```
1.  #include <stdio.h>
2.  #include <cuda.h>

3.  #define N 128
4.  #define M 32
5.  #define K 2

6.  __device__ volatile int vint = 0;

7.  __device__ void entry( volatile int* foo )
    {
8.    for (int i = 0; i < N; ++i) {
9.         atomicAdd((int*)foo, 1);
       }
     }
10. extern "C"
11. __global__ void
12. diverge_cta( volatile int *foo )
{
13.     __shared__ int x;
14.     if ((threadIdx.x%32) != 0) {
15.     return;
        }
16.     entry(foo);

17.     if (threadIdx.x == 0) {
18.       x = 5;
19.       return;
       }
20.     __syncthreads();

21.    atomicAdd((int*)foo, x);
   }

22. int main( int argc, char **argv )
{
23.    int *foo;
24.    int h_foo;

25.    cudaMalloc((void**)&foo, sizeof(int));
26.    cudaMemset(foo, 0, sizeof(int));
27.    printf("foo addr: 0x%x\n", (unsigned)(size_t)foo);

28.    diverge_cta<<<K,M*32>>>( foo );
29.    cudaDeviceSynchronize();
30.    cudaMemcpy(&h_foo, foo, sizeof(int), cudaMemcpyDe viceToHost);
31.    if (h_foo == K*(M*N+5*(M-1))) {
32.       printf("simple_scan_test test PASSED\n");
33.    }
34.    else {
35.        printf("Result: %d\n", h_foo);
36.        printf("simple_scan_test test FAILED\n");
37.     }

38.     return 0;
39. }
```