%9: %mul.i.i = shl i64 %6, 8 %cmp222.i = icmp sgt i32 %4, 0 %10 = sext i 32 % 3 to i 64%wide.trip.count.i = zext i32 %4 to i64 br i1 %cmp222.i, label %pregion for entry.entry.i.us.preheader, label ... %pregion for entry.entry.i.preheader F pregion for entry.entry.i.us.preheader: br label %pregion for entry.entry.i.us pregion for entry.entry.i.preheader: %div.i = fdiv float 0.000000e+00, %2%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.splatinsert13 = insertelement <8 x i32> undef, i32 %3, i32 0 %broadcast.splat14 = shufflevector <8 x i32> %broadcast.splatinsert13, <8 x ... i32> undef, <8 x i32> zeroinitializer %broadcast.splatinsert15 = insertelement <8 x i32> undef, i32 %3, i32 0 %broadcast.splat16 = shufflevector <8 x i32> %broadcast.splatinsert15, <8 x ... i32> undef, <8 x i32> zeroinitializer %broadcast.splatinsert17 = insertelement <8 x float> undef, float %div.i, ... i32 0 %broadcast.splat18 = shufflevector <8 x float> %broadcast.splatinsert17, <8 ... x float> undef. <8 x i32> zeroinitializer %broadcast.splatinsert19 = insertelement <8 x float> undef, float %div.i, ... i32 0 %broadcast.splat20 = shufflevector <8 x float> %broadcast.splatinsert19, <8 ... x float> undef, <8 x i32> zeroinitializer %11 = or <8 x i64> %broadcast.splat, <i64 0, i64 1, i64 2, i64 3, i64 4, i64 ... 5, i64 6, i64 7> %12 = trunc <8 x i64> %11 to <8 x i32>, !llvm.access.group !12 %13 = trunc i64 %mul.i.i to i32 %14 = or i32 %13, 8%15 = insertelement <8 x i32> undef, i32 %14, i64 0 $%16 = \text{shufflevector} < 8 \times i32 > \%15, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ %17 = or < 8 x i32 > %16, < i32 0, i32 1, i32 2, i32 3, i32 4, i32 5, i32 6,... i32 7> %18 = icmp sqt <8 x i32> %broadcast.splat14, %12, !llvm.access.group !12 %19 = icmp sqt <8 x i32> %broadcast.splat16, %17, !llvm.access.group !12 $%20 = \text{extractelement} < 8 \times i64 > %11, i32 0$ %21 = shl i64 %20, 32, !llvm.access.group !12 %22 = ashr exact i64 %21, 32, !llvm.access.group !12 %23 = getelementptr inbounds float, float* %0, i64 %22, !llvm.access.group ... !12 %24 = bitcast float* %23 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %24, i32 4, <8 x i1> %18), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, <8 x float>* %26, i32 4, <8 x i1> %19), !tbaa !14, !llvm.access.group !12 %27 = or <8 x i64> %broadcast.splat, <i64 16, i64 17, i64 18, i64 19, i64 ... 20, i64 21, i64 22, i64 23> %28 = trunc <8 x i64> %27 to <8 x i32>, !llvm.access.group !12 %29 = trunc i64 %mul.i.i to i32 %30 = or i32 %29, 8%31 = insertelement <8 x i32> undef, i32 %30, i64 0 $\%32 = \text{shufflevector} < 8 \times i32 > \%31, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ %33 = or < 8 x i32 > %32, < i32 16, i32 17, i32 18, i32 19, i32 20, i32 21, i32... 22, i32 23> %34 = icmp sqt <8 x i32> %broadcast.splat14, %28, !llvm.access.group !12 %35 = icmp sgt <8 x i32> %broadcast.splat16, %33, !llvm.access.group !12 $%36 = \text{extractelement} < 8 \times i64 > \%27, i32 0$ %37 = shl i64 %36, 32, !llvm.access.group !12 %38 = ashr exact i64 %37, 32, !llvm.access.group !12 %39 = getelementptr inbounds float, float* %0, i64 %38, !llvm.access.group ... !12 %40 = bitcast float* %39 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %40, i32 4, <8 x i1> %34), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %42, i32 4, <8 x i1> %35), !tbaa !14, !llvm.access.group !12 %43 = or <8 x i64> %broadcast.splat, <i64 32, i64 33, i64 34, i64 35, i64 ... 36, i64 37, i64 38, i64 39> $%44 = trunc < 8 \times i64 > %43 \text{ to } < 8 \times i32 >$, !llvm.access.group !12 %45 = trunc i64 %mul.i.i to i32 %46 = or i32 %45, 8%47 = insertelement <8 x i32> undef, i32 %46, i64 0 $%48 = \text{shufflevector} < 8 \times i32 > \%47, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ %49 = or < 8 x i32 > %48, < i32 32, i32 33, i32 34, i32 35, i32 36, i32 37, i32... 38, i32 39> %50 = icmp sgt <8 x i32> %broadcast.splat14, %44, !llvm.access.group !12 %51 = icmp sgt <8 x i32> %broadcast.splat16, %49, !llvm.access.group !12 $\%52 = \text{extractelement} < 8 \times \text{i}64 > \%43, \text{i}32 \text{ 0}$ %53 = shl i64 %52, 32, !llvm.access.group !12 %54 = ashr exact i64 %53, 32, !llvm.access.group !12 %55 = getelementptr inbounds float, float* %0, i64 %54, !llvm.access.group ... !12 %56 = bitcast float* %55 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %56, i32 4, <8 x i1> %50), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %58, i32 4, <8 x i1> %51), !tbaa !14, !llvm.access.group !12 %59 = or <8 x i64> %broadcast.splat, <i64 48, i64 49, i64 50, i64 51, i64 ... 52, i64 53, i64 54, i64 55> $\%60 = \text{trunc} < 8 \times i64 > \%59 \text{ to} < 8 \times i32 >$, !llvm.access.group !12 %61 = trunc i64 %mul.i.i to i32 %62 = or i32 %61, 8%63 = insertelement <8 x i32> undef, i32 %62, i64 0 %64 = shufflevector < 8 x i 32 > %63, < 8 x i 32 > undef, < 8 x i 32 > zeroinitializer $\%65 = \text{or} < 8 \times 32 > \%64$, $< 32 \times 48$, 32×49 , 32×50 , 32×51 , 32×52 , 32×53 , 32×51 , 32×52 , 32×53 , 32×51 , 32×52 , 32×53 , ... 54, i32 55> %66 = icmp sqt <8 x i32> %broadcast.splat14, %60, !llvm.access.group !12 %67 = icmp sgt <8 x i32> %broadcast.splat16, %65, !llvm.access.group !12 $\%68 = \text{extractelement} < 8 \times i64 > \%59, i32 0$ %69 = shl i64 %68, 32, !llvm.access.group !12 %70 = ashr exact i64 %69, 32, !llvm.access.group !12 %71 = getelementptr inbounds float, float* %0, i64 %70, !llvm.access.group ... !12 %72 = bitcast float* %71 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %72, i32 4, <8 x i1> %66), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %74, i32 4, <8 x i1> %67), !tbaa !14, !llvm.access.group !12 %75 = or <8 x i64> %broadcast.splat, <i64 64, i64 65, i64 66, i64 67, i64 ... 68, i64 69, i64 70, i64 71> $\%76 = \text{trunc} < 8 \times i64 > \%75 \text{ to } < 8 \times i32 >$, !llvm.access.group !12 %77 = trunc i64 %mul.i.i to i32 %78 = or i32 %77, 8%79 = insertelement <8 x i32> undef, i32 %78, i64 0 $\%80 = \text{shufflevector} < 8 \times i32 > \%79, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ $\%81 = \text{or} < 8 \times 32 > \%80$, $< 32 \times 64$, 32×65 , 32×66 , 32×67 , 32×68 , 32×69 , ... 70, i32 71> %82 = icmp sgt <8 x i32> %broadcast.splat14, %76, !llvm.access.group !12 %83 = icmp sqt <8 x i32> %broadcast.splat16, %81, !llvm.access.group !12 $\%84 = \text{extractelement} < 8 \times i64 > \%75, i32 0$ %85 = shl i64 %84, 32, !llvm.access.group !12 %86 = ashr exact i64 %85, 32, !llvm.access.group !12 %87 = getelementptr inbounds float, float* %0, i64 %86, !llvm.access.group ... !12 %88 = bitcast float* %87 to <8 x float>*call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %88, i32 4, <8 x i1> %82), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, .. <8 x float>* %90, i32 4, <8 x i1> \(^{8}x i1> \(^{8}x), !tbaa !14, !llvm.access.group !12 %91 = or <8 x i64> %broadcast.splat, <i64 80, i64 81, i64 82, i64 83, i64 ... 84, i64 85, i64 86, i64 87> %92 = trunc <8 x i64> %91 to <8 x i32>, !llvm.access.group !12 %93 = trunc i64 %mul.i.i to i32 %94 = or i32 %93, 8%95 = insertelement <8 x i32> undef, i32 %94, i64 0 $\%96 = \text{shufflevector} < 8 \times i32 > \%95, < 8 \times i32 > \text{undef}, < 8 \times i32 > \text{zeroinitializer}$ $\%97 = \text{or} < 8 \times 32 > \%96$, $< 32 \times 80$, 32×81 , 32×82 , 32×83 , 32×84 , 32×85 , 32×87 ... 86, i32 87> %98 = icmp sqt <8 x i32> %broadcast.splat14, %92, !llvm.access.group !12 %99 = icmp sgt <8 x i32> %broadcast.splat16, %97, !llvm.access.group !12 $%100 = \text{extractelement} < 8 \times i64 > %91, i32 0$ %101 = shl i64 %100, 32, !llvm.access.group !12 %102 = ashr exact i64 %101, 32, !llvm.access.group !12 %103 = getelementptr inbounds float, float* %0, i64 %102, !llvm.access.group ... !12 %104 = bitcast float* %103 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %104, i32 4, <8 x i1> %98), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %106, i32 4, <8 x i1> %99), !tbaa !14, !llvm.access.group !12 %107 = or <8 x i64> %broadcast.splat, <i64 96, i64 97, i64 98, i64 99, i64 ... 100, i64 101, i64 102, i64 103> $%108 = \text{trunc} < 8 \times i64 > %107 \text{ to } < 8 \times i32 >$, !llvm.access.group !12 %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> ... zeroinitializer %113 = or < 8 x i32 > %112, < i32 96, i32 97, i32 98, i32 99, i32 100, i32 101,... i32 102, i32 103> %114 = icmp sgt <8 x i32> %broadcast.splat14, %108, !llvm.access.group !12 %115 = icmp sgt <8 x i32> %broadcast.splat16, %113, !llvm.access.group !12 %116 = extractelement <8 x i64> %107, i32 0 %117 = shl i64 %116, 32, !llvm.access.group !12 %118 = ashr exact i64 %117, 32, !llvm.access.group !12 %119 = getelementptr inbounds float, float* %0, i64 %118, !llvm.access.group ... !12 %120 = bitcast float* %119 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %120, i32 4, <8 x i1> %114), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %122, i32 4, <8 x i1> %115), !tbaa !14, !llvm.access.group !12 %123 = or <8 x i64> %broadcast.splat, <i64 112, i64 113, i64 114, i64 115, ... i64 116, i64 117, i64 118, i64 119> $%124 = \text{trunc} < 8 \times i64 > %123 \text{ to} < 8 \times i32 >$, !llvm.access.group !12 %125 = trunc i64 %mul.i.i to i32 %126 = or 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> ... zeroinitializer %129 = or < 8 x i32 > %128, < i32 112, i32 113, i32 114, i32 115, i32 116, i32... 117, i32 118, i32 119> %130 = icmp sgt <8 x i32> %broadcast.splat14, %124, !llvm.access.group !12 %131 = icmp sgt <8 x i32> %broadcast.splat16, %129, !llvm.access.group !12 %132 = extractelement <8 x i64> %123, i32 0 pregion for entry.entry.i.us: %133 = shl i64 %132, 32, !llvm.access.group !12 $\sqrt{1000}$ local id x.0.us = phi i64 [%270, %if.end.r exit.i.us], [0, %134 = ashr exact i64 %133, 32, !llvm.access.group !12 ... %pregion for entry.entry.i.us.preheader] %135 = getelementptr inbounds float, float* %0, i64 %134, !llvm.access.group %add1.i.i.us = add nuw nsw i64% local id x.0.us, %mul.i.i,... !12 ..!llvm.access.group!12 %136 = bitcast float* %135 to <8 x float>* %conv.i.us = trunc i64 %add1.i.i.us to i32, !llvm.access.group !12 call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, %cmp.i.us = icmp slt i32 %conv.i.us, %3, !llvm.access.group !12 ... <8 x float>* %136, i32 4, <8 x i1> %130), !tbaa !14, !llvm.access.group !12 br i1 %cmp.i.us, label %if.then.i.us, label %if.end.r exit.i.us, %137 = getelementptr inbounds float, float* %135, i64 8 ...!llvm.access.group!12 %138 = bitcast float* %137 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat20, F ... <8 x float>* %138, i32 4, <8 x i1> %131), !tbaa !14, !llvm.access.group !12 $%139 = \text{or} < 8 \times 164 > \%$ broadcast.splat, $< 164 \times 128$, 164×129 , 164×130 , 164×131 , ... i64 132, i64 133, i64 134, i64 135> $%140 = \text{trunc} < 8 \times i64 > %139 \text{ to} < 8 \times i32 >$, !llvm.access.group !12 %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> ... zeroinitializer %145 = or < 8 x i32 > %144, < i32 128, i32 129, i32 130, i32 131, i32 132, i32... 133, i32 134, i32 135> %146 = icmp sgt <8 x i32> %broadcast.splat14, %140, !llvm.access.group !12 %147 = icmp sgt <8 x i32> %broadcast.splat16, %145, !llvm.access.group !12 $%148 = \text{extractelement} < 8 \times i64 > %139, i32 0$ %149 = shl i64 %148, 32, !llvm.access.group !12 %150 = ashr exact i64 %149, 32, !llvm.access.group !12 %151 = getelementptr inbounds float, float* %0, i64 %150, !llvm.access.group %152 = bitcast float* %151 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %152, i32 4, <8 x i1> %146), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %154, i32 4, <8 x i1> %147), !tbaa !14, !llvm.access.group !12 %155 = or <8 x i64> %broadcast.splat, <i64 144, i64 145, i64 146, i64 147, ... i64 148, i64 149, i64 150, i64 151> $%156 = \text{trunc} < 8 \times i64 > %155 \text{ to} < 8 \times i32 >$, !llvm.access.group !12 %157 = trunc i64 %mul.i.i to i32 %158 = or i32 %157, 8%159 = insertelement <8 x i32> undef, i32 %158, i64 0 $%160 = \text{shufflevector} < 8 \times i32 > %159, < 8 \times i32 > \text{undef}, < 8 \times i32 >$... zeroinitializer $%161 = \text{or} < 8 \times i32 > %160$, < i32 144, i32 145, i32 146, i32 147, i32 148, i32... 149, i32 150, i32 151> %162 = icmp sqt <8 x i32> %broadcast.splat14, %156, !llvm.access.group !12 %163 = icmp sgt <8 x i32> %broadcast.splat16, %161, !llvm.access.group !12 %164 = extractelement <8 x i64> %155, i32 0 %165 = shl i64 %164, 32, !llvm.access.group !12 %166 = ashr exact i64 %165, 32, !llvm.access.group !12 %167 = getelementptr inbounds float, float* %0, i64 %166, !llvm.access.group ... !12 %168 = bitcast float* %167 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %168, i32 4, <8 x i1> %162), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %170, i32 4, <8 x i1> %163), !tbaa !14, !llvm.access.group !12 %171 = or <8 x i64> %broadcast.splat, <i64 160, i64 161, i64 162, i64 163, ... i64 164, i64 165, i64 166, i64 167> $%172 = \text{trunc} < 8 \times i64 > %171 \text{ to } < 8 \times i32 >$, !llvm.access.group !12 %173 = trunc i64 %mul.i.i to i32 %174 = or i32 %173, 8%175 = insertelement <8 x i32> undef, i32 %174, i64 0 $%176 = \text{shufflevector} < 8 \times i32 > \%175, < 8 \times i32 > \text{undef}, < 8 \times i32 >$... zeroinitializer %177 = or < 8 x i32 > %176, $< i32\ 160$, $i32\ 161$, $i32\ 162$, $i32\ 163$, $i32\ 164$, i32... 165, i32 166, i32 167> %178 = icmp sgt <8 x i32> %broadcast.splat14, %172, !llvm.access.group !12 %179 = icmp sgt <8 x i32> %broadcast.splat16, %177, !llvm.access.group !12 $%180 = \text{extractelement} < 8 \times i64 > %171, i32 0$ %181 = shl i64 %180, 32, !llvm.access.group !12 %182 = ashr exact i64 %181, 32, !llvm.access.group !12 %183 = getelementptr inbounds float, float* %0, i64 %182, !llvm.access.group ... !12 %184 = bitcast float* %183 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %184, i32 4, <8 x i1> %178), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %186, i32 4, <8 x i1> %179), !tbaa !14, !llvm.access.group !12 %187 = or <8 x i64> %broadcast.splat, <i64 176, i64 177, i64 178, i64 179, ... i64 180, i64 181, i64 182, i64 183> $%188 = \text{trunc} < 8 \times i64 > %187 \text{ to } < 8 \times i32 >$, !llvm.access.group !12 %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> ... zeroinitializer %193 = or < 8 x i32 > %192, < i32 176, i32 177, i32 178, i32 179, i32 180, i32... 181, i32 182, i32 183> %194 = icmp sgt <8 x i32> %broadcast.splat14, %188, !llvm.access.group !12 %195 = icmp sgt <8 x i32> %broadcast.splat16, %193, !llvm.access.group !12 %196 = extractelement <8 x i64> %187, i32 0 %197 = shl i64 %196, 32, !llvm.access.group !12 %198 = ashr exact i64 %197, 32, !llvm.access.group !12 %199 = getelementptr inbounds float, float* %0, i64 %198, !llvm.access.group ... !12 %200 = bitcast float* %199 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %200, i32 4, <8 x i1> %194), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %202, i32 4, <8 x i1> %195), !tbaa !14, !llvm.access.group !12 %203 = or <8 x i64> %broadcast.splat, <i64 192, i64 193, i64 194, i64 195, ... i64 196, i64 197, i64 198, i64 199> $\%204 = \text{trunc} < 8 \times i64 > \%203 \text{ to} < 8 \times i32 >$, !llvm.access.group !12 %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> ... zeroinitializer %209 = or < 8 x i32 > %208, < i32 192, i32 193, i32 194, i32 195, i32 196, i32... 197, i32 198, i32 199> %210 = icmp sgt <8 x i32> %broadcast.splat14, %204, !llvm.access.group !12 %211 = icmp sgt <8 x i32> %broadcast.splat16, %209, !llvm.access.group !12 %212 = extractelement <8 x i64> %203, i32 0 %213 = shl i64 %212, 32, !llvm.access.group !12 %214 = ashr exact i64 %213, 32, !llvm.access.group !12 %215 = getelementptr inbounds float, float* %0, i64 %214, !llvm.access.group ... !12 %216 = bitcast float* %215 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %216, i32 4, <8 x i1> %210), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %218, i32 4, <8 x i1> %211), !tbaa !14, !llvm.access.group !12 %219 = or <8 x i64> %broadcast.splat, <i64 208, i64 209, i64 210, i64 211, ... i64 212, i64 213, i64 214, i64 215> $\%220 = \text{trunc} < 8 \times i64 > \%219 \text{ to } < 8 \times i32 >$, !llvm.access.group !12 %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> ... zeroinitializer $\%225 = \text{or} < 8 \times i32 > \%224$, $< i32\ 208$, $i32\ 209$, $i32\ 210$, $i32\ 211$, $i32\ 212$, i32... 213, i32 214, i32 215> %226 = icmp sgt <8 x i32> %broadcast.splat14, %220, !llvm.access.group !12 %227 = icmp sgt <8 x i32> %broadcast.splat16, %225, !llvm.access.group !12 %228 = extractelement <8 x i64> %219, i32 0 %229 = shl i64 %228, 32, !llvm.access.group !12 %230 = ashr exact i64 %229, 32, !llvm.access.group !12 %231 = getelementptr inbounds float, float* %0, i64 %230, !llvm.access.group %232 = bitcast float* %231 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %232, i32 4, <8 x i1> %226), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %234, i32 4, <8 x i1> %227), !tbaa !14, !llvm.access.group !12 %235 = or < 8 x i64 > % broadcast.splat, < i64 224, i64 225, i64 226, i64 227,... i64 228, i64 229, i64 230, i64 231> $\%236 = \text{trunc} < 8 \times i64 > \%235 \text{ to} < 8 \times i32 >$, !llvm.access.group !12 %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> ... zeroinitializer $%241 = \text{or} < 8 \times i32 > %240$, < i32 224, i32 225, i32 226, i32 227, i32 228, i32... 229, i32 230, i32 231> %242 = icmp sgt <8 x i32> %broadcast.splat14, %236, !llvm.access.group !12 %243 = icmp sgt <8 x i32> %broadcast.splat16, %241, !llvm.access.group !12 $%244 = \text{extractelement} < 8 \times i64 > %235, i32 0$ %245 = shl i64 %244, 32, !llvm.access.group !12 %246 = ashr exact i64 %245, 32, !llvm.access.group !12 %247 = getelementptr inbounds float, float* %0, i64 %246, !llvm.access.group ... !12 %248 = bitcast float* %247 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %248, i32 4, <8 x i1> %242), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %250, i32 4, <8 x i1> %243), !tbaa !14, !llvm.access.group !12 %251 = or <8 x i64> %broadcast.splat, <i64 240, i64 241, i64 242, i64 243, ... i64 244, i64 245, i64 246, i64 247> %252 = trunc <8 x i64> %251 to <8 x i32>, !llvm.access.group !12 %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> ... zeroinitializer %257 = or < 8 x i32 > %256, < i32 240, i32 241, i32 242, i32 243, i32 244, i32... 245, i32 246, i32 247> %258 = icmp sgt <8 x i32> %broadcast.splat14, %252, !llvm.access.group !12 %259 = icmp sgt <8 x i32> %broadcast.splat16, %257, !llvm.access.group !12 $%260 = \text{extractelement} < 8 \times i64 > %251, i32 0$ %261 = shl i64 %260, 32, !llvm.access.group !12 %262 = ashr exact i64 %261, 32, !llvm.access.group !12 %263 = getelementptr inbounds float, float* %0, i64 %262, !llvm.access.group ... !12 %264 = bitcast float* %263 to <8 x float>* call void @llvm.masked.store.v8f32.p0v8f32(<8 x float> %broadcast.splat18, ... <8 x float>* %264, i32 4, <8 x i1> %258), !tbaa !14, !llvm.access.group !12 %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> %broadcast.splat20, ... <8 x float>* %266, i32 4, <8 x i1> %259), !tbaa !14, !llvm.access.group !12 br label %mean kernel.exit if.then.i.us: %sext.i.us = shl i64 %add1.i.i.us, 32, !llvm.access.group !12 %idxprom.i.us = ashr exact i64 %sext.i.us, 32, !llvm.access.group !12 %arrayidx.i.us = getelementptr inbounds float, float* %0, i64 %idxprom.i.us, ...!llvm.access.group!12 store float 0.000000e+00, float* %arrayidx.i.us, align 4, !tbaa !14, ...!llvm.access.group!12 br label %for.body.i.us, !llvm.access.group !12 for.body.i.us: %indvars.iv.next.i5.us = phi i64 [%indvars.iv.next.i.us, %for.body.i.us], ... [0, %if.then.i.us] %add8.i2.us = phi float [%add8.i.us, %for.body.i.us], [0.000000e+00, ... %if.then.i.us Ī %267 = mul nsw i64 %indvars.iv.next.i5.us, %10, !llvm.access.group !12 %268 = add nsw i64 %267, %idxprom.i.us, !llvm.access.group !12 %arrayidx5.i.us = getelementptr inbounds float, float* %1, i64 %268, ...!llvm.access.group!12 %269 = load float, float* %arrayidx5.i.us, align 4, !tbaa !14, ...!llvm.access.group!12 %add8.i.us = fadd float %add8.i2.us, %269, !llvm.access.group !12 store float %add8.i.us, float* %arrayidx.i.us, align 4, !tbaa !14, ...!llvm.access.group!12 %indvars.iv.next.i.us = add nuw nsw i64 %indvars.iv.next.i5.us, 1, ...!llvm.access.group!12 %exitcond.not.i.us = icmp eq i64 %indvars.iv.next.i.us, %wide.trip.count.i, ...!llvm.access.group!12 br i1 %exitcond.not.i.us, label %for.end.loopexit.i.us, label ... %for.body.i.us, !llvm.loop !18, !llvm.access.group !12 F for.end.loopexit.i.us: %add8.i.us.lcssa = phi float [%add8.i.us, %for.body.i.us] %div.i.us = fdiv float %add8.i.us.lcssa, %2, !fpmath !20, !llvm.access.group store float %div.i.us, float* %arrayidx.i.us, align 4, !tbaa !14, ...!llvm.access.group!12 br label %if.end.r exit.i.us, !llvm.access.group !12 if.end.r exit.i.us: %270 = add nuw nsw i64 % local id x.0.us, 1%exitcond.not = icmp eq $i6\overline{4}$ %270, $\overline{2}56$ br i1 %exitcond.not, label %mean kernel.exit.loopexit, label ... %pregion for entry.entry.i.us, !llvm.loop !21 mean_kernel.exit.loopexit: br label %mean kernel.exit mean_kernel.exit: ret void CFG for 'pocl kernel mean kernel' function