INTRODUCCIÓN A CUDA

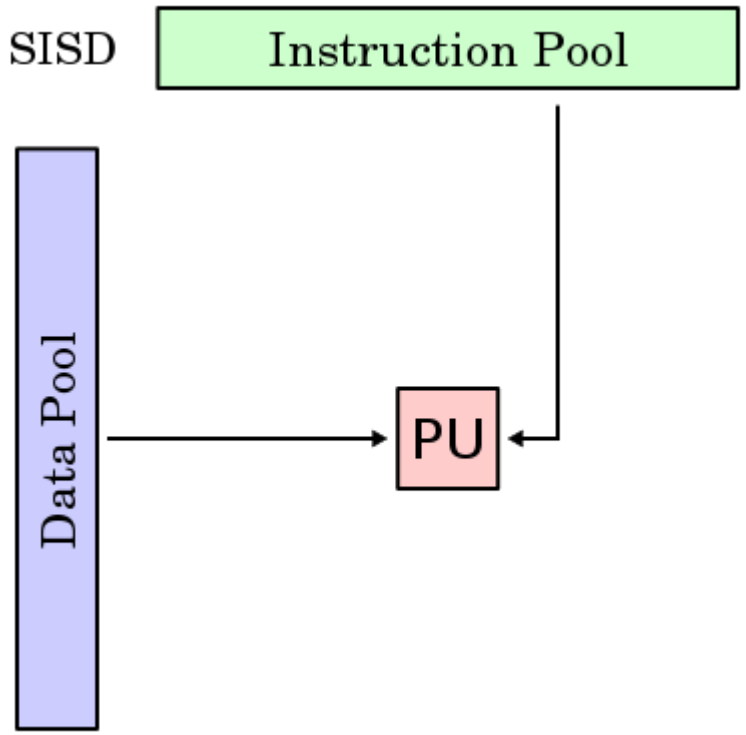
JGPU 2020

Albert García-García < agarcia@dtic.ua.es >

CONTENIDO

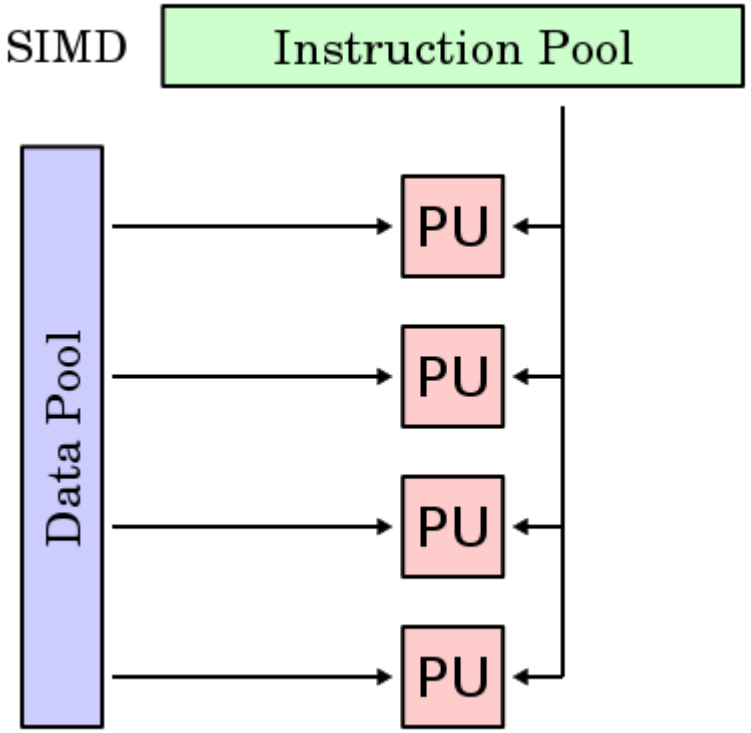
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (SISD)



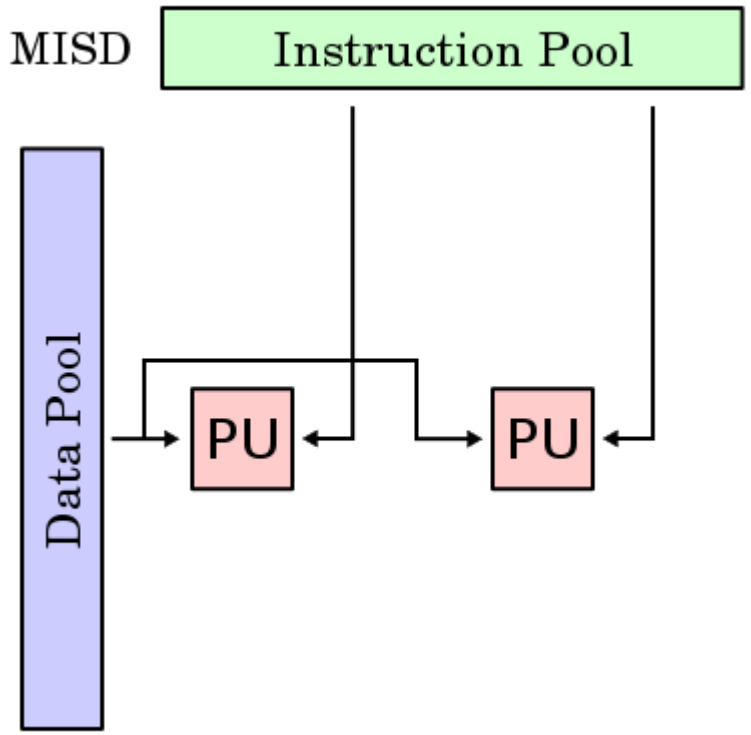
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (SIMD)



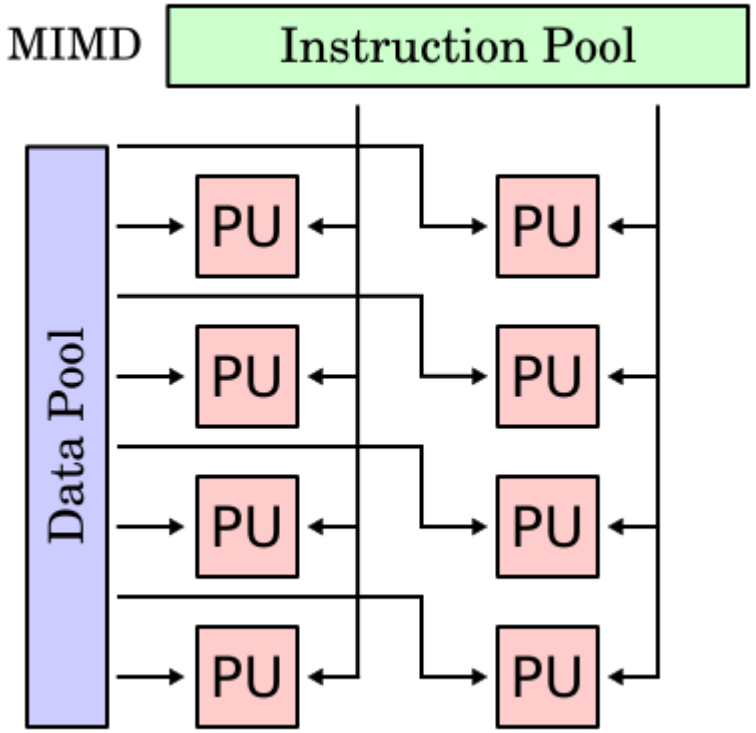
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (MISD)



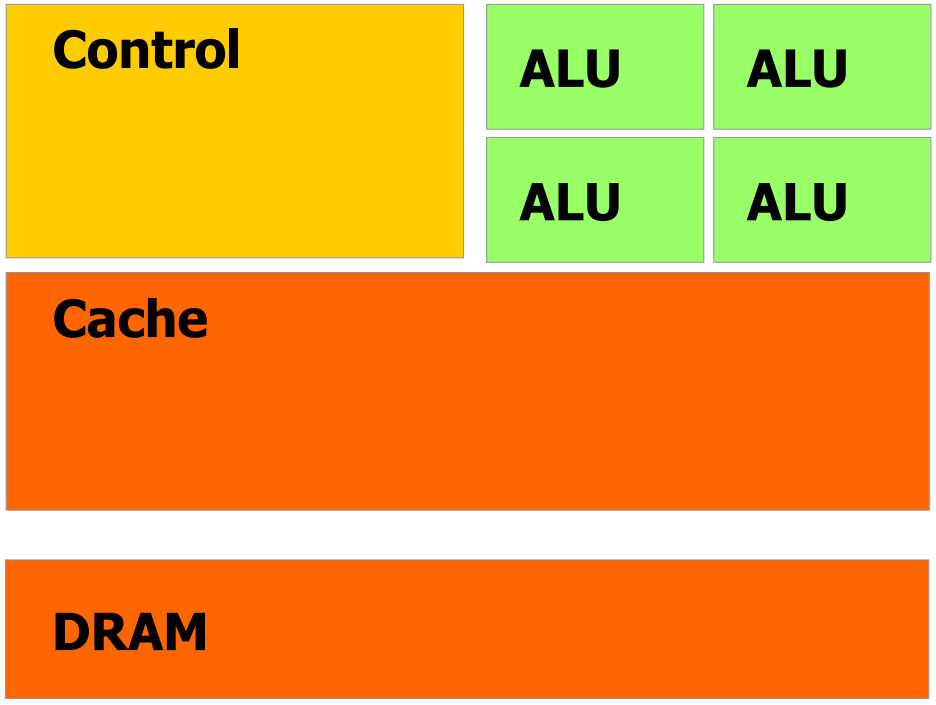
ARQUITECTURA HARDWARE

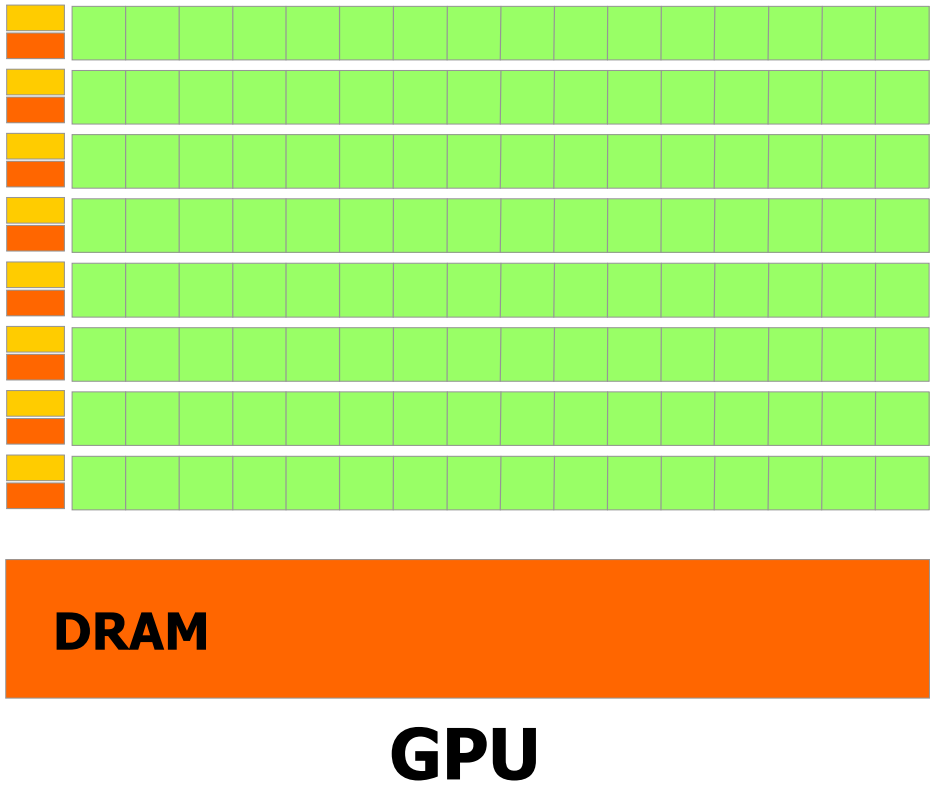
TAXONOMÍA DE FLYNN (MIMD)



ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU





ARQUITECTURA HARDWARE

COMPONENTES BÁSICOS DE UNA GPU CUDA

INTERFAZ CON EL HOST (CPU) QUE CONECTA LA GPU AL BUS PCI EXPRESS

0-2 COPY ENGINES PARA TRANSFERENCIAS DE MEMORIA ASÍNCRONAS

INTERFAZ DE MEMORIA DRAM QUE CONECTA A LA GPU CON SU MEMORIA INTERNA

CIERTO NÚMERO DE GRAPHICS PROCESSING CLUSTERS (GPCS) CON STREAMING MULTIPROCESSORS (SMS) O PROCESADORES DE GENERACIÓN ACTUAL

ARQUITECTURA HARDWARE

STREAMING MULTIPROCESSORS

INSTRUCTION CACHE/DECODER

SCHEDULER

CUDA CORES

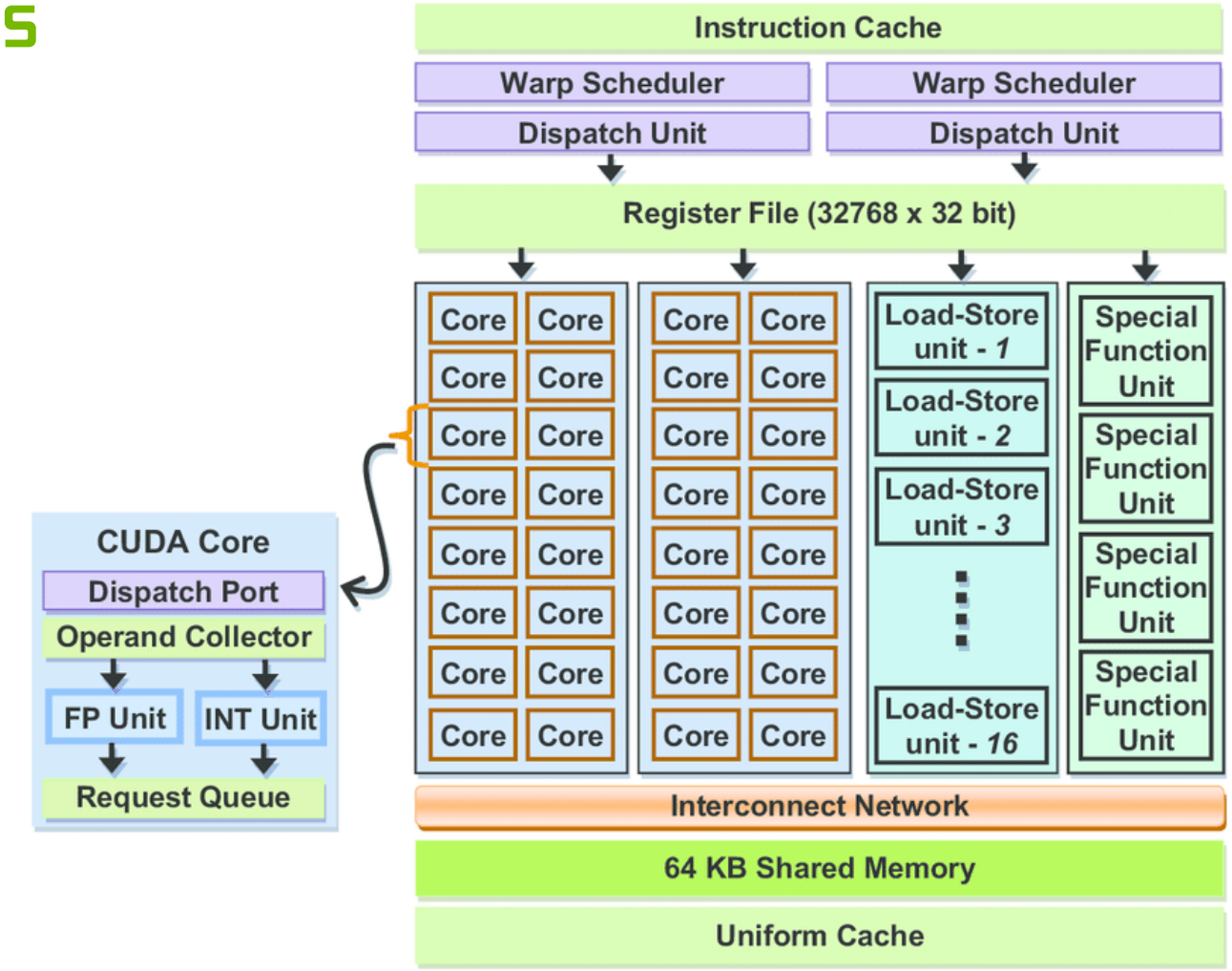
UNIDADES LOAD-STORE

UNIDADES SFU

MEMORIA COMPARTIDA

CACHÉS DE TEXTURAS

ARCHIVO DE REGISTROS



ARQUITECTURA HARDWARE

TURING

VOLTA

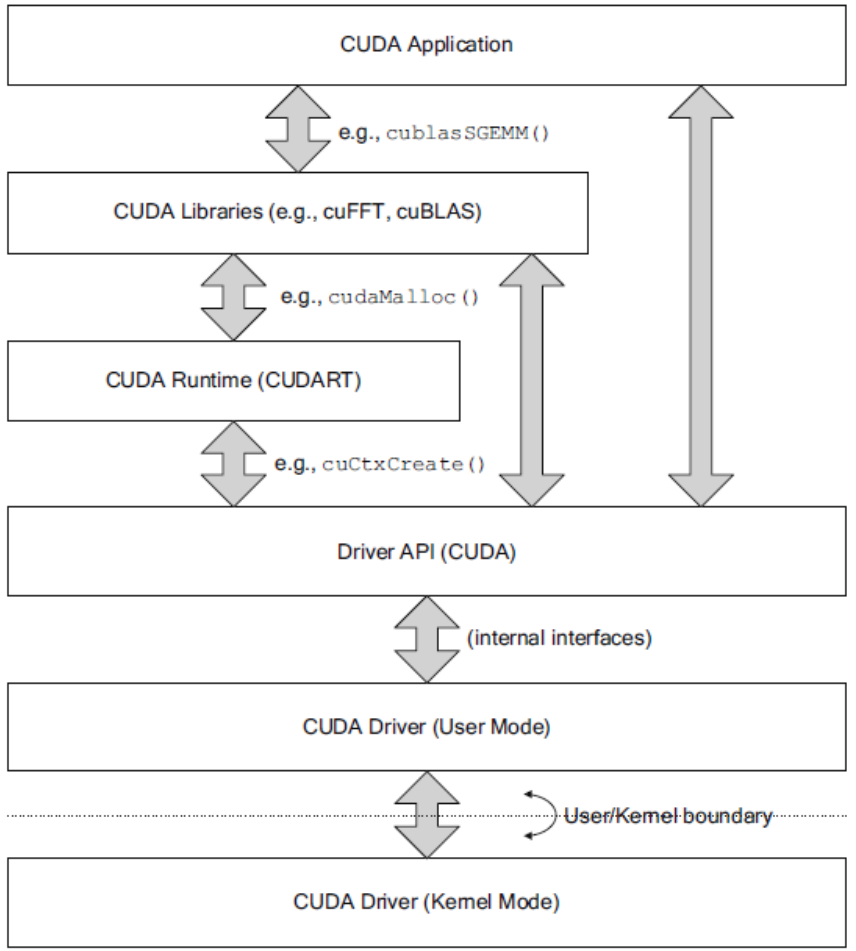
EVOLUCIÓN DE MICROARQUITECTURAS

ARQUITECTURA HARDWARE

GPU TECHNOLOGY CONFERENCE 2018

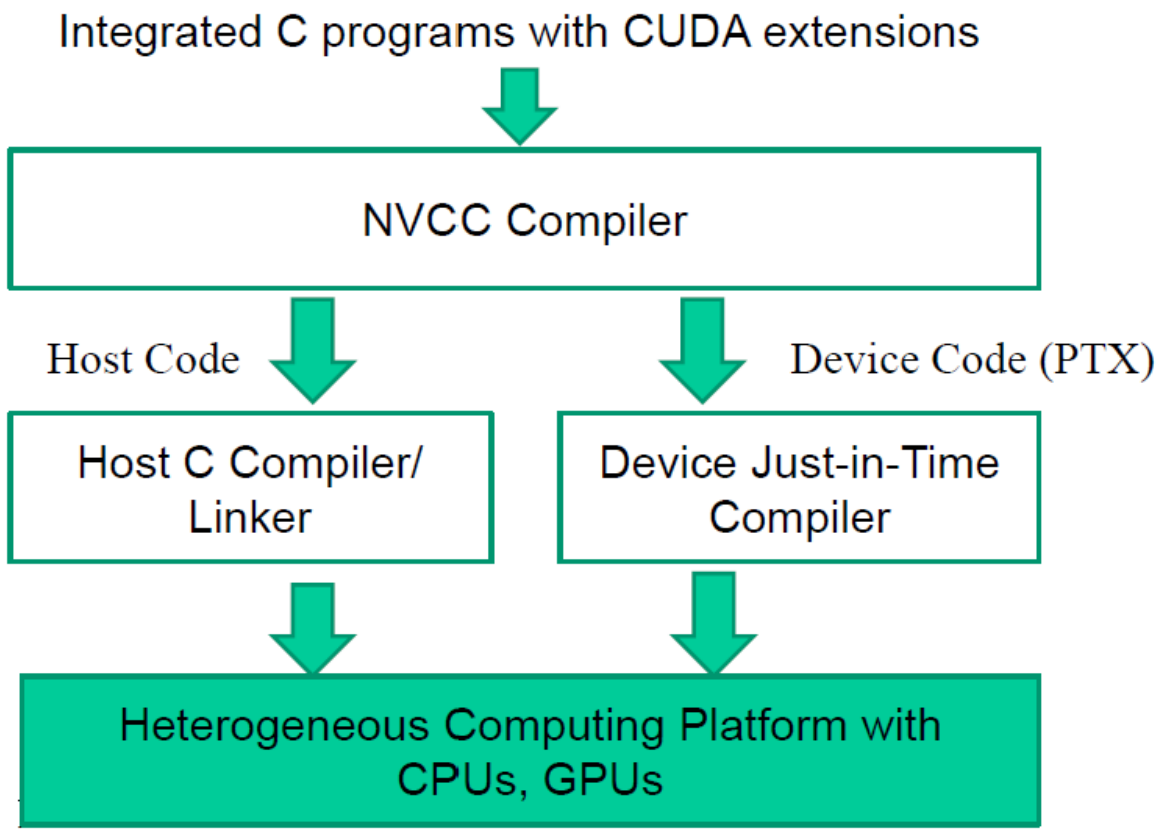
ARQUITECTURA SOFTWARE

STACK CUDA



ARQUITECTURA SOFTWARE

COMPILACIÓN



ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

KERNEL: es una función que al ejecutarse lo hará en una gran cantidad de hilos y en la GPU.

BLOCK: es una agrupación de hilos en 1D, 2D ó 3D. Cada bloque se ejecuta sobre un único SM, pero un SM puede tener asignados varios bloques para ejecución.

GRID: es una forma de estructurar los bloques, bien en 1D, 2D ó 3D

ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

INVOCACIÓN DE KERNEL:

```
kernel_routine<<<grid_dim, block_dim>>> (args...);
```

TAMAÑOS DE BLOQUE Y MALLA DEFINIDOS CON DIM3:

```
dim3 block_dim (32, 32, 1); // 1024 hilos en bloque de 32 x 32 x 1 (2D)
dim3 grid_dim (4, 4, 1); // 16 bloques en malla de 4 x 4 x 1 (2D)
```

ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

CADA HILO EJECUTA UNA COPIA DEL KERNEL, Y DISPONE DE LA SIGUIENTE INFORMACIÓN:

VARIABLES PASADAS POR ARGUMENTO (COMO PUNTEROS A MEMORIA GPU)

CONSTANTES GLOBALES EN MEMORIA GPU

VARIABLES ESPECIALES (DIM3) PARA IDENTIFICAR AL HILO:

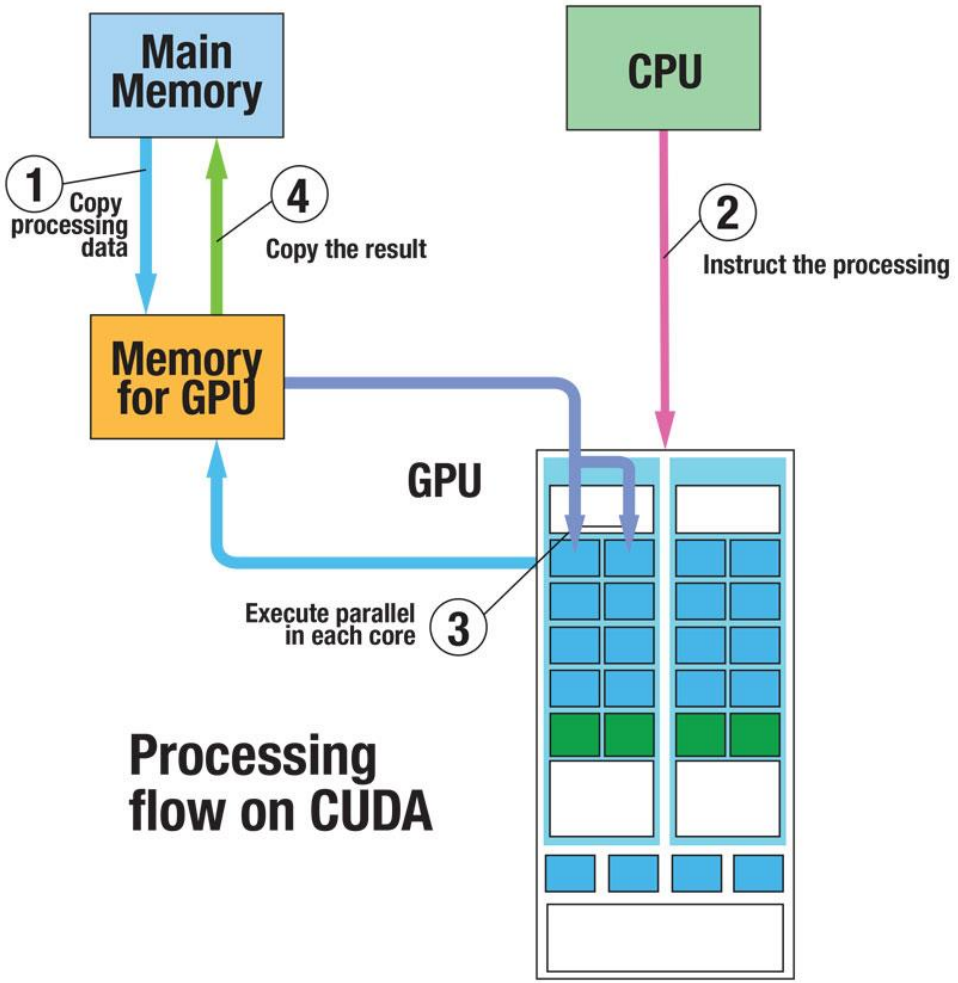
gridDim (tamaño de la malla)
blockDim (tamaño de los bloques)
blockIdx (identificador de bloque) LOCAL PARA CADA BLOQUE
threadIdx (identificador de hilo) LOCAL PARA CADA BLOQUE

ARQUITECTURA SOFTWARE

EJECUCIÓN DE KERNEL Y DIRECCIONAMIENTO

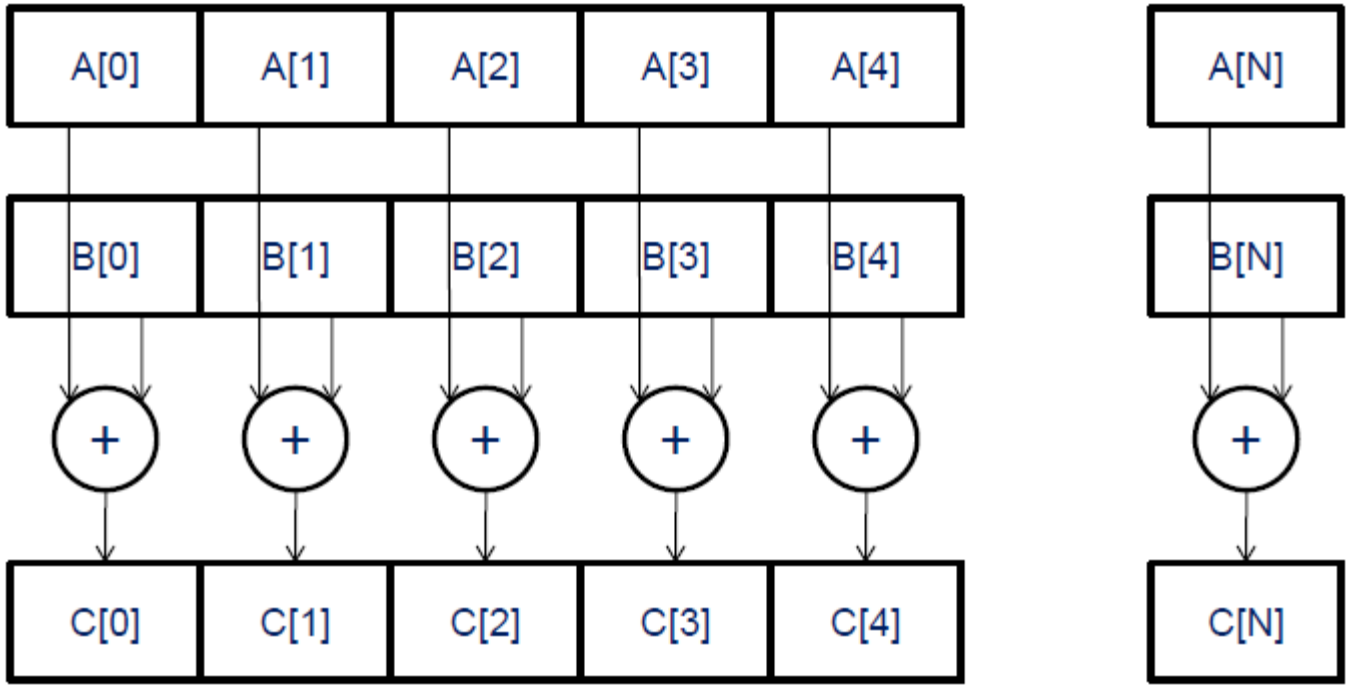
FLUJO DE EJECUCIÓN

SUBTÍTULO QUE NADIE LEE



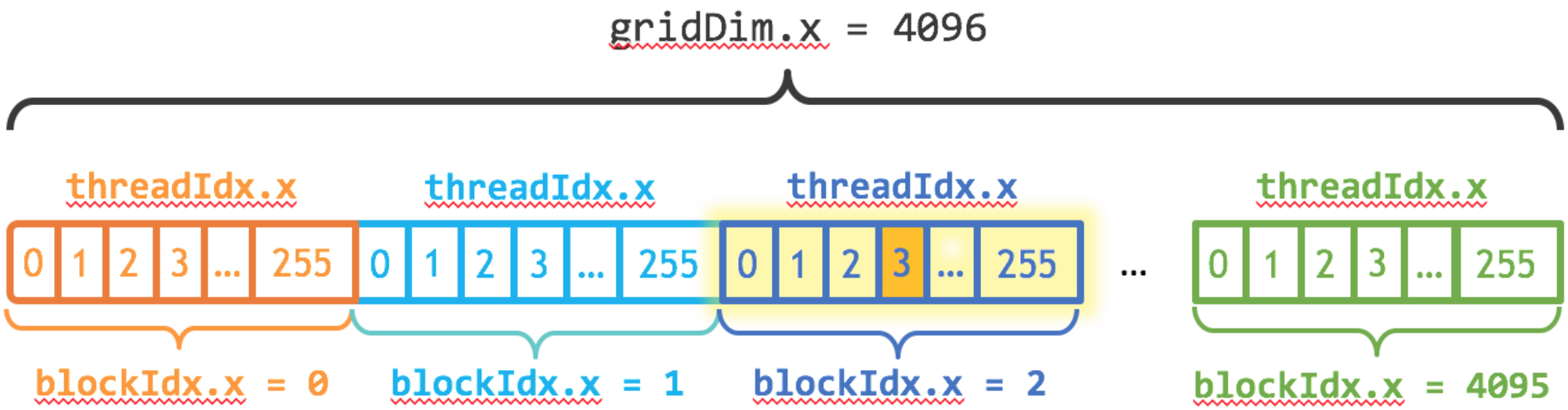
PRIMER PROGRAMA CUDA

SUMA DE VECTORES



PRIMER PROGRAMA CUDA

EJEMPLO




```
index = blockIdx.x  * blockDim.x + threadIdx.x

index =    (2)      *   (256)    +     (3)      = 515
```

PRIMER PROGRAMA CUDA

SUMA DE VECTORES

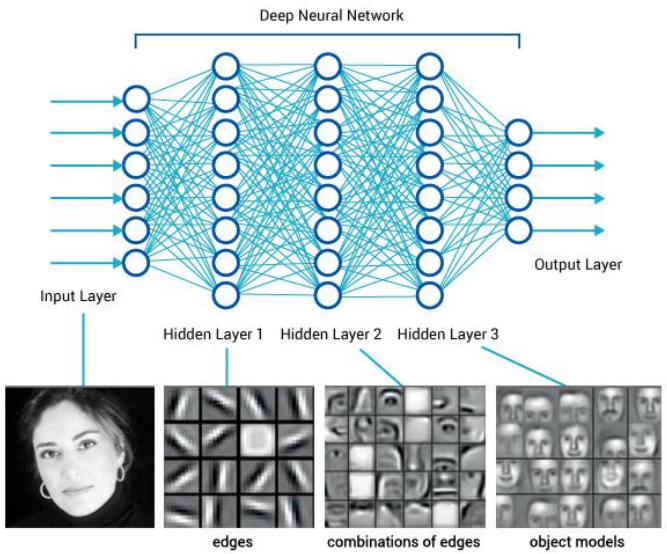
```
// Compute vector sum C = A+B
void vecAdd(float* A, float* B, float* C, int n)
{
  for (i = 0, i < n, i++)
    C[i] = A[i] + B[i];
}
```

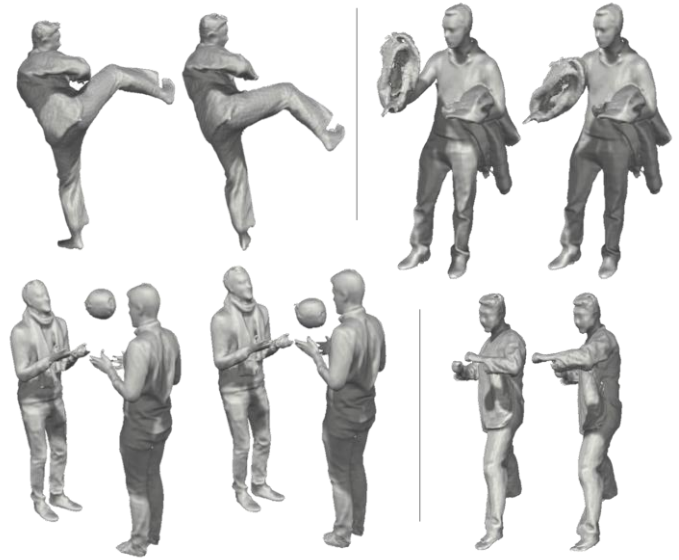
Código paralelizado GPU

```
__global__
void vecAddkernel(float* d_A, float* d_B, float* d_C, int n)
{
    int i = threadIdx.x + blockDim.x * blockIdx.x;
    if(i<n) d_C[i] = d_A[i] + d_B[i];
}
```

LÍNEAS DE INVESTIGACIÓN

¡COLABORA CON NOSOTROS!





LÍNEAS DE INVESTIGACIÓN

¡COLABORA CON NOSOTROS!

INTRODUCCIÓN A CUDA

JGPU 2020

Albert García-García < agarcia@dtic.ua.es >