```
__global__ void MatrixMulKernel(float *M, float *N, float *P, int n) {
    ...
}

void MatrixMulACC(float *M, float *N, float *P, int n) {
#pragma acc parallel loop deviceptr(M, N, P)
{
    ...
}
}

void matrixMul(float *M, float *N, float *P, int n) {
    unsigned int size = n * n * sizeof(float);
    float *Md = NULL;
    float *Nd = NULL;
    float *Pd = NULL;

    cudaMalloc((void**) &Md, size);
    cudaMalloc((void**) &Nd, size);
    cudaMalloc((void**) &Pd, size);

    cudaMemcpy(Md, M, size, cudaMemcpyHostToDevice);
    cudaMemcpy(Nd, N, size, cudaMemcpyHostToDevice);

    dim3 dimBlock(TILE_WIDTH,TILE_WIDTH);
    dim3 dimGrid(n/TILE_WIDTH, n/TILE_WIDTH);

    // Use CUDA Kernel
    MatrixMulKernel<<<dimGrid, dimBlock>>>(Md, Nd, Pd, n);
    cudaThreadSynchronize();
    cudaMemcpy(P, Pd, size, cudaMemcpyDeviceToHost);

    // Use OpenACC
    MatrixMulACC(Md, Nd, Pd, n);
    cudaMemcpy(P, Pd, size, cudaMemcpyDeviceToHost);

}
```