ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA
Ciencias de la Computación

PANORAMA DE LA EXPLOTACIÓN DEL PARALELISMO A NIVEL DE INSTRUCCIÓN.

Computación Digital

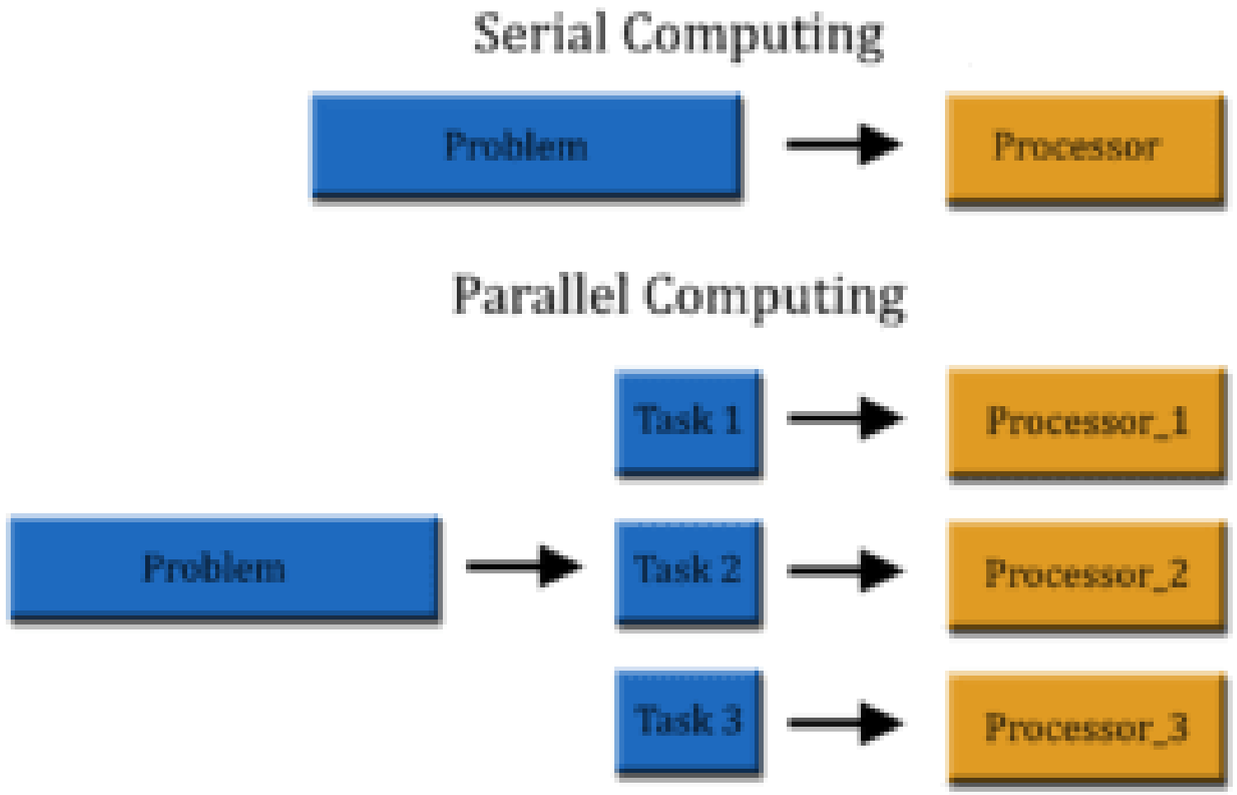
NRC: 24857

Llumiquinga M. Jerson
Jaguaco J.Jonathan
Jami Klever

Tabla de contenido

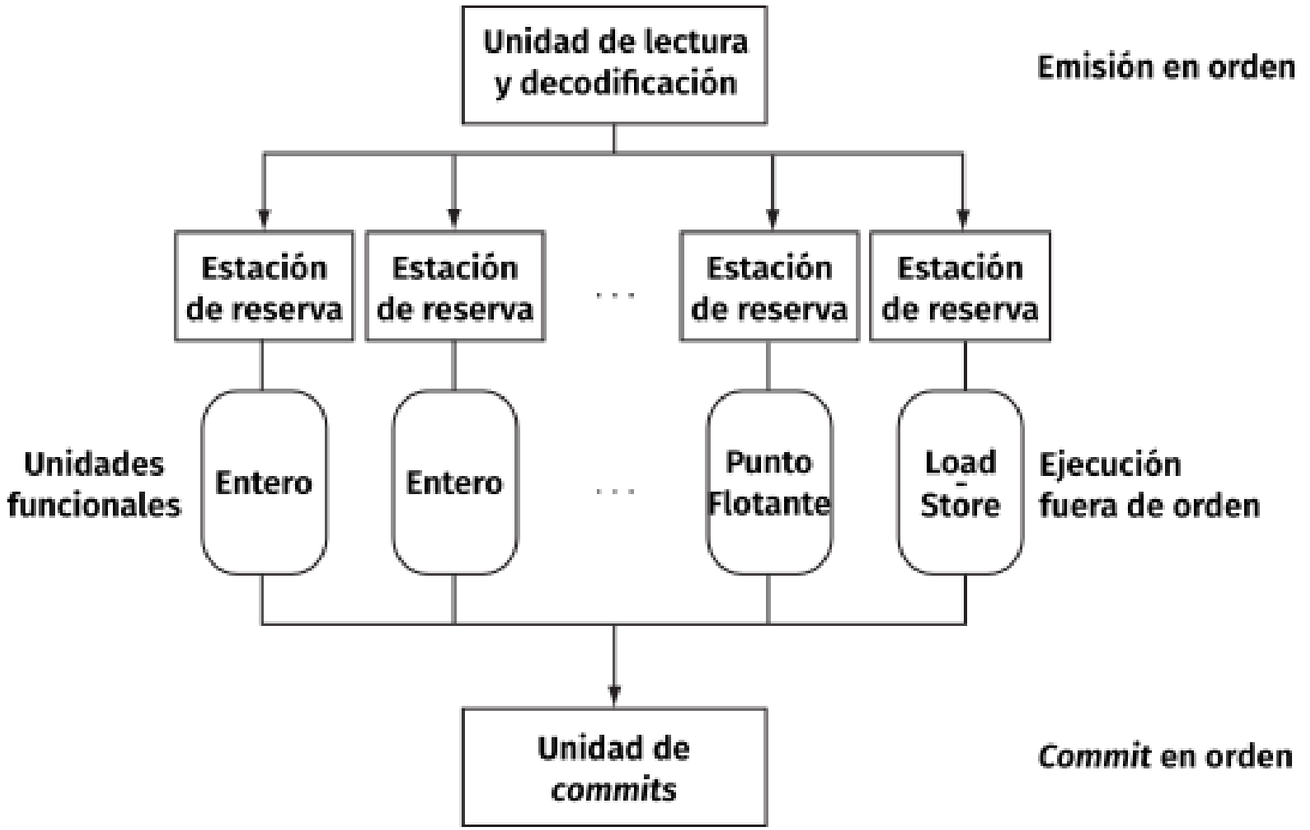
PARALELISMO

El paralelismo en computación consiste en el uso simultáneo de múltiples procesadores o núcleos que ejecutan cada uno una serie de instrucciones que conforman las distintas partes en las que se ha descompuesto un problema computacional para resolver.

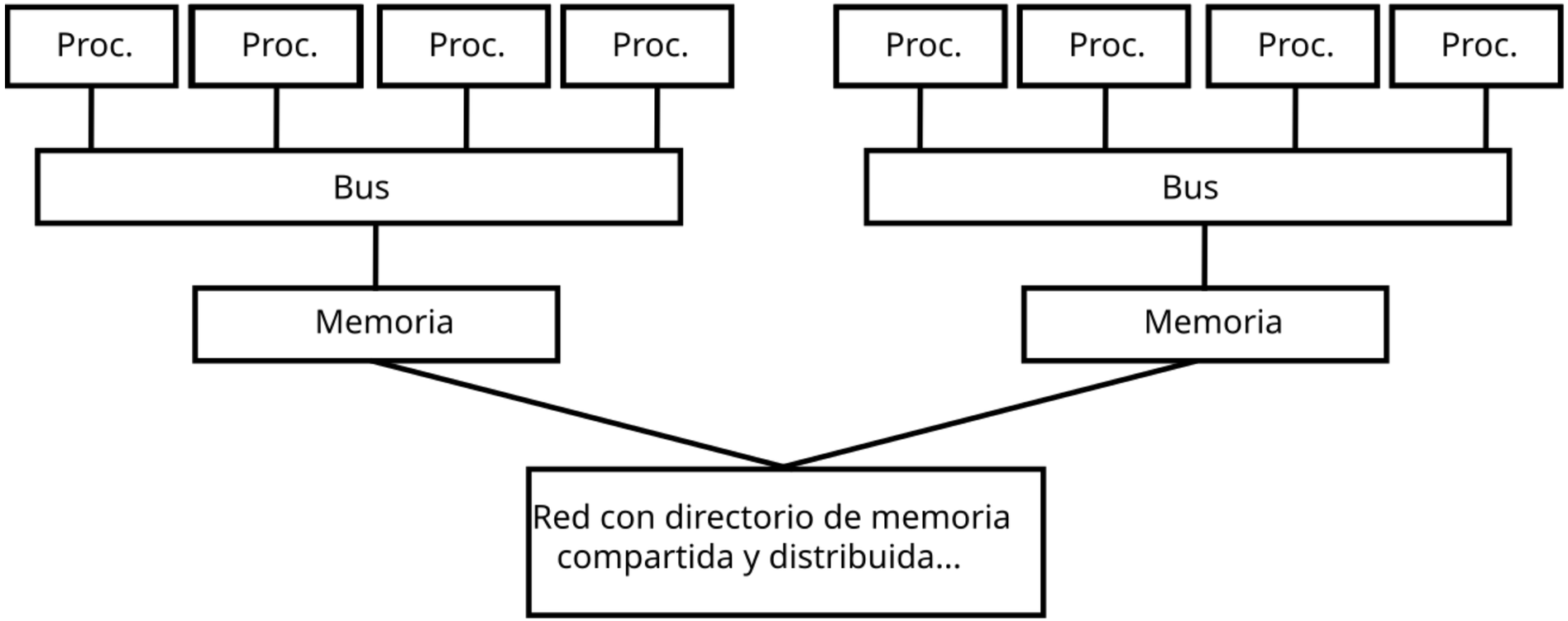


PARALELISMO A NIVEL DE INSTRUCCIÓN

En la arquitectura de computadores, el paralelismo a nivel de instrucción (ILP) es una técnica que permite ejecutar varias instrucciones de un mismo programa al mismo tiempo, siempre que no dependan entre sí.



PANORAMA DE LA EXPLOTACIÓN DEL PARALELISMO A NIVEL DE INSTRUCCIÓN (ILP)



Técnicas de Explotación ILP

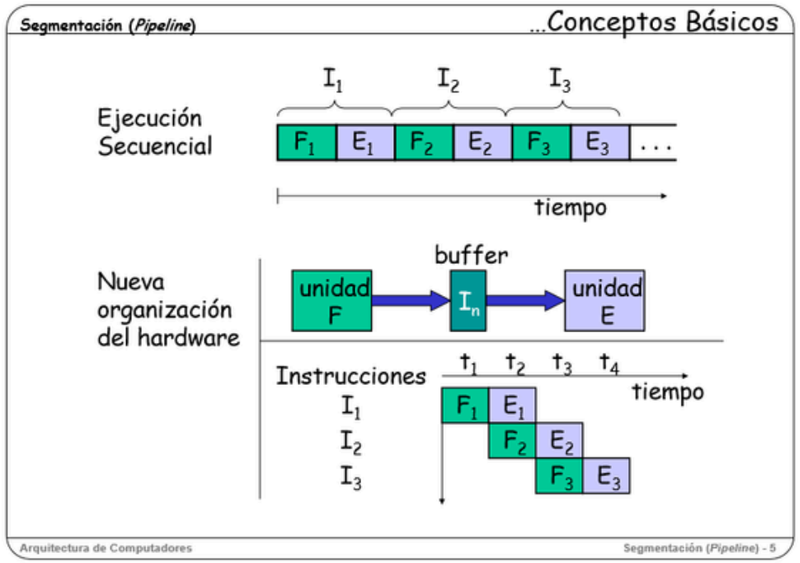
Los procesadores logran ejecutar varias instrucciones al mismo tiempo, gracias a distintas técnicas de hardware y software.

¿POR QUÉ HACEN FALTA ESTAS TÉCNICAS?

El objetivo del ILP es que el procesador ejecute la mayor cantidad posible de instrucciones en cada ciclo de reloj. Para lograrlo, se aplican métodos que permiten identificar y aprovechar las instrucciones independientes, evitando que el procesador se detenga innecesariamente.

Pipeline

El procesador divide la ejecución de instrucciones en varias etapas (como buscar, decodificar y ejecutar). Así, varias instrucciones pueden estar en diferentes fases al mismo tiempo, como una línea de montaje en una fábrica.



Ejecución fuera de orden

Si una instrucción necesita esperar datos, el procesador no se detiene, sino que ejecuta otras instrucciones que ya tienen sus datos listos. Esto mantiene el procesador trabajando de manera continua y eficiente.

Renombramiento de registros

Para evitar conflictos cuando varias instrucciones usan el mismo registro, el procesador asigna diferentes registros físicos. Así, se eliminan dependencias innecesarias y se permite que más instrucciones se ejecuten en paralelo.

Predicción de saltos y ejecución especulativa

Cuando el flujo del programa depende de una decisión (como un “if”), el procesador predice cuál será el camino correcto y sigue ejecutando instrucciones de ese camino. Si la predicción es correcta, se gana tiempo; si no, se corrige el flujo. Esto reduce las pausas y mantiene el pipeline lleno.

Desenrollado de bucles (Loop Unrolling)

El compilador puede modificar los bucles para que realicen varias operaciones por iteración, generando más instrucciones independientes que el procesador puede ejecutar en paralelo.

Reordenamiento de instrucciones

El compilador también puede cambiar el orden de las instrucciones para evitar esperas y mantener ocupadas las unidades del procesador, aprovechando mejor el paralelismo disponible.

Ejemplo

Bucle independiente

Imaginemos un bucle que suma los elementos de una lista

Compilador

El compilador puede desenrollar el bucle y reordenar las sumas.

Hardware

El hardware, por su parte, ejecuta esas sumas fuera de orden, renombra registros y predice saltos.

Rendimiento

De esta forma, el procesador puede ejecutar muchas instrucciones al mismo tiempo, incluso si el código original no estaba pensado para ello.

DESAFÍOS DEL ILP

DEPENDENCIAS DE DATOS

RAW: Una instrucción necesita de una datos que aun no se ha producido

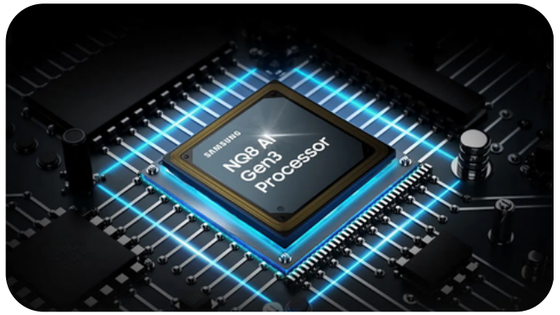
```
a = 5 + 3
b = a * 2
imprimir(b)
```

WAR: Se intenta escribir un valor antes que otra variable lo lea.

```
a = 10
b = a + 5
a = 20
imprimir(b)
```

WAW: Dos instrucciones escriben en el mismo lugar

```
1. a = 7
2. a = 10
imprimir(a)
```



SOLUCIONES:

1. Stalling
2. Reordenamiento de instrucciones
3. Renombramiento de registros

DESAFÍOS DEL ILP

RIESGOS DE CONTROL

- Procesador no sabe que instrucción ejecutar cuando existe un bucle

```
if (x > 0) {
  hacerA();
} else {
  hacerB();
}
```

¿CÓMO SE SOLUCIONA?

RIESGOS ESTRUCTURALES

- Dos o mas instrucciones requieren del mismo recurso, pero estos son insuficientes

```
A = B + C
D = E * F
```

¿CÓMO SE SOLUCIONA?

LIMITACIONES FISICAS

LIMITACIONES FISICAS

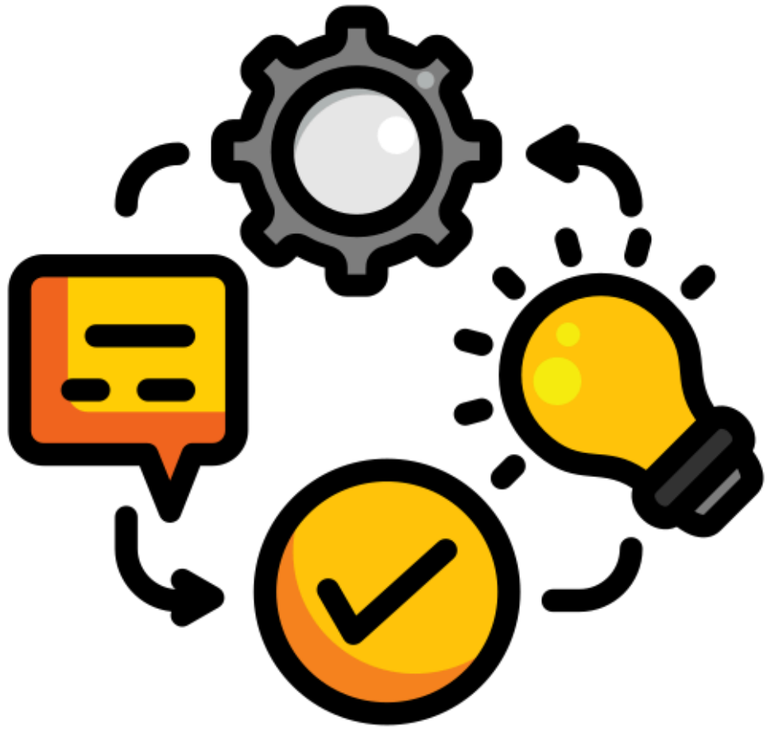
ANCHO LIMITE DEL PIPELINE

PIPELINE

Ejemplo simple de etapas típicas:

1. IF – Fetch (traer la instrucción)
2. ID – Decode (decodificar)
3. EX – Execute (ejecutar)
4. MEM – Acceder a memoria
5. WB – Write Back (escribir el resultado)

¿POR QUE ES UNA LIMITANTE?

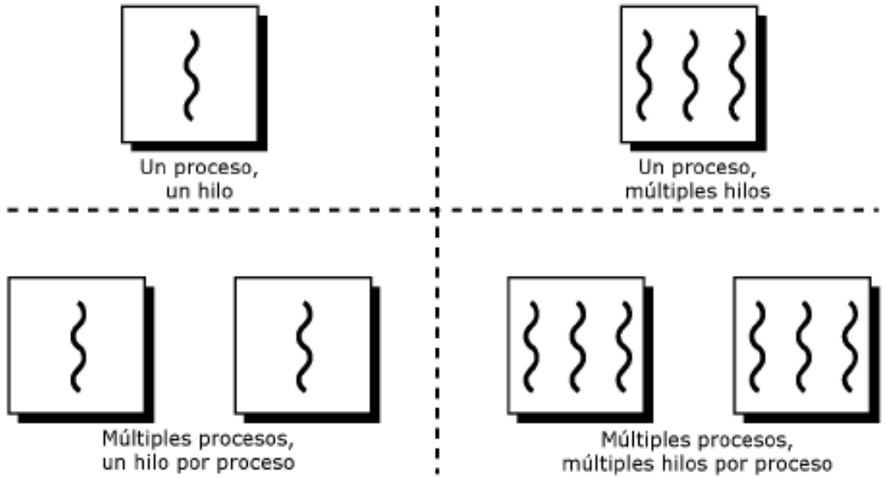


SOLUCIONES A FUTURO

TRANSICIÓN HACIA TLP

¿En qué se diferencia del ILP?

- ILP ejecuta varias instrucciones al mismo tiempo dentro de un solo hilo.
- TLP ejecuta varios hilos al mismo tiempo, en distintos núcleos o unidades lógicas.



CONCLUSIONES:

El ILP fue clave para acelerar programas, pero sus desafíos —como dependencias, riesgos y límites físicos— muestran que ya no basta. Por eso, hoy combinamos ILP con TLP para seguir escalando el rendimiento de forma eficiente.

Las técnicas de ILP permiten que los procesadores ejecuten múltiples instrucciones simultáneamente, mejorando significativamente el rendimiento sin necesidad de aumentar el número de núcleos.

Estas técnicas, tanto de hardware como de software, son esenciales para aprovechar al máximo los recursos internos del procesador y mantener un flujo continuo de ejecución eficiente.

MUCHAS GRACIAS POR SU ATENCIÓN