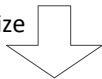


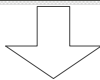
Splitting the Iteration Space into Two

```
bb20:
    edx = *(memLd+(edi + eax*4)/sizeof(int));
    *(memSt + (ebx+eax*4)/sizeof(int)) = edx;
    eax = eax+1;
    if(eax != ecx)
        goto bb20;
end:
```

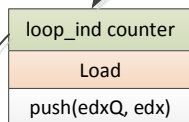
canonicalize



```
loop_ind = 0;
upper_bound = ecx-eax;
bb20:
    edx = *(memLd+(edi + (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;
bb_end:
```

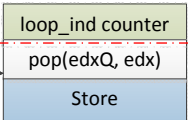


branchTag
token for bb20:



edxQ

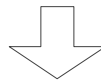
Duplicated loop_ind
counter node in CDFG



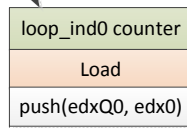
branchTag token
for bb_end:

⋮

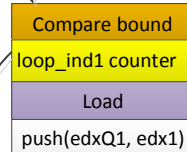
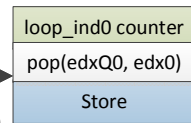
```
loop_ind0 = 0;
upper_bound0 = (ecx-eax)/2;
loop_ind1 = (ecx-eax)/2;
upper_bound1 = ecx-eax;
bb20:
    edx0 = *(memLd+(edi + (eax+loop_ind0)*4)/sizeof(int));
    *(memSt + (ebx+(eax+loop_ind0)*4)/sizeof(int)) = edx0;
    loop_ind0 = loop_ind0+1;
    if(loop_ind0 != upper_bound0)
        goto bb20;
if(upper_bound1 <= upper_bound0)
    goto bb_end;
bb20_dup:
    edx1 = *(memLd+(edi + (eax+loop_ind1)*4)/sizeof(int));
    *(memSt + (ebx+(eax+loop_ind1)*4)/sizeof(int)) = edx1;
    loop_ind1 = loop_ind1+1;
    if(loop_ind1 != upper_bound1)
        goto bb20_dup;
bb_end:
```



branchTag
token for bb20:



edxQ0



branchTag token
for bb_end:

edxQ1

