x i32> undef, i32 %222, i64 0 ;
%224 = shufflevector <8 x i32> %223, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%225 = or <8 x i32> %224, <32 208, 32 209, 32 210, 32 211, 32 212, 32
  213, 32 214, 32 215> ;
%226 = icmp sgt <8 x i32> %broadcast.splat112, %220, !llvm.access.group 112 ;
%227 = icmp sgt <8 x i32> %broadcast.splat114, %225, !llvm.access.group 112 ;
%228 = extractelement <8 x i64> %219, i32 0 ;
%229 = add i64 %228, 32, !llvm.access.group 112 ;
%230 = sext i64 %229, 32, !llvm.access.group 112 ;
%231 = getelementptr inbounds float, float* %2, i64 %230, !llvm.access
  group 112 ;
%232 = bitcast float* %231 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %232, i32 4, <8 x i1> %226), !llvm.114, !llvm.access.group 112 ;
%233 = getelementptr inbounds float, float* %231, i64 8 ;
%234 = bitcast float* %233 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %234, i32 4, <8 x i1> %227), !llvm.114, !llvm.access.group 112 ;
%235 = or <8 x i64> %broadcast.splat, <64 224, 64 225, 64 226, 64 227,
  64 228, 64 229, 64 230, 64 231> ;
%236 = trunc <8 x i64> %235 to <8 x i32>, !llvm.access.group 112 ;
%237 = trunc i64 %mul.i.i to i32 ;
%238 = or i32 %237, 8 ;
%239 = insertelement <8 x i32> undef, i32 %238, i64 0 ;
%240 = shufflevector <8 x i32> %239, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%241 = or <8 x i32> %240, <32 224, 32 225, 32 226, 32 227, 32 228, 32
  229, 32 230, 32 231> ;
%242 = icmp sgt <8 x i32> %broadcast.splat112, %236, !llvm.access.group 112 ;
%243 = icmp sgt <8 x i32> %broadcast.splat114, %241, !llvm.access.group 112 ;
%244 = extractelement <8 x i64> %235, i32 0 ;
%245 = add i64 %244, 32, !llvm.access.group 112 ;
%246 = sext i64 %245, 32, !llvm.access.group 112 ;
%247 = getelementptr inbounds float, float* %2, i64 %246, !llvm.access.group
  112 ;
%248 = bitcast float* %247 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %248, i32 4, <8 x i1> %242), !llvm.114, !llvm.access.group 112 ;
%249 = getelementptr inbounds float, float* %247, i64 8 ;
%250 = bitcast float* %249 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %250, i32 4, <8 x i1> %243), !llvm.114, !llvm.access.group 112 ;
%251 = or <8 x i64> %broadcast.splat, <64 240, 64 241, 64 242, 64 243,
  64 244, 64 245, 64 246, 64 247> ;
%252 = trunc <8 x i64> %251 to <8 x i32>, !llvm.access.group 112 ;
%253 = trunc i64 %mul.i.i to i32 ;
%254 = or i32 %253, 8 ;
%255 = insertelement <8 x i32> undef, i32 %254, i64 0 ;
%256 = shufflevector <8 x i32> %255, <8 x i32> undef, <8 x i32>
  zerominitializer ;
%257 = or <8 x i32> %256, <32 240, 32 241, 32 242, 32 243, 32 244, 32
  245, 32 246, 32 247> ;
%258 = icmp sgt <8 x i32> %broadcast.splat112, %252, !llvm.access.group 112 ;
%259 = icmp sgt <8 x i32> %broadcast.splat114, %257, !llvm.access.group 112 ;
%260 = extractelement <8 x i64> %251, i32 0 ;
%261 = add i64 %260, 32, !llvm.access.group 112 ;
%262 = sext i64 %261, 32, !llvm.access.group 112 ;
%263 = getelementptr inbounds float, float* %2, i64 %262, !llvm.access.group
  112 ;
%264 = bitcast float* %263 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %264, i32 4, <8 x i1> %258), !llvm.114, !llvm.access.group 112 ;
%265 = getelementptr inbounds float, float* %263, i64 8 ;
%266 = bitcast float* %265 to <8 x float>* ;
call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> zeroinitializer, <8 x
  float>* %266, i32 4, <8 x i1> %259), !llvm.114, !llvm.access.group 112 ;
br label %loopkernel2.exit
```

```
if.then.i.no:
%cmp1.i.no = shl i64 %add1.i.i.no, 32, !llvm.access.group 112
%sext.i.no = ashr exact i64 %cmp1.i.no, 32, !llvm.access.group 112
%arrayidx.i.no = getelementptr inbounds float, float* %2, i64 %sext.i.no,
  !llvm.access.group 112
%arrayidx.i.no.i = bitcast float* %arrayidx.i.no, i8* %1, i64 %mul.i.i,
  !llvm.access.group 112
br label %for.body.i.no, !llvm.access.group 112
```

```
for.body.i.no:
%indvars.iv.next.i3.no = phi i64 [ %indvars.iv.next.i.no, %for.body.i.no ],
  [ 0, %if.then.i.no ]
%267 = add i64 %add1.i.i.no, %for.body.i.no.i.1.0.000000e+00, %d.dim.i.no ;
%268 = mul nsw i64 %indvars.iv.next.i3.no, %10, !llvm.access.group 112 ;
%269 = add nsw i64 %268, %indvars.iv.next.i3.no, !llvm.access.group 112 ;
%arrayidx5.i.no = getelementptr inbounds float, float* %3, i64 %269,
  !llvm.access.group 112 ;
%270 = bitcast float* %arrayidx5.i.no, <8 x float>* %arrayidx5.i.no.i, align 4,
  !llvm.114, !llvm.access.group 112 ;
%arrayidx7.i.no = getelementptr inbounds float, float* %11, i64
  %indvars.iv.next.i3.no, !llvm.access.group 112 ;
%271 = load float, float* %arrayidx7.i.no, align 4, !llvm.114,
  !llvm.access.group 112 ;
%272 = bitcast float* %arrayidx7.i.no, <8 x float>* %arrayidx7.i.no.i, align 4,
  !llvm.114, !llvm.access.group 112 ;
%273 = call <8 x float> @llvm.fmuladd.f32(float %270, float %271, float %267)
  #2, !llvm.access.group 112 ;
store float %272, float* %arrayidx.i.no, align 4, !llvm.114,
  !llvm.access.group 112 ;
%indvars.iv.next.i.no = add nsw i64 %indvars.iv.next.i3.no, 1,
  !llvm.access.group 112 ;
%exitcond.i.no = icmp eq i64 %indvars.iv.next.i.no, %wide.trip.count.i,
  !llvm.access.group 112 ;
br i1 %exitcond.next.i.no, label %if.end.i.no.loopexit, label %for.body.i.no,
  !llvm.loop 118, !llvm.access.group 112
```

```
if.end.i.no.loopexit:
  br label %if.end.i.no
```

```
if.end.i.no:
%add1.i.i.no = add %, %mul.i.i, i32 0, !llvm.access.group 112
%add1.i.i.no = add nsw nsw nsw i64 %, %add1.i.i.no, !llvm.access.group 112
%cmp.i.no.i = trunc i64 %add1.i.i.no.i.1 to i32, !llvm.access.group 112
br i1 %cmp.i.no.i, label %if.then.i.no, label %if.end.i.no, !llvm.access.group
  112
```

```
if.then.i.no.1:
%cmp1.i.no.1 = shl i64 %add1.i.i.no.1, 32, !llvm.access.group 112
%sext.i.no.1 = ashr exact i64 %cmp1.i.no.1, 32, !llvm.access.group 112
%arrayidx.i.no.1 = getelementptr inbounds float, float* %2, i64
  %sext.i.no.1, !llvm.access.group 112
%arrayidx.i.no.i.1 = bitcast float* %arrayidx.i.no.1, <8 x float>*
  %arrayidx.i.no.i.1, !llvm.access.group 112
br label %for.body.i.no.1, !llvm.access.group 112
```

```
for.body.i.no.1:
%indvars.iv.next.i3.no.1 = phi i64 [ %indvars.iv.next.i.no.1,
  %for.body.i.no.1 ], [ 0, %if.then.i.no.1 ]
%274 = add i64 %add1.i.i.no.1, %for.body.i.no.i.1.1.0.000000e+00, %d.dim.i.no.1
  ;
%275 = mul nsw i64 %indvars.iv.next.i3.no.1, %10, !llvm.access.group 112
%276 = add nsw i64 %275, %indvars.iv.next.i3.no.1, !llvm.access.group 112
%arrayidx5.i.no.1 = getelementptr inbounds float, float* %3, i64 %276,
  !llvm.access.group 112
%277 = load float, float* %arrayidx5.i.no.1, align 4, !llvm.114,
  !llvm.access.group 112
%arrayidx7.i.no.1 = getelementptr inbounds float, float* %11,
  %indvars.iv.next.i3.no.1, !llvm.access.group 112
%278 = load float, float* %arrayidx7.i.no.1, align 4, !llvm.114,
  !llvm.access.group 112
%279 = call float @llvm.fmuladd.f32(float %277, float %278, float %274)
  #2, !llvm.access.group 112
store float %279, float* %arrayidx.i.no.1, align 4, !llvm.114,
  !llvm.access.group 112
%indvars.iv.next.i.no.1 = add nsw nsw i64 %indvars.iv.next.i3.no.1, 1,
  !llvm.access.group 112
%exitcond.i.no.1 = icmp eq i64 %indvars.iv.next.i.no.1, %wide.trip.count.i,
  !llvm.access.group 112
br i1 %exitcond.next.i.no.1, label %if.end.i.no.1.loopexit, label
  %for.body.i.no.1, !llvm.loop 118, !llvm.access.group 112
```

```
if.end.i.no.1.loopexit:
  br label %if.end.i.no.1
```

```
if.end.i.no.1:
%280 = add nsw nsw nsw i64 %, %add1.i.i.no.1, !llvm.access.group 112
%cmp.i.no.i.1 = trunc i64 %add1.i.i.no.i.1 to i32, !llvm.access.group 112
br i1 %exitcond.next.i, label %loopkernel2.exit.loopexit, label
  %pregion_for_entry.entry.i.i, !llvm.loop 120
```

```
loopkernel2.exit.loopexit:
  br label %loopkernel2.exit
```

```
loopkernel2.exit:
  ret void
```

CFG for '__pocl_kernel_loopkernel2' function