2º curso / 2º cuatr.

Grado en
Ing. Informática

Arquitectura de Computadores
Tema 1

Arquitecturas Paralelas: Clasificación y Prestaciones

Material elaborado por los profesores responsables de la asignatura:
Mancia Anguita – Julio Ortega

ugr | Universidad de Granada

ETSIIT
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

ATC
Departamento de Arquitectura y Tecnología de Computadores
UNIVERSIDAD DE GRANADA

Lecciones

- Lección 1. Clasificación del paralelismo implícito en una aplicación
- Lección 2. Clasificación de arquitecturas paralelas
- Lección 3. Evaluación de prestaciones

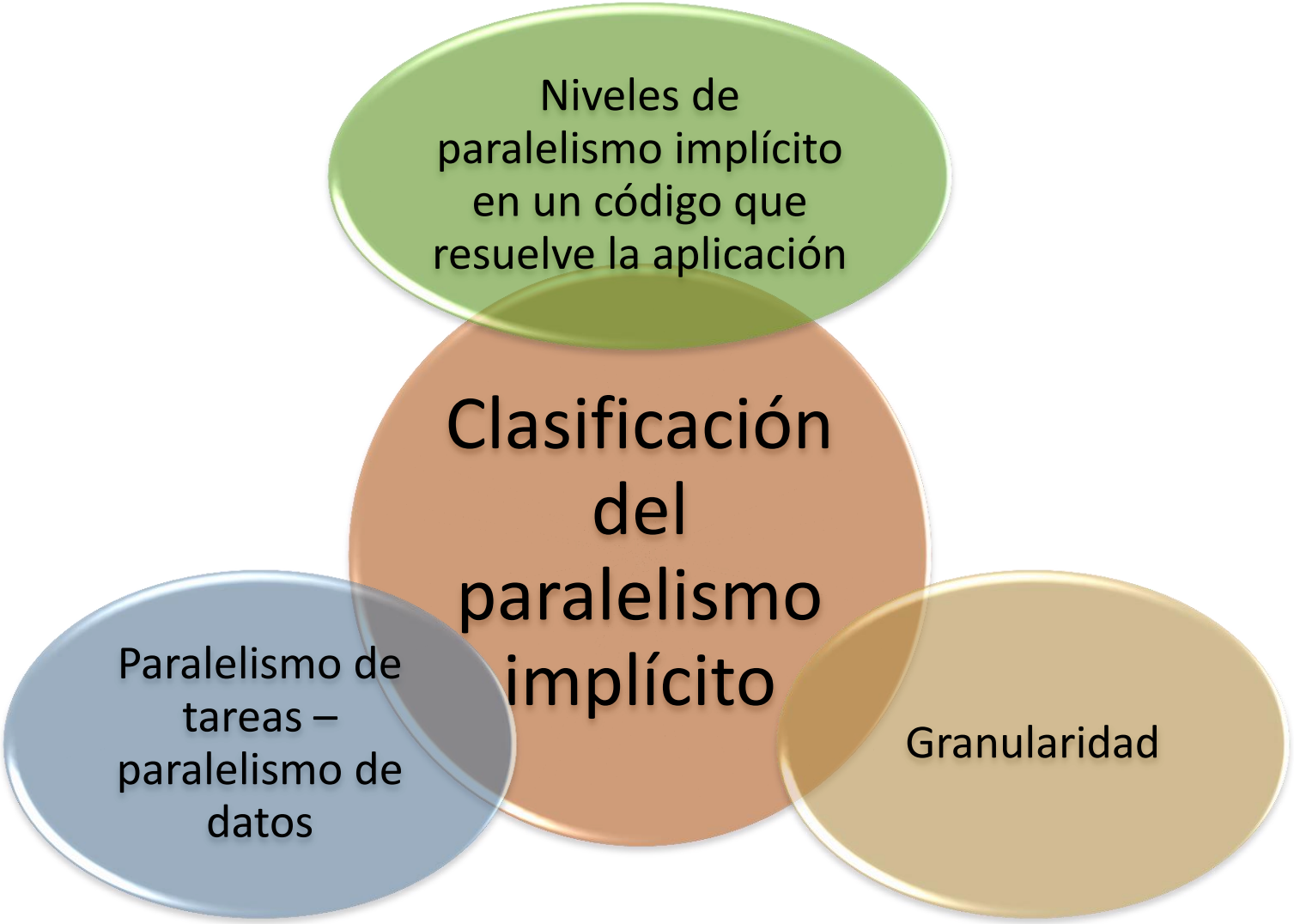
Objetivos Lección 1

- Conocer las clasificaciones usuales del paralelismo implícito en una aplicación. Distinguir entre paralelismo de tareas y paralelismo de datos.
- Distinguir entre dependencias RAW, WAW, WAR.
- Distinguir entre *thread* y proceso.
- Relacionar el paralelismo implícito en una aplicación con el nivel en el que se hace explícito para que se pueda utilizar (instrucción, thread, proceso) y con las arquitecturas paralelas que lo aprovechan.

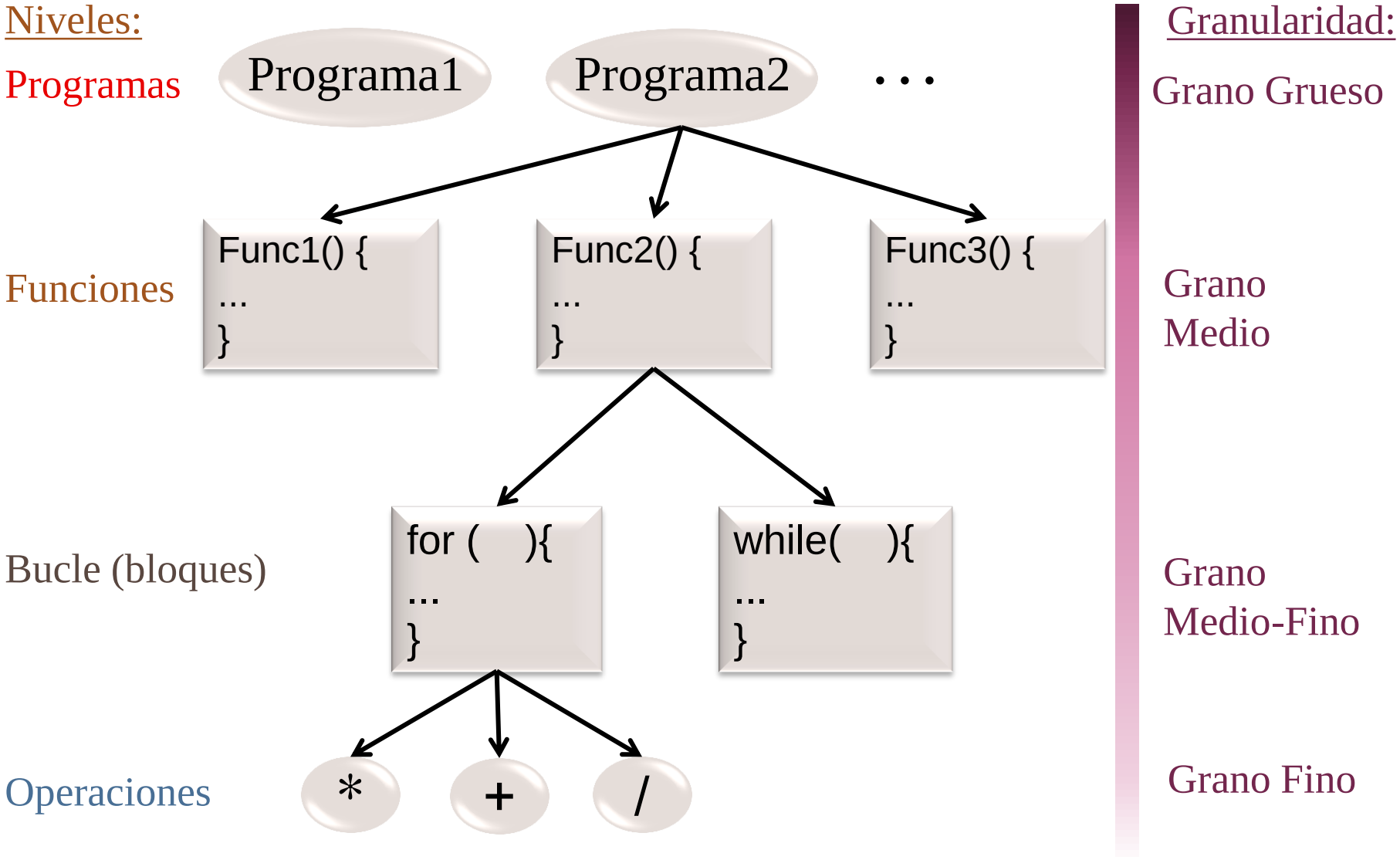
Bibliografía

- Fundamental
  - Sección 7.1. J. Ortega, M. Anguita, A. Prieto. *Arquitectura de Computadores*, Thomson, 2005. ESIIT/C.1 ORT arq
- Complementaria
  - Secciones 3.7.1, 3.7.2. T. Rauber, G. Ründer. *Parallel Programming: for Multicore and Cluster Systems*. Springer 2010. Disponible en línea (biblioteca UGR): http://dx.doi.org/10.1007/978-3-642-04818-0

Criterios de clasificaciones del paralelismo implícito en una aplicación

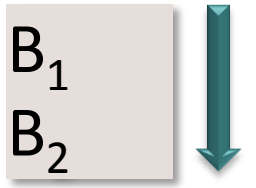


Niveles de paralelismo implícito en una aplicación



Dependencias de datos

- Condiciones que se deben cumplir para que el bloque de código B_2 presente dependencia de datos con respecto a B_1:
  - Deben hacer referencia a una misma posición de memoria M (variable).
  - B_1 aparece en la secuencia de código antes que B_2



- Tipos de dependencias de datos (de B_2 respecto a B_1):
  - RAW (*Read After Write*) o dependencia verdadera
  - WAW (*Write After Write*) o dependencia de salida
  - WAR (*Write After Read*) o antidependencia

```
...
a=b+c
d=a+c
...
```

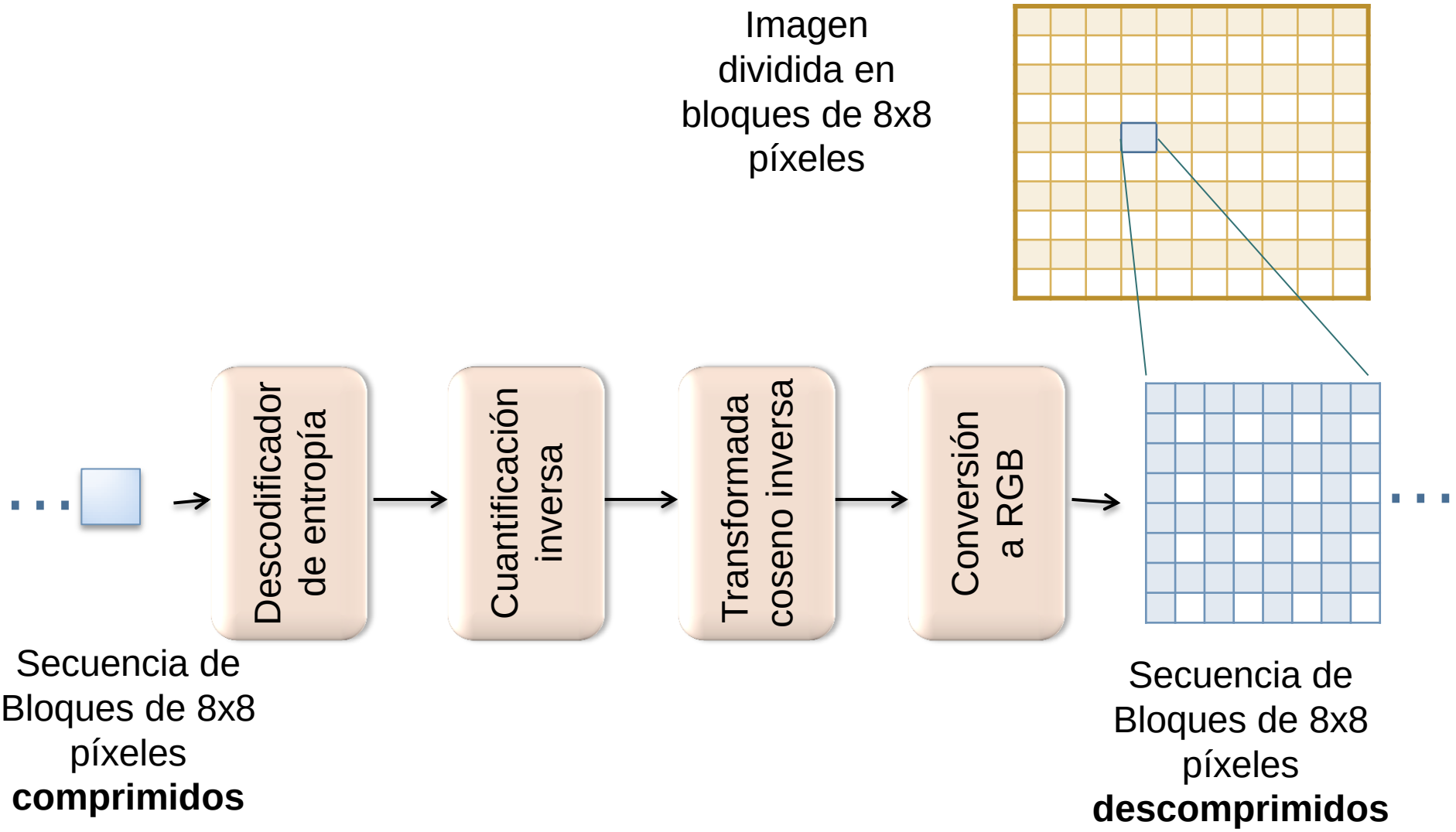
```
...
a=b+c
...
a=d+e
...
```

```
...
b=a+1
...
a=d+e
...
```

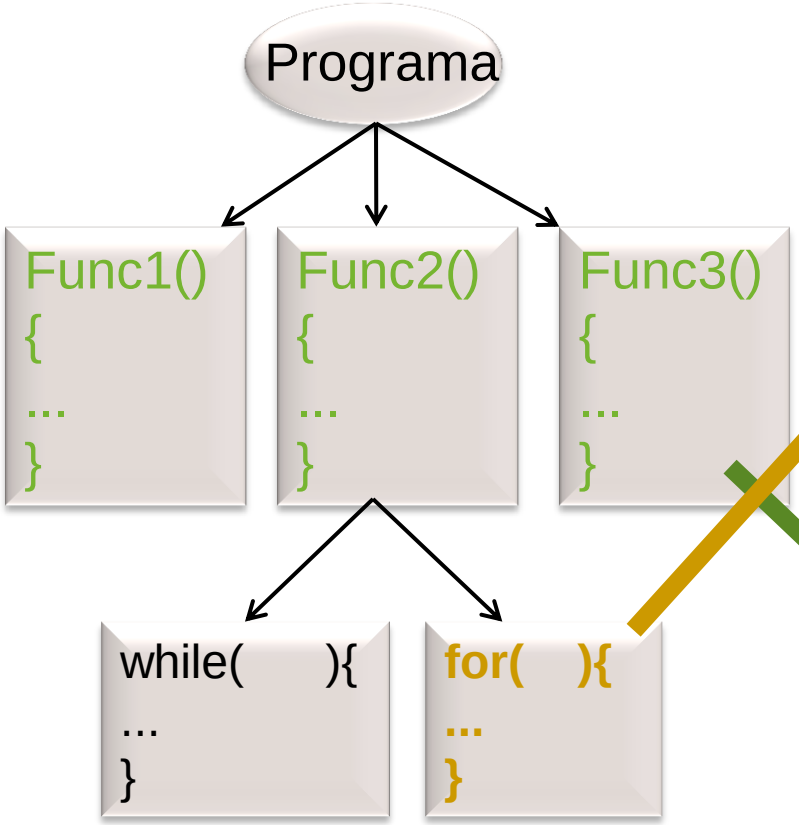
Paralelismo implícito en una aplicación

- Paralelismo de tareas (*task parallelism o TLP -Task Level Par.*)
  - Se encuentra extrayendo la estructura lógica de funciones de una aplicación.
  - Está relacionado con el paralelismo a ***nivel de función***
- Paralelismo de datos (*data parallelism o DLP-Data Level Par.*)
  - Se encuentra implícito en las operaciones con estructuras de datos (vectores y matrices)
    - Por ejemplo, la operación vectorial V1=V2+V3 engloba múltiples sumas de escalares.
  - Se puede extraer de la representación matemática de la aplicación.
  - Está relacionado principalmente con el paralelismo a ***nivel de bucle***

Estructura de funciones lógica de una aplicación. Ej.: decodificador JPEG

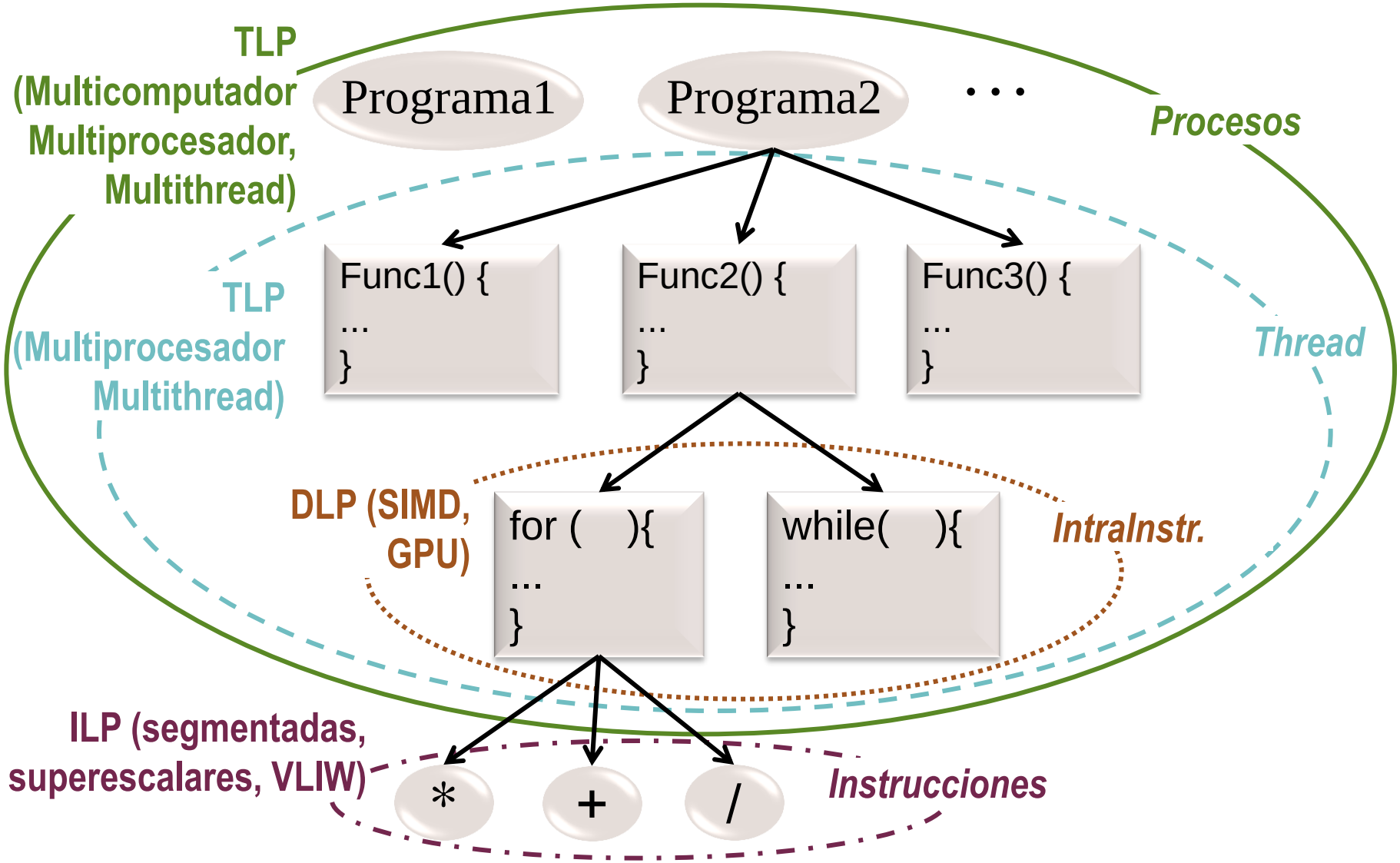


Paralelismo de datos y paralelismo de tareas en OpenMP (Prácticas 1, 2 y 3)




```
Func1() {…}
Func2() { …
  #pragma omp parallel for
    for (i=0;i<N;i=++){
          código para i
    }
…}
Func3() {…}
Main () {
…
 #pragma omp parallel sections
  { #pragma omp section
          Func1();
    #pragma omp section
          Func2();
    #pragma omp section
          Func3();
  }
…}
```

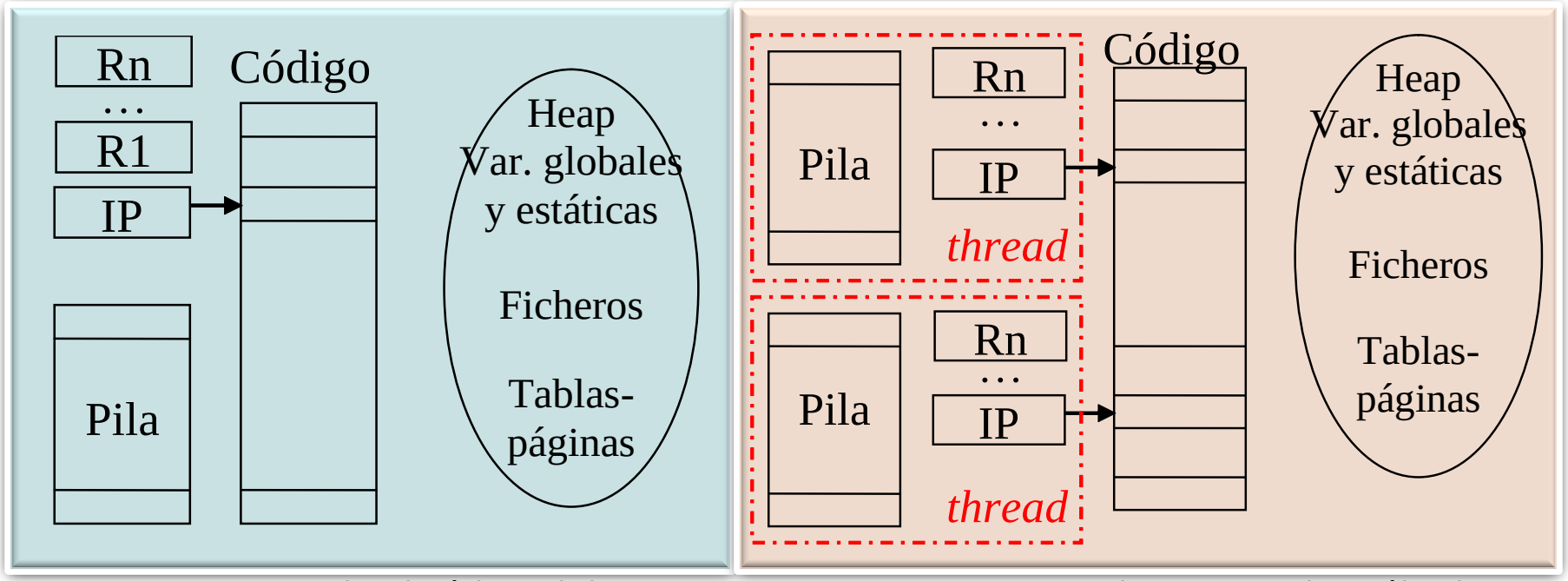
Paralelismo implícito (nivel de detección), explícito y arquitecturas paralelas



Nivel de paralelismo explícito. Unidades en ejecución en un computador

- Instrucciones
  - La unidad de control de un core o procesador gestiona la ejecución de instrucciones por la unidad de procesamiento
- *Thread* o *light process* (concepto del SO)
  - Es la menor unidad de ejecución que gestiona el SO
  - Menor secuencia de instrucciones que se pueden ejecutar en paralelo o concurrentemente
- Proceso o *process* (concepto del SO)
  - Mayor unidad de ejecución que gestiona el SO
  - Un proceso consta de uno o varios thread

Nivel de paralelismo explícito. Threads versus procesos I

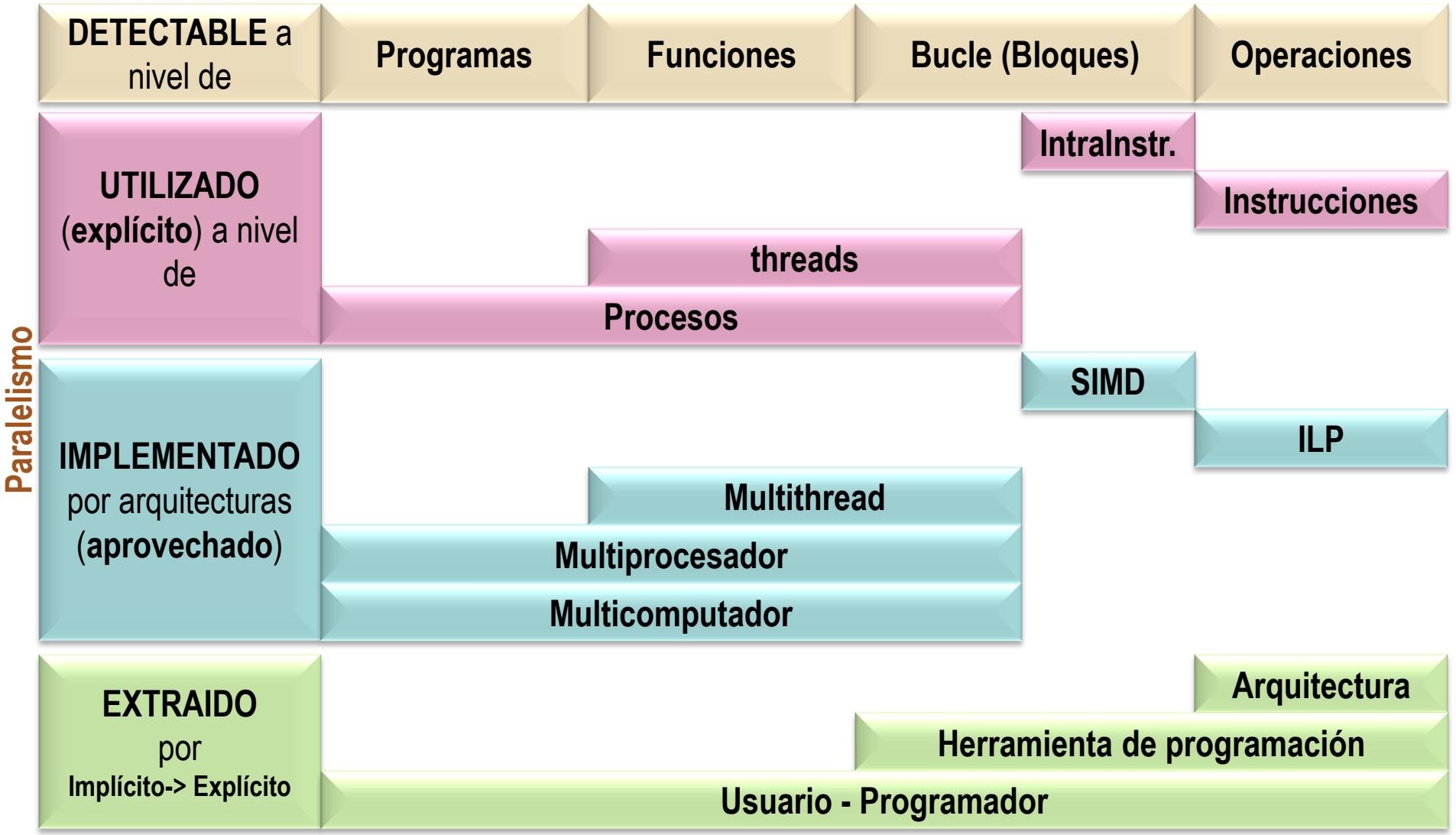


- Proceso: comprende el código del programa y todo lo que hace falta para su ejecución:
  - Datos en pila, segmentos (variables globales y estáticas) y en heap
  - Contenido de los registros
  - Tabla de páginas
  - Tabla de ficheros abiertos
- Para comunicar procesos hay que usar llamadas al SO

- Un proceso puede constar de múltiples flujos de control, llamados threads o procesos ligeros. Cada thread tiene:
  - Su propia pila
  - Contenido de los registros, en particular el contador de programa o *instruction pointer* y el puntero de pila o *stack pointer*
- Para comunicar threads de un proceso se usa la memoria que comparten

Nivel de paralelismo explícito. Threads versus procesos II

Detección, utilización, implementación y extracción del paralelismo



Para ampliar .....

- Páginas Web:
  - http://en.wikipedia.org/wiki/Instruction-level_parallelism
  - http://en.wikipedia.org/wiki/Task_parallelism
  - http://en.wikipedia.org/wiki/Data_parallelism