Assume the following piece of code is running on G80 (google to get all info about the G80):

```c
#define VECTOR_N 1024
#define ELEMENT_N 256
const int DATA_N = VECTOR_N * ELEMENT_N;
const int DATA_SZ = DATA_N * sizeof(float);
const int RESULT_SZ = VECTOR_N * sizeof(float);
...
float *d_A, *d_B, *d_C;
...
cudaMalloc((void **) &d_A, DATA_SZ);
cudaMalloc((void **) &d_B, DATA_SZ);
cudaMalloc((void **) &d_C, RESULT_SZ);
...
scalarProd<<<VECTOR_N, ELEMENT_N>>>(d_C, d_A, d_B, ELEMENT_N);

__global__ void scalarProd(float *d_C, float *d_A, float *d_B, int ElementN)
{
    __shared__ float accumResult[ElementN];
    //Current vectors bases
    float *A = d_A + ElementN * blockIdx.x;
    float *B = d_B + ElementN * blockIdx.x;
    int tx = threadIdx.x;
    for (int stride = ElementN / 2; stride > 0; stride >>= 1)
    {
        __syncthreads();
        if (tx < stride)
            accumResult[tx] += accumResult[stride + tx];
    }
    d_C[blockIdx.x] = accumResult[0];
}
```
a. How many threads are there in total?

b. How many threads are there in a warp?

c. How many threads are there in a block?

d. How many global memory loads and stores are done for each thread?

e. How many accesses to shared memory are done for each block?

f. How many iterations of the for loop (Line 23) will have branch divergence? Show your derivation.

g. Identify an opportunity to significantly reduce the bandwidth requirement on the global memory. How would you achieve this? How many accesses can you eliminate?