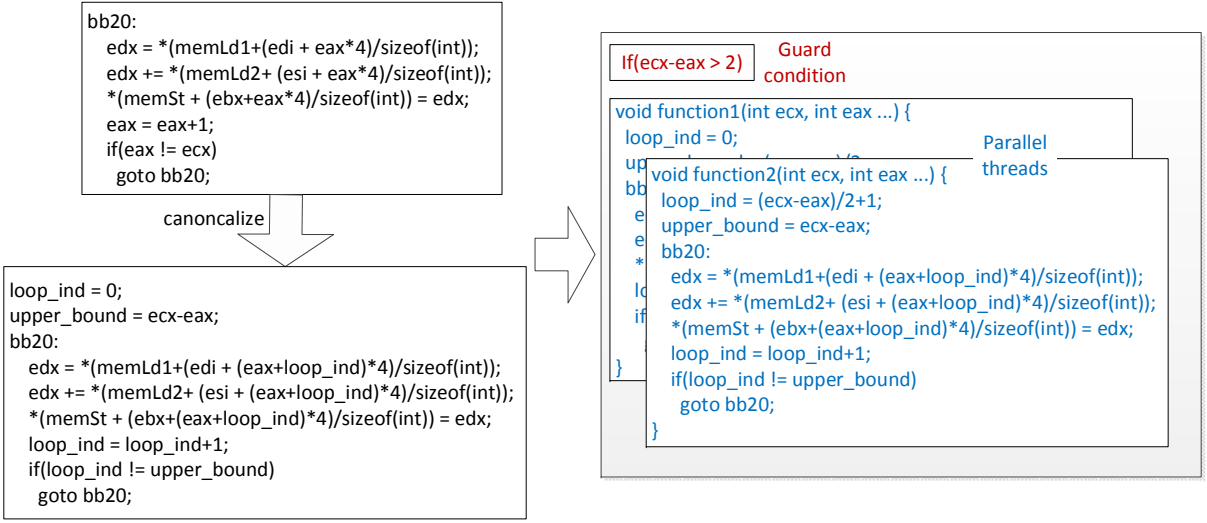
bb20:
edx = *(memLd1+(edi + eax*4)/sizeof(int));
edx += *(memLd2+ (esi + eax*4)/sizeof(int));
*(memSt + (ebx+eax*4)/sizeof(int)) = edx;
eax = eax+1;
if(eax != ecx)
goto bb20;
canoncalize
loop_ind = 0;
upper_bound = ecx-eax;
bb20:
edx = *(memLd1+(edi + (eax+loop_ind)*4)/sizeof(int));
edx += *(memLd2+ (esi + (eax+loop_ind)*4)/sizeof(int));
*(memSt + (ebx+(eax+loop_ind)*4)/sizeof(int)) = edx;
loop_ind = loop_ind+1;
if(loop_ind != upper_bound)
goto bb20;
If(ecx-eax > 2)
Guard condition
void function1(int ecx, int eax ...) {
Parallel threads
void function2(int ecx, int eax ...) {
loop_ind = (ecx-eax)/2+1;
upper_bound = ecx-eax;
bb20:
edx = *(memLd1+(edi + (eax+loop_ind)*4)/sizeof(int));
edx += *(memLd2+ (esi + (eax+loop_ind)*4)/sizeof(int));
*(memSt + (ebx+(eax+loop_ind)*4)/sizeof(int)) = edx;
loop_ind = loop_ind+1;
if(loop_ind != upper_bound)
goto bb20;
}