**Hritik Bansal**

**CSE A 15**

**180905105**

**PCAP WEEK 6**

**Q1)** %%cu

#include<stdio.h>

#include<cuda.h>

#include<stdlib.h>

\_\_global\_\_ void AddRowPerThread(int \*a,int \*b,int \*c,int m,int n)

{

  int id=threadIdx.x;

  if(id<m)

  {

  for(int i=0;i<n;i++)

  {

    c[id\*n+i]=a[id\*n+i]+b[id\*n+i];

  }

  }

}

\_\_global\_\_ void AddColPerThread(int \*a,int \*b,int \*c,int m,int n)

{

  int id=threadIdx.x;

  if(id<n)

  {

    for(int i=0;i<m;i++)

    {

      c[id+i\*n]=a[id+i\*n]+b[id+i\*n];

    }

  }

}

\_\_global\_\_ void AddElePerThread(int \*a,int \*b,int \*c,int m,int n)

{

  int id=threadIdx.x;

  if(id<m\*n)

  {

      c[id]=a[id]+b[id];

  }

}

int main()

{

  int m=3,n=3;

  //declare matrices

  int a[3][3]={{1,1,1},{2,2,2},{3,3,3}};

  int b[3][3]={{1,1,1},{2,2,2},{3,3,3}};

  int c[m][n];

  int \*d\_a,\*d\_b,\*d\_c;

  //allocate memory

  int size=m\*n\*sizeof(int);

  cudaMalloc((void\*\*)&d\_a,size);

  cudaMalloc((void\*\*)&d\_b,size);

  cudaMalloc((void\*\*)&d\_c,size);

  //copy from host to device

  cudaMemcpy(d\_a,a,size,cudaMemcpyHostToDevice);

  cudaMemcpy(d\_b,b,size,cudaMemcpyHostToDevice);

   // Launch add() kernels on GPU

   //Number of blocks=1 threads=m

  printf("ADDITION\n");

   printf("row result per thread:\n");

   AddRowPerThread<<<1,m>>>(d\_a, d\_b, d\_c,m,n);

   cudaMemcpy(c, d\_c,size,cudaMemcpyDeviceToHost);

   for(int i=0;i<m;i++)

   {

    for(int j=0;j<n;j++)

    {

      printf("%d ",c[i][j]);

    }

    printf("\n");

   }

   //Number of blocks=1 threads=n

   printf("column result per thread:\n");

   AddColPerThread<<<1,n>>>(d\_a, d\_b, d\_c,m,n);

   cudaMemcpy(c, d\_c,size,cudaMemcpyDeviceToHost);

   for(int i=0;i<m;i++)

   {

    for(int j=0;j<n;j++)

    {

      printf("%d ",c[i][j]);

    }

    printf("\n");

   }

   //Number of blocks=1 threads=m\*n

   printf("element result per thread:\n");

   AddElePerThread<<<1,m\*n>>>(d\_a, d\_b, d\_c,m,n);

   cudaMemcpy(c, d\_c,size,cudaMemcpyDeviceToHost);

   for(int i=0;i<m;i++)

   {

    for(int j=0;j<n;j++)

    {

      printf("%d ",c[i][j]);

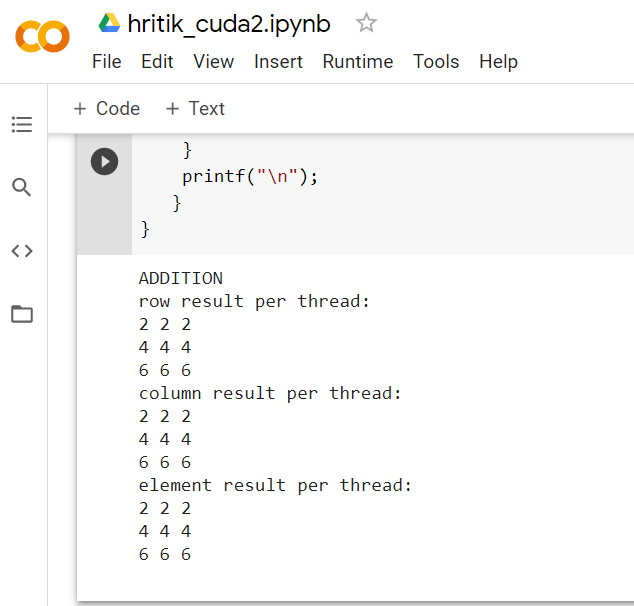
    }

    printf("\n");

   }

}

**OUTPUT:**

****

**Q2)** %%cu

#include<stdio.h>

#include<cuda.h>

#include<stdlib.h>

#include<iostream>

\_\_global\_\_ void MulRowPerThread(int \*a,int \*b,int \*c,int m,int n,int o)

{

  int id=threadIdx.x;

  if(id<m)

  {

  for(int i=0;i<o;i++)

  {

    c[i+id\*o]=0;

    for(int k=0;k<n;k++)

    c[i+id\*o]+=a[id\*n+k]\*b[i+k\*o];

  }

  }

}

\_\_global\_\_ void MulColPerThread(int \*a,int \*b,int \*c,int m,int n,int o)

{

  int id=threadIdx.x;

  if(id<o)

  {

    for(int i=0;i<m;i++)

    {

      c[id+i\*o]=0;

      for(int k=0;k<n;k++)

      c[id+i\*o]+=a[i\*n+k]\*b[id+k\*o];

    }

  }

}

\_\_global\_\_ void MulElePerThread(int \*a,int \*b,int \*c,int m,int n,int o)

{

  int id=threadIdx.x;

  if(id<m\*o)

  {

    c[id]=0;

    for(int k=0;k<n;k++)

    c[id]+=a[(id/o)\*n+k]\*b[(id%o)+k\*o];

  }

}

int main()

{

  int m=3,n=3,o=3;

  //declare matrices

  int a[3][3]={{1,1,1},{2,2,2},{3,3,3}};

  int b[3][3]={{1,1,1},{2,2,2},{3,3,3}};

  int c[m][o];

  int \*d\_a,\*d\_b,\*d\_c;

  //allocate memory

  int size=sizeof(int);

  cudaMalloc((void\*\*)&d\_a,m\*n\*size);

  cudaMalloc((void\*\*)&d\_b,n\*o\*size);

  cudaMalloc((void\*\*)&d\_c,m\*o\*size);

  //copy from host to device

  cudaMemcpy(d\_a,a,m\*n\*size,cudaMemcpyHostToDevice);

  cudaMemcpy(d\_b,b,n\*o\*size,cudaMemcpyHostToDevice);

   // Launch add() kernels on GPU

   //Number of blocks=1 threads=m

 printf("MULTIPLICATION\n");

   printf("row result per thread\n");

   MulRowPerThread<<<1,m>>>(d\_a, d\_b, d\_c,m,n,o);

   cudaMemcpy(c, d\_c,m\*o\*size,cudaMemcpyDeviceToHost);

   for(int i=0;i<m;i++)

   {

    for(int j=0;j<o;j++)

    {

      printf("%d ",c[i][j]);

    }

    printf("\n");

   }

   //Number of blocks=1 threads=n

   printf("column result per thread\n");

   MulColPerThread<<<1,o>>>(d\_a, d\_b, d\_c,m,n,o);

   cudaMemcpy(c, d\_c,m\*o\*size,cudaMemcpyDeviceToHost);

   for(int i=0;i<m;i++)

   {

    for(int j=0;j<o;j++)

    {

      printf("%d ",c[i][j]);

    }

    printf("\n");

   }

   //Number of blocks=1 threads=m\*n

   printf("element result per thread\n");

   MulElePerThread<<<1,m\*n>>>(d\_a, d\_b, d\_c,m,n,o);

   cudaMemcpy(c, d\_c,m\*o\*size,cudaMemcpyDeviceToHost);

   for(int i=0;i<m;i++)

   {

    for(int j=0;j<o;j++)

    {

      printf("%d ",c[i][j]);

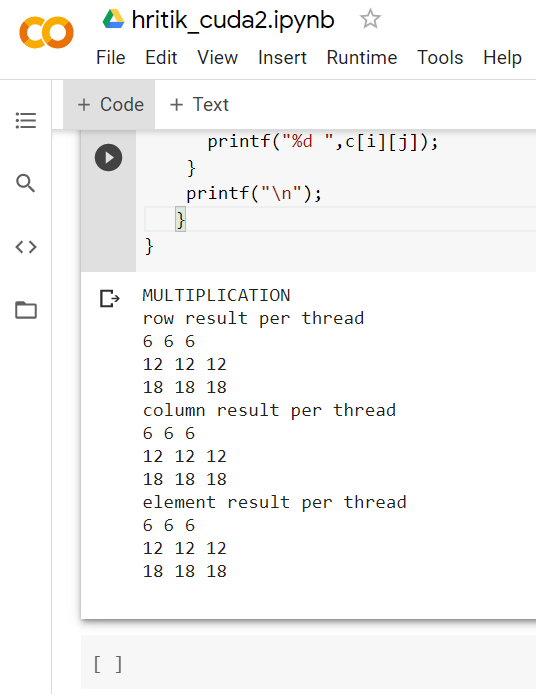
    }

    printf("\n");

   }

}

**OUTPUT:**

****