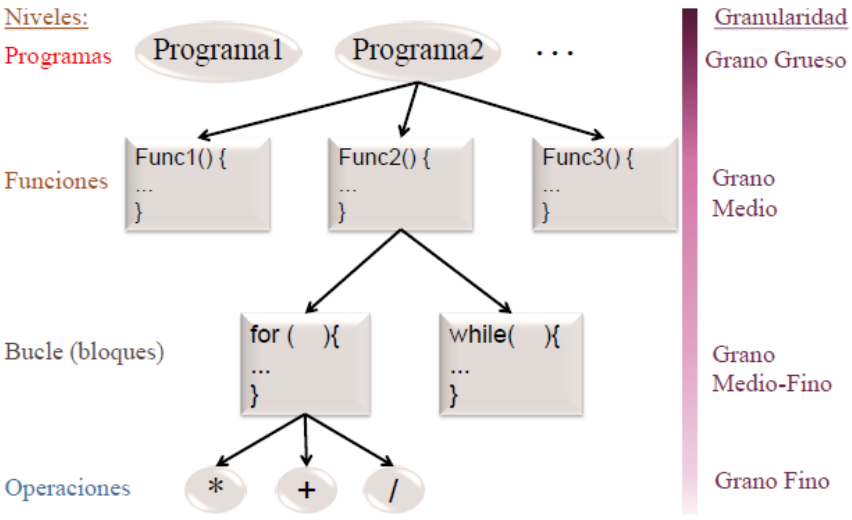
Arquitectura de Computadores. Tema 1

Arquitecturas paralelas: clasificación y prestaciones

Lección 1: clasificación del paralelismo implícito en una aplicación

Criterios de clasificaciones del paralelismo implícito en una aplicación

Niveles de paralelismo implícito en una aplicación



- Programas: Los diferentes programas que intervienen en una aplicación o en diferentes aplicaciones se pueden ejecutar en paralelo. Es poco probable que exista dependencia entre ellos.
- Funciones: en un nivel de abstracción más bajo, un programa puede considerarse constituido por funciones. Las funciones llamada en un programa se pueden ejecutar en paralelo, siempre que no haya entre ellas dependencias (riesgos) inevitables, como dependencias de datos verdaderas (lectura después de escritura (RAW)).
- Bucle (bloque): una función puede estar basada en la ejecución de uno o varios bucles (for, while,do-while). El código dentro de un bucle se ejecuta múltiples veces, en cada

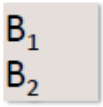
iteración se completa una tarea. Se pueden ejecutar en paralelo las iteraciones de un bucle, siempre que se eliminen los problemas derivados de dependencias verdaderas. Para detectar las dependencias habrá que analizar las entradas y salidas de las iteraciones del bucle.

- Operaciones: en este nivel se extrae el paralelismo disponible entre operaciones. Las operaciones independientes se pueden ejecutar en paralelo. Por otra parte, en los procesadores de propósito específico y en los de propósito general (por ej. Dentro de los repertorios multimedia) podemos encontrar instrucciones compuestas de varias operaciones que se aplican en secuencia al mismo flujo de datos de entrada. Por ejemplo la instrucción *mac*, introducida primero en los DSP(Digital Signal Processor), que realiza una multiplicación seguida de una suma, o instrucciones que facilitan el cálculo de la media, que realizan una suma seguida de una división. En este nivel se puede detectar la posibilidad de usar instrucciones compuestas, que van a evitar penalizaciones por dependencias verdaderas.

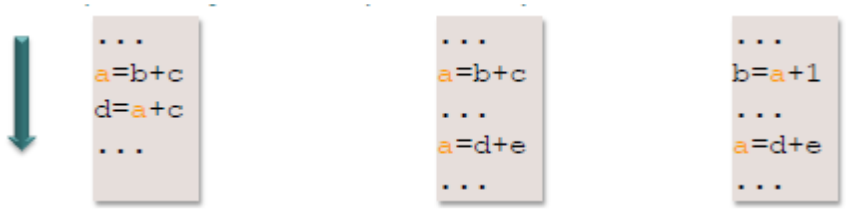
A este paralelismo que se puede detectar en distintos niveles de un código secuencial, se le ha denominado *paralelismo funcional*.

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 B2 respecto a B1):
  - RAW o dependencia verdadera
  - WAW o dependencia de salida
  - WAR o antidependencia

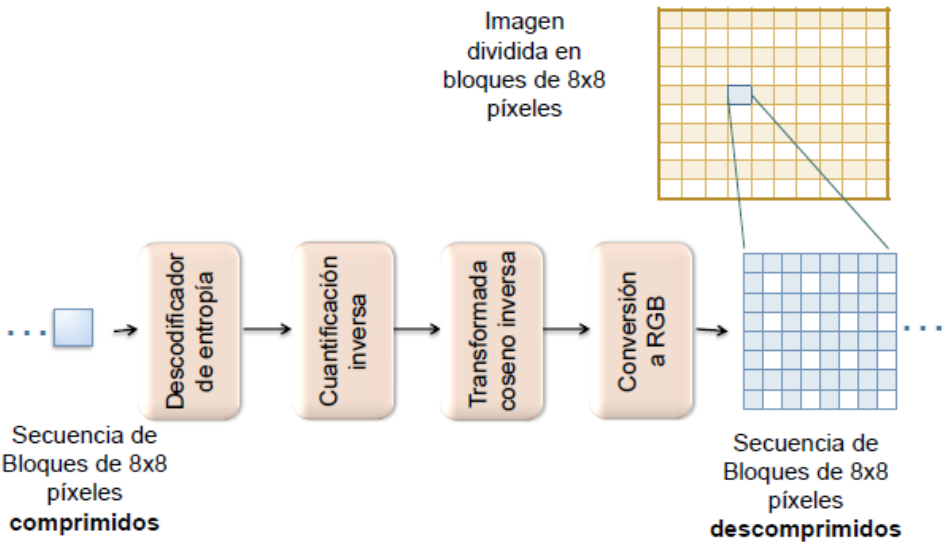


Paralelismo implícito en una aplicación

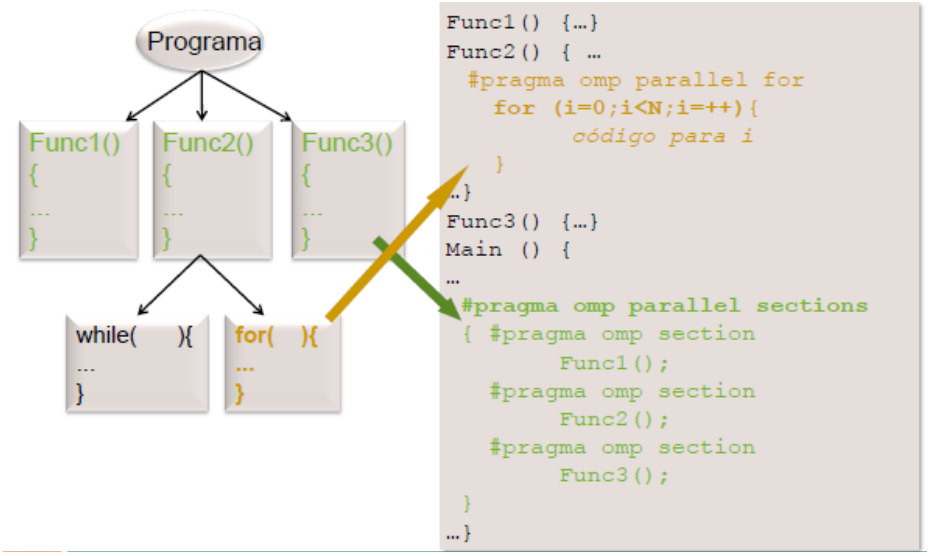
- Paralelismo de tareas (task parallelism o TLP (task level par))
  - Se encuentra extrayendo la estructura lógica de funciones de una aplicación
  - Está relacionado con el paralelismo a ***nivel de función***
  - en esta estructura, los bloque son funciones, y las conexiones entre ellos reflejan el flujo de datos entre funciones. Analizando esta estructura se puede encontrar paralelismo entre las funciones. Equivaldría al paralelismo a nivel de función dentro del código de alto nivel.
- Paralelismo de datos (data parallelism o DLP (data level par))
  - Se encuentra implícito en las operaciones con estructuras de datos (vectores y matrices)
    - 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***
  - El paralelismo de datos se puede extraer de los bucles analizando las operaciones realizadas con la misma estructura de datos en las diferentes iteraciones del bucle.

El paralelismo puede también clasificarse en función de la ***granularidad*** o magnitud de la tarea (nº de operaciones) candidata a la paralelización, que se suele hacer corresponder con los distintos niveles de paralelismo. El grano más pequeño (grano fino) se asocia generalmente al paralelismo entre operaciones o instrucciones, y el grano grueso, al paralelismo entre programas.

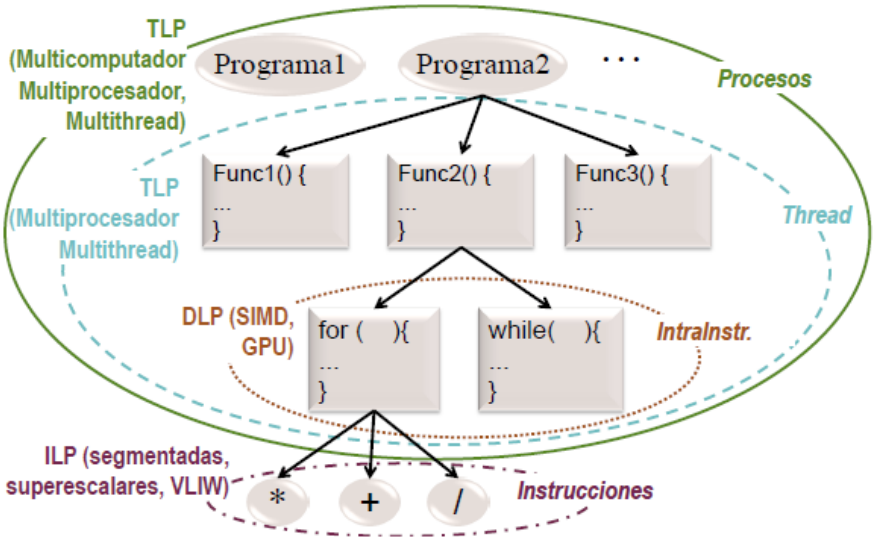
Estructura de funciones lógica de una aplicación. Ej: decodificador JPEG



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



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



Nivel de paralelismo explícito. Unidades en ejecución en un computador

- Instrucciones
  - La UC de un core o procesador gestiona la ejecución de instrucciones por la unidad de procesamiento (CPU)
- 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

Relación entre paralelismo implícito, explícito y arquitecturas paralelas

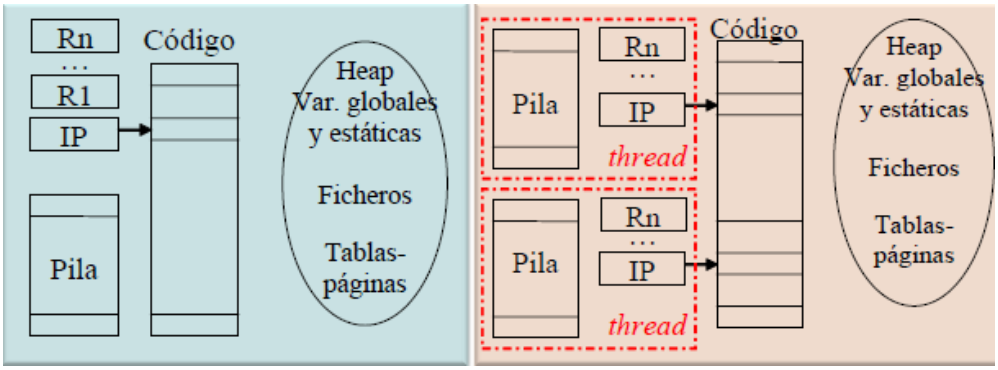
En la anterior figura se relacionan los distintos niveles en los que se encuentra el paralelismo implícito en el código, con los niveles en los que se puede hacer explícito y con arquitecturas paralelas que aprovechan el paralelismo. El paralelismo entre programas se utiliza a nivel de procesos. En el momento en el que se ejecuta un programa, se crea el proceso asociado al programa. El paralelismo disponible entre funciones se puede extraer para utilizarlo a nivel de procesos o hebras. El paralelismo dentro de un bucle se puede extraer también a nivel de procesos o hebras. Se puede aumentar la granularidad asociando un mayor número de iteraciones del bucle a cada unidad a ejecutar en paralelo. El paralelismo entre operaciones se puede aprovechar en arquitecturas con paralelismo a nivel de instrucción (ILP) ejecutando en paralelo las instrucciones asociadas a estas operaciones independientes.

El hardware se encarga de gestionar la ejecución de las instrucciones. A nivel superior, el SO se encarga de gestionar la ejecución de unidades de mayor granularidad, *procesos y hebras.* Los SO's multihebra permiten que un proceso se componga de una o varias hebras (hilos). Una hebra tiene su propia pila y contenido de registros, entre ellos IP (puntero de instrucciones) que almacena la dirección de la siguiente instrucción a ejecutar de la hebra, pero comparte el código, las variables globales y otros recursos (como por ejemplo archivos abiertos) con las hebras del mismo proceso. Estas características hacen que las hebras se puedan crear y destruir en menor tiempo que los procesos, y que la comunicación, sincronización y conmutación entre hebras de un proceso sea más rápida que entre procesos. Todo ello permite que las hebras puedan tener una granularidad menor que los procesos.

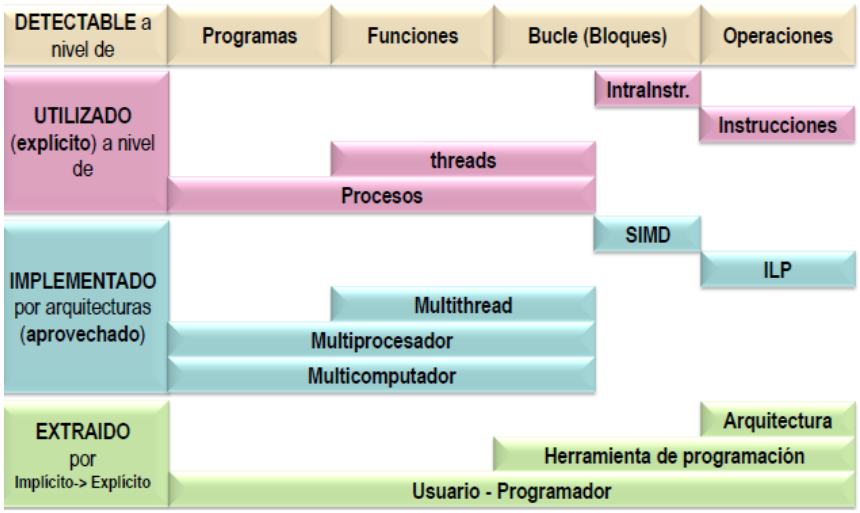
El paralelismo implícito en el código de una aplicación se puede hacer explícito a nivel de instrucciones, a nivel de hebras, a nivel de procesos o dentro de una instrucción.

Nivel de Paralelismo explícito. Threads vs 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 CP y el puntero de pila o stack pointer
- Para comunicar threads de un proceso se usa la memoria que comparten



Detección, utilización, implementación y extracción del paralelismo



En los procesos ILP superescalares o segmentados la arquitectura extrae paralelismo. Para ello, eliminan dependencias de datos falsas entre instrucciones y evitan problemas debidos a dependencias de datos, a dependencias de control y dependencias de recursos. En estos procesadores, la arquitectura extrae paralelismo implícito en las entradas en tiempo de ejecución (*dinámicamente*). El grado de paralelismo de las instrucciones aprovechado se puede incrementar con ayuda del compilador y del programador. En general, podemos definir el grado de paralelismo de un conjunto de entradas a un sistema, como el máximo número de entradas del conjunto que se puede ejecutar en paralelo. Debido a las dependencias entre entradas, este grado máximo será generalmente inferior al número de entradas del conjunto. Para los procesadores las entradas son instrucciones. Fuera del hardware se determinan las instrucciones que se pueden ejecutar en paralelo. El análisis de dependencias entre instrucciones en este caso es estático. El compilador es el principal responsable de extraer paralelismo.