Arquitectura y Computación de Altas Prestaciones

grado en ingeniería informática

Ejercicios de teoría

Reducción

Autor
Vladislav Nikolov Vasilev

Rama
Ingeniería de Computadores

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

Índice

1. Ejercicio transparencia 23

En este ejercicio se pide determinar qué elementos tendría la **hebra 20 del bloque 1** teniendo en cuenta que se tiene un vector de 256 elementos y 2 bloques con 64 hebras cada uno. A continuación se proporciona el código comentado:

```
unsigned int dimHebras = 128 / 2; // dimHebras = 64

// sumaParcial[2*64] -> sumaParcial[128]
__ shared__ float sumaParcial[2*dimHebras];

unsigned int h = id_hebra; // h = 20

// start = 2 * 1 * 64 = 128
unsigned int start=2*id_bloque*dimHebras;

// sumaParcial[20] = Input[128 + 20] -> sumaParcial[20] = Input[148]
sumaParcial[h]=Input[start+h];

// sumaParcial[64+20]=Input[128+64+20] -> sumaParcial[84]=Input[212]
sumaParcial[N+h]=Input[start+dimHebras+h];
```

2. Ejercicio transparencia 22: versión modificada

Se ha proporcionado una versión modificada del ejemplo que se puede ver en la transparencia 22. Aquí se tiene un **vector de entrada de 32 elementos**, los cuáles se reparten entre **2 bloques con 8 hebras cada uno**. Se ha calculado qué elementos del vector de entrada tendrían las dos primeras hebras de cada bloque.

2.1. Hebra 0 del bloque 0

```
unsigned int dimHebras = 16 / 2; // dimHebras = 8

// sumaParcial[2*8] -> sumaParcial[16]
__ shared__ float sumaParcial[2*dimHebras];

unsigned int h = id_hebra; // h = 0

// start = 2 * 0 * 8 = 0
unsigned int start=2*id_bloque*dimHebras;

// sumaParcial[0] = Input[0 + 0] -> sumaParcial[0] = Input[0]
sumaParcial[h]=Input[start+h];

// sumaParcial[8+0]=Input[0+8+0] -> sumaParcial[8] = Input[8]
sumaParcial[N+h]=Input[start+dimHebras+h];
```

2.2. Hebra 1 del bloque 0

```
unsigned int dimHebras = 16 / 2; // dimHebras = 8

// sumaParcial[2*8] -> sumaParcial[16]
__ shared__ float sumaParcial[2*dimHebras];

unsigned int h = id_hebra; // h = 1

// start = 2 * 0 * 8 = 0
unsigned int start=2*id_bloque*dimHebras;

// sumaParcial[1] = Input[0 + 1] -> sumaParcial[1] = Input[1]
sumaParcial[h]=Input[start+h];

// sumaParcial[8+1]=Input[0+8+1] -> sumaParcial[9] = Input[9]
sumaParcial[N+h]=Input[start+dimHebras+h];
```

2.3. Hebra 0 del bloque 1

```
unsigned int dimHebras = 16 / 2; // dimHebras = 8

// sumaParcial[2*8] -> sumaParcial[16]
__ shared__ float sumaParcial[2*dimHebras];

unsigned int h = id_hebra; // h = 0

// start = 2 * 1 * 8 = 16
unsigned int start=2*id_bloque*dimHebras;

// sumaParcial[0] = Input[16 + 0] -> sumaParcial[0] = Input[16]
sumaParcial[h]=Input[start+h];

// sumaParcial[8+0]=Input[16+8+0] -> sumaParcial[8] = Input[24]
sumaParcial[N+h]=Input[start+dimHebras+h];
```

2.4. Hebra 1 del bloque 0

```
unsigned int dimHebras = 16 / 2; // dimHebras = 8

// sumaParcial[2*8] -> sumaParcial[16]
__ shared__ float sumaParcial[2*dimHebras];

unsigned int h = id_hebra; // h = 1

// start = 2 * 1 * 8 = 16
unsigned int start=2*id_bloque*dimHebras;

// sumaParcial[0] = Input[16 + 1] -> sumaParcial[1] = Input[16]
sumaParcial[h]=Input[start+h];

// sumaParcial[8+1]=Input[16+8+1] -> sumaParcial[9] = Input[25]
```

```
15 sumaParcial[N+h]=Input[start+dimHebras+h];
```