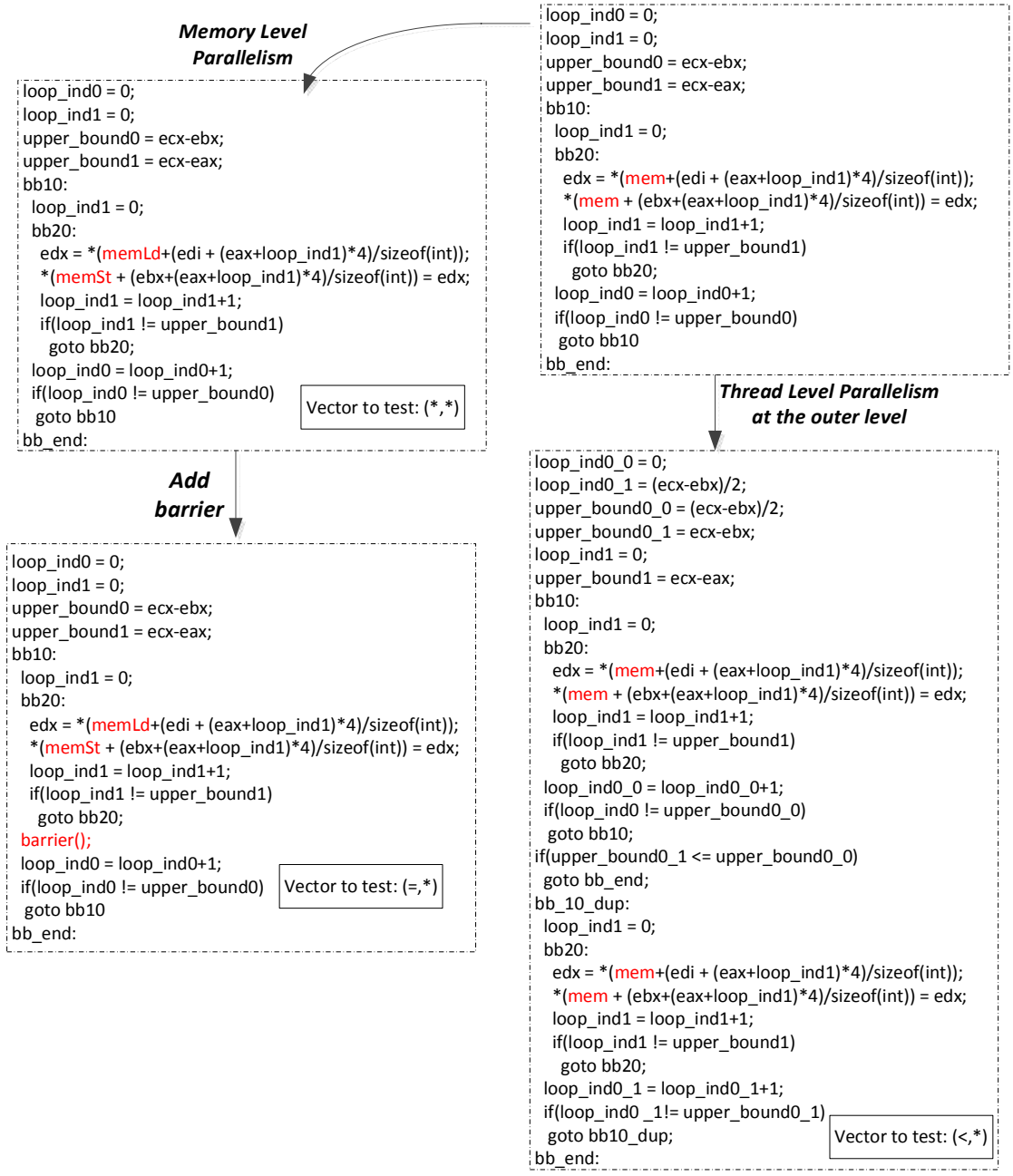
Memory Level Parallelism
loop_ind0 = 0;
loop_ind1 = 0;
upper_bound0 = ecx-ebx;
upper_bound1 = ecx-eax;
bb10:
loop_ind1 = 0;
bb20:
edx = *(memLd+(edi + (eax+loop_ind1)*4)/sizeof(int));
*(memSt + (ebx+(eax+loop_ind1)*4)/sizeof(int)) = edx;
loop_ind1 = loop_ind1+1;
if(loop_ind1 != upper_bound1)
goto bb20;
loop_ind0 = loop_ind0+1;
if(loop_ind0 != upper_bound0)
goto bb10
bb_end:
Vector to test: (*,*)
loop_ind0 = 0;
loop_ind1 = 0;
upper_bound0 = ecx-ebx;
upper_bound1 = ecx-eax;
bb10:
loop_ind1 = 0;
bb20:
edx = *(mem+(edi + (eax+loop_ind1)*4)/sizeof(int));
*(mem + (ebx+(eax+loop_ind1)*4)/sizeof(int)) = edx;
loop_ind1 = loop_ind1+1;
if(loop_ind1 != upper_bound1)
goto bb20;
loop_ind0 = loop_ind0+1;
if(loop_ind0 != upper_bound0)
goto bb10
bb_end:
Thread Level Parallelism at the outer level
Add barrier
loop_ind0 = 0;
loop_ind1 = 0;
upper_bound0 = ecx-ebx;
upper_bound1 = ecx-eax;
bb10:
loop_ind1 = 0;
bb20:
edx = *(memLd+(edi + (eax+loop_ind1)*4)/sizeof(int));
*(memSt + (ebx+(eax+loop_ind1)*4)/sizeof(int)) = edx;
loop_ind1 = loop_ind1+1;
if(loop_ind1 != upper_bound1)
goto bb20;
barrier();
loop_ind0 = loop_ind0+1;
if(loop_ind0 != upper_bound0)
goto bb10
bb_end:
Vector to test: (=,*)
loop_ind0_0 = 0;
loop_ind0_1 = (ecx-ebx)/2;
upper_bound0_0 = (ecx-ebx)/2;
upper_bound0_1 = ecx-ebx;
loop_ind1 = 0;
upper_bound1 = ecx-eax;
bb10:
loop_ind1 = 0;
bb20:
edx = *(mem+(edi + (eax+loop_ind1)*4)/sizeof(int));
*(mem + (ebx+(eax+loop_ind1)*4)/sizeof(int)) = edx;
loop_ind1 = loop_ind1+1;
if(loop_ind1 != upper_bound1)
goto bb20;
loop_ind0_0 = loop_ind0_0+1;
if(loop_ind0 != upper_bound0_0)
goto bb10;
if(upper_bound0_1 <= upper_bound0_0)
goto bb_end;
bb_10_dup:
loop_ind1 = 0;
bb20:
edx = *(mem+(edi + (eax+loop_ind1)*4)/sizeof(int));
*(mem + (ebx+(eax+loop_ind1)*4)/sizeof(int)) = edx;
loop_ind1 = loop_ind1+1;
if(loop_ind1 != upper_bound1)
goto bb20;
loop_ind0_1 = loop_ind0_1+1;
if(loop_ind0 _1!= upper_bound0_1)
goto bb10_dup;
bb_end:
Vector to test: (<,*)