



ugr

Universidad  
de Granada

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

<b>1. Ejercicio transparencia 23</b>	<b>2</b>
<b>2. Ejercicio transparencia 22: versión modificada</b>	<b>2</b>
2.1. Hebra 0 del bloque 0 . . . . .	2
2.2. Hebra 1 del bloque 0 . . . . .	3
2.3. Hebra 0 del bloque 1 . . . . .	3
2.4. Hebra 1 del bloque 1 . . . . .	3

## 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:

```
1 unsigned int dimHebras = 128 / 2; // dimHebras = 64
2
3 // sumaParcial[2*64] -> sumaParcial[128]
4 __shared__ float sumaParcial[2*dimHebras];
5
6 unsigned int h = id_hebra; // h = 20
7
8 // start = 2 * 1 * 64 = 128
9 unsigned int start=2*id_bloque*dimHebras;
10
11 // sumaParcial[20] = Input[128 + 20] -> sumaParcial[20] = Input[148]
12 sumaParcial[h]=Input[start+h];
13
14 // sumaParcial[64+20]=Input[128+64+20] -> sumaParcial[84]=Input[212]
15 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

```
1 unsigned int dimHebras = 16 / 2; // dimHebras = 8
2
3 // sumaParcial[2*8] -> sumaParcial[16]
4 __shared__ float sumaParcial[2*dimHebras];
5
6 unsigned int h = id_hebra; // h = 0
7
8 // start = 2 * 0 * 8 = 0
9 unsigned int start=2*id_bloque*dimHebras;
10
11 // sumaParcial[0] = Input[0 + 0] -> sumaParcial[0] = Input[0]
12 sumaParcial[h]=Input[start+h];
13
14 // sumaParcial[8+0]=Input[0+8+0] -> sumaParcial[8] = Input[8]
15 sumaParcial[N+h]=Input[start+dimHebras+h];
```

## 2.2. Hebra 1 del bloque 0

```
1 unsigned int dimHebras = 16 / 2; // dimHebras = 8
2
3 // sumaParcial[2*8] -> sumaParcial[16]
4 __shared__ float sumaParcial[2*dimHebras];
5
6 unsigned int h = id_hebra; // h = 1
7
8 // start = 2 * 0 * 8 = 0
9 unsigned int start=2*id_bloque*dimHebras;
10
11 // sumaParcial[1] = Input[0 + 1] -> sumaParcial[1] = Input[1]
12 sumaParcial[h]=Input[start+h];
13
14 // sumaParcial[8+1]=Input[0+8+1] -> sumaParcial[9] = Input[9]
15 sumaParcial[N+h]=Input[start+dimHebras+h];
```

## 2.3. Hebra 0 del bloque 1

```
1 unsigned int dimHebras = 16 / 2; // dimHebras = 8
2
3 // sumaParcial[2*8] -> sumaParcial[16]
4 __shared__ float sumaParcial[2*dimHebras];
5
6 unsigned int h = id_hebra; // h = 0
7
8 // start = 2 * 1 * 8 = 16
9 unsigned int start=2*id_bloque*dimHebras;
10
11 // sumaParcial[0] = Input[16 + 0] -> sumaParcial[0] = Input[16]
12 sumaParcial[h]=Input[start+h];
13
14 // sumaParcial[8+0]=Input[16+8+0] -> sumaParcial[8] = Input[24]
15 sumaParcial[N+h]=Input[start+dimHebras+h];
```

## 2.4. Hebra 1 del bloque 1

```
1 unsigned int dimHebras = 16 / 2; // dimHebras = 8
2
3 // sumaParcial[2*8] -> sumaParcial[16]
4 __shared__ float sumaParcial[2*dimHebras];
5
6 unsigned int h = id_hebra; // h = 1
7
8 // start = 2 * 1 * 8 = 16
9 unsigned int start=2*id_bloque*dimHebras;
10
11 // sumaParcial[1] = Input[16 + 1] -> sumaParcial[1] = Input[17]
12 sumaParcial[h]=Input[start+h];
13
14 // sumaParcial[8+1]=Input[16+8+1] -> sumaParcial[9] = Input[25]
```

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