# Question-02

// Declare host variables

unsigned int numRows, numCols;

// Original CSR

unsigned int \*rowPtrsCSR, \*colIdxCSR;

float \*dataCSR;

// ELL

unsigned int numElemELL;

unsigned int \*colIdxELL;

float \*dataELL;

// COO

unsigned int nnzCOO;

unsigned int \*rowIdxCOO, \*colIdxCOO;

float \*dataCOO;

// Vector

float \*vecSrc, \*vecDst;

// Initialize original host CSR matrix and other variables using the data from Problem 1 of this assignment

numRows = 4;

numCols = 4;

rowPtrsCSR = new unsigned int[] {0,2,3,5,7};

colIdxCSR = new unsigned int[] {0,2,2,1,2,0,3};

dataCSR = new float[] {1,7,8,4,3,2,1};

numElemELL = 1; // pick one element from each row for ELL, rest to COO

nnzCOO = 0; // setting counter to zero

// Transform to hybrid ELL-COO

for(unsigned int rowIdx = 0; rowIdx < numRows; ++rowIdx)

{

unsigned int nnz = rowPtrsCSR[rowIdx + 1] - rowPtrsCSR[rowIdx];

for(unsigned int idx = 0; idx < nnz; ++idx)

{

if(idx < numElemELL)

{

unsigned int ellIdx = idx\*numRows + rowIdx;

colIdxELL[ellIdx] = colIdxCSR[rowPtrsCSR[rowIdx] + idx];

dataELL[ellIdx] = dataCSR[rowPtrsCSR[rowIdx] + idx];

}

else

{

rowIdxCOO[nnzCOO] = rowIdx;

colIdxCOO[nnzCOO] = colIdxCSR[rowPtrsCSR[rowIdx] + idx];

dataCOO[nnzCOO] = dataCSR[rowPtrsCSR[rowIdx] + idx];

++nnzCOO;

}

}

}

// Declare device memory variables

unsigned int \*colIdxELL\_d;

float \*dataELL\_d;

float \*vecSrc\_d, \*vecDst\_d;

// Allocate device memory

cudaMalloc((void\*\*) &colIdxELL\_d, numRows\*numElemELL\*sizeof(unsigned int));

cudaMalloc((void\*\*) &dataELL\_d, numRows\*numElemELL\*sizeof(float));

cudaMalloc((void\*\*) &vecSrc\_d, numCols\*sizeof(float));

cudaMalloc((void\*\*) &vecDst\_d, numRows\*sizeof(float));

// Copy device memory

cudaMemcpy(colIdxELL\_d, colIdxELL, numRows\*numElemELL\*sizeof(unsigned int), cudaMemcpyHostToDevice);

cudaMemcpy(dataELL\_d, dataELL, numRows\*numElemELL\*sizeof(float), cudaMemcpyHostToDevice);

cudaMemcpy(vecSrc\_d, vecSrc, numRows\*sizeof(float), cudaMemcpyHostToDevice);

cudaMemcpy(vecDst\_d, vecDst, numRows\*sizeof(float), cudaMemcpyHostToDevice);

// Launch kernel

SpMV\_ELL <<< (numRows - 1)/BLOCK\_SIZE + 1 , BLOCK\_SIZE >>>(numRows, dataELL\_d, colIdxELL\_d, vecSrc\_d, vecDst\_d);

// Copy device memory back

cudaMemcpy(vecDst, vecDst\_d, numRows\*sizeof(float), cudaMemcpyDevicetoHost);

// Complete COO contributions

for(int i = 0; i< nnzCOO; i++ )

{

vecDst[rowIdxCOO[i]] += dataCOO[i]\*vecSrc[colIdxCOO[i]];

}