Aritmética Computacional
Punto Flotante y Algoritmos de División en Hardware

El límite del Punto Fijo (FXP) frente a magnitudes científicas

La aritmética de punto fijo obliga a usar un único factor de escala. En cadenas largas de cálculo científico, magnitudes extremas sufren pérdidas masivas de precisión o provocan desbordamientos. La solución es permitir que cada magnitud flote con su propio factor de escala.

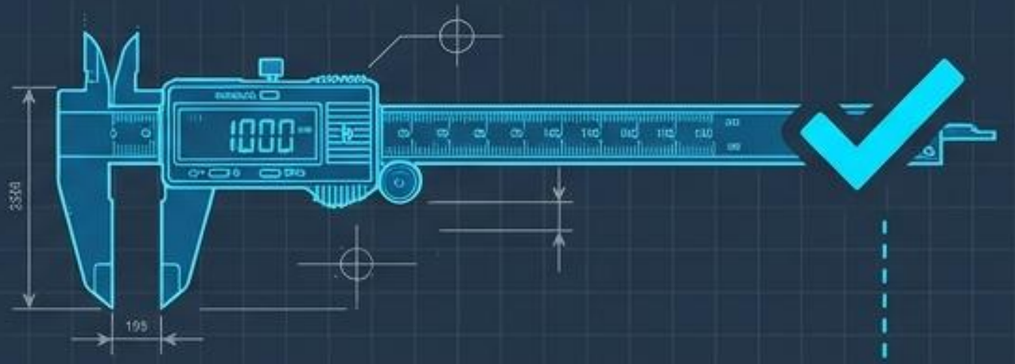
Punto Fijo - FXP

- Rango limitado
- Factor de escala único
- Alta pérdida de significancia en múltiples operaciones

FXP: scale_factor = 2^-Q

DESBORDAMIENTO
PRECISIÓN CERO

Punto Flotante - FLP



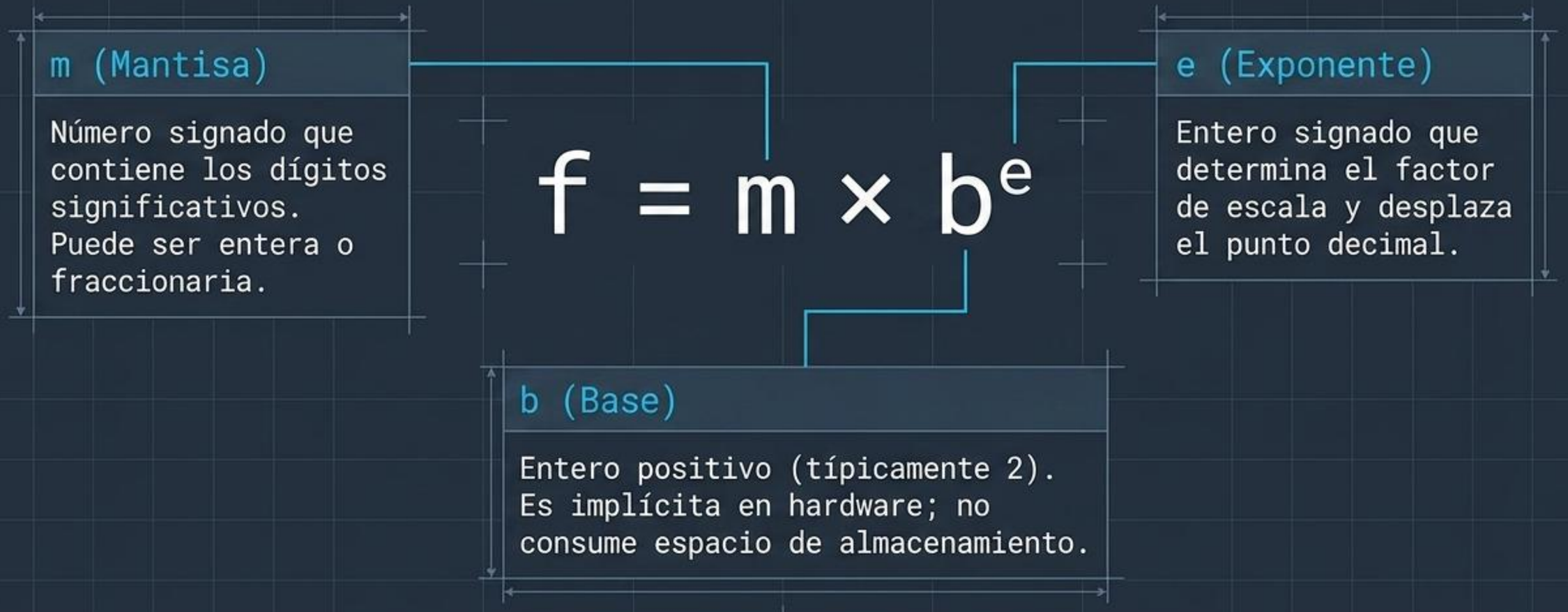
- Rango dinámico
- Factor de escala individual
- Máxima precisión conservada hasta el final del cálculo

FLP: value = mantissa * 2^exponent

PRECISIÓN MÁXIMA

Anatomía estructural del Punto Flotante

Todo número real en punto flotante se expresa separando sus bits de significancia de su magnitud escalar.



Normalización y el Hidden Bit

El proceso de normalización ajusta la mantisa para eliminar ceros sin significancia, maximizando la precisión. En base 2, los números normalizados siempre inician con un 1.

Estado No Normalizado 0.001011×2^{5} → 2^{3}

Estado Normalizado 0.101100×2^{3}

Hidden Bit

~~1~~.101100

Optimización: El primer bit de significancia (1) se asume implícitamente por el hardware, ahorrando un bit de almacenamiento.

El espectro de valores representables en FLP

El verdadero cero constituye la línea divisoria, flanqueado por infinitesimales y limitados por magnitudes cuasi-infinitas.

Aritmética de extremos: Infinito y Epsilon

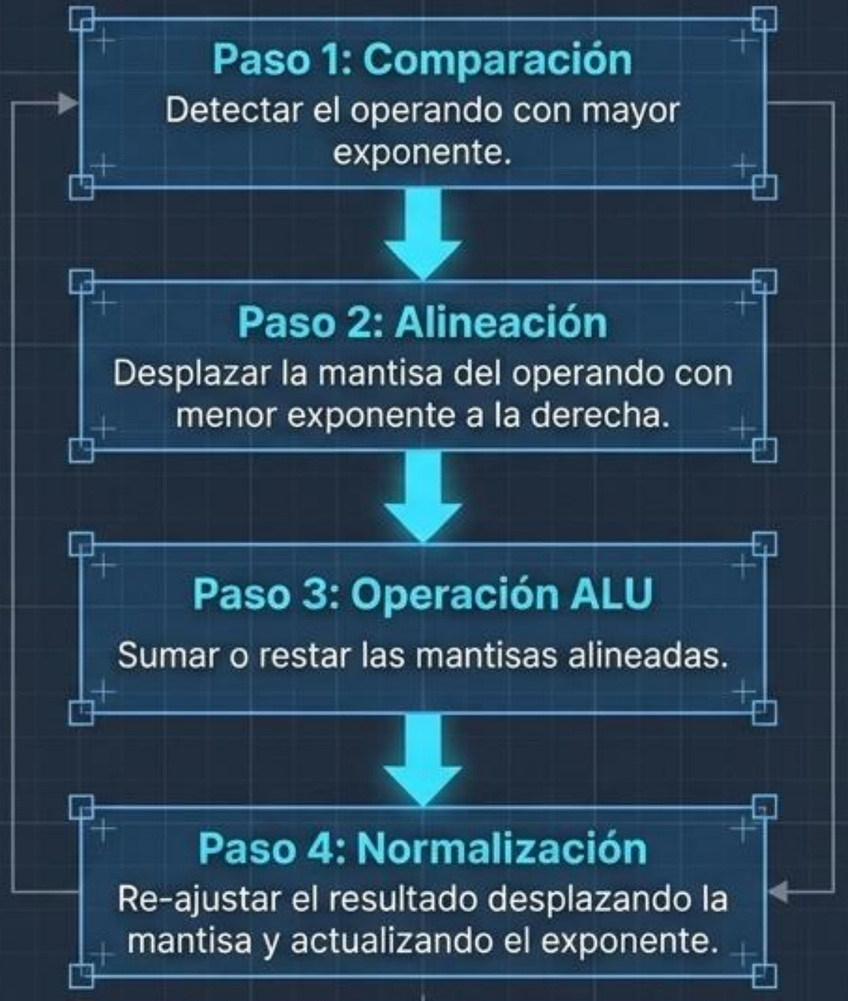
El hardware debe resolver matemáticamente las interacciones de valores límite sin interrumpir el flujo de datos, salvo en excepciones fatales.

Suma/Resta	Multiplicación	División
$\infty \pm N = \infty$ $\varepsilon \pm \varepsilon = \varepsilon$	$\infty \times N = \infty$ $\varepsilon \times \varepsilon = \varepsilon$	$N / \infty = \varepsilon$ $\varepsilon / N = \varepsilon$
Casos Indeterminados		
$\infty - \infty = \infty$	$\infty \times \varepsilon = \infty\varepsilon$	$\infty / \infty = \infty$

Trampa de Hardware: Intento de N / ε o $N / 0$ activa el flag de división por cero y dispara una interrupción inmediata.

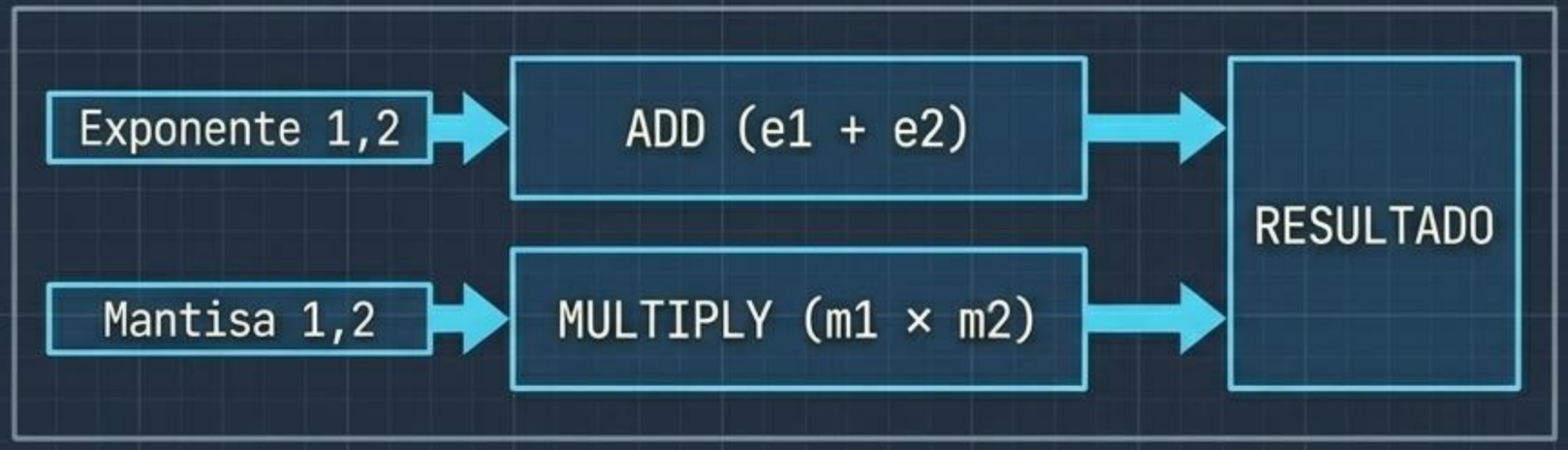
La secuencia lógica de Suma y Resta

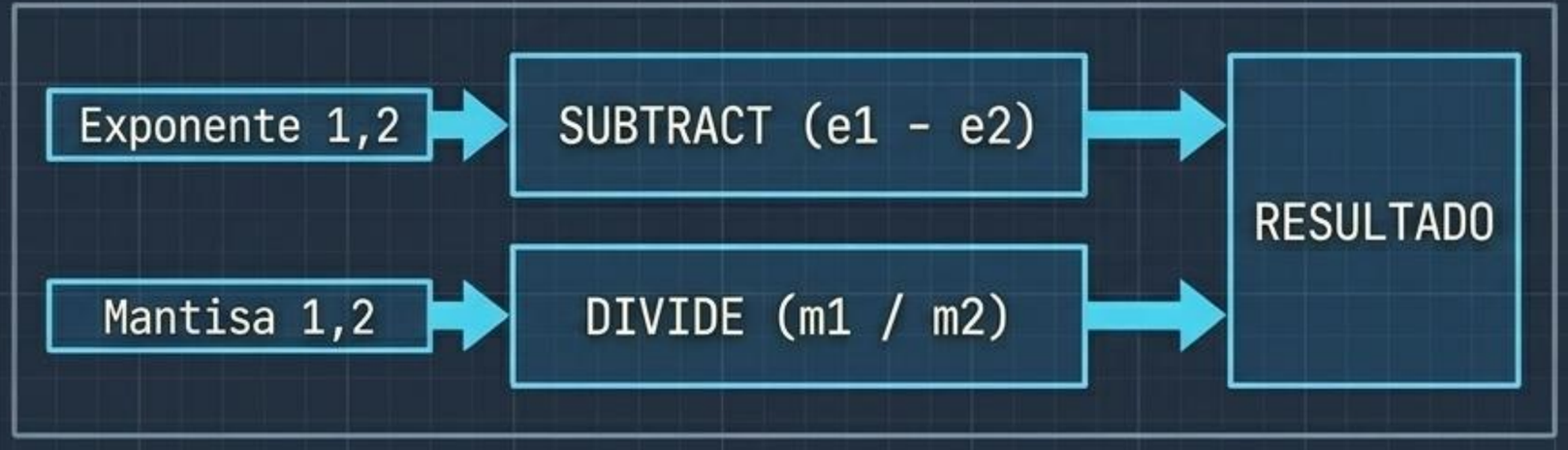
Para sumar o restar mantisas, ambos operandos deben compartir exactamente el mismo exponente. El hardware procesa esta alineación secuencialmente.



Procesamiento paralelo: Multiplicación y División

A diferencia de la suma, la multiplicación y división calculan mantisa y exponente simultáneamente, enfrentando el desafío del **overflow virtual**.





Alerta: Overflow Virtual

Un resultado multiplicativo puede violar la regla $0 \leq |m| < 2$.

$1[0 >> \longrightarrow |m1| + 0$
$>> \longrightarrow |m1] - 1$

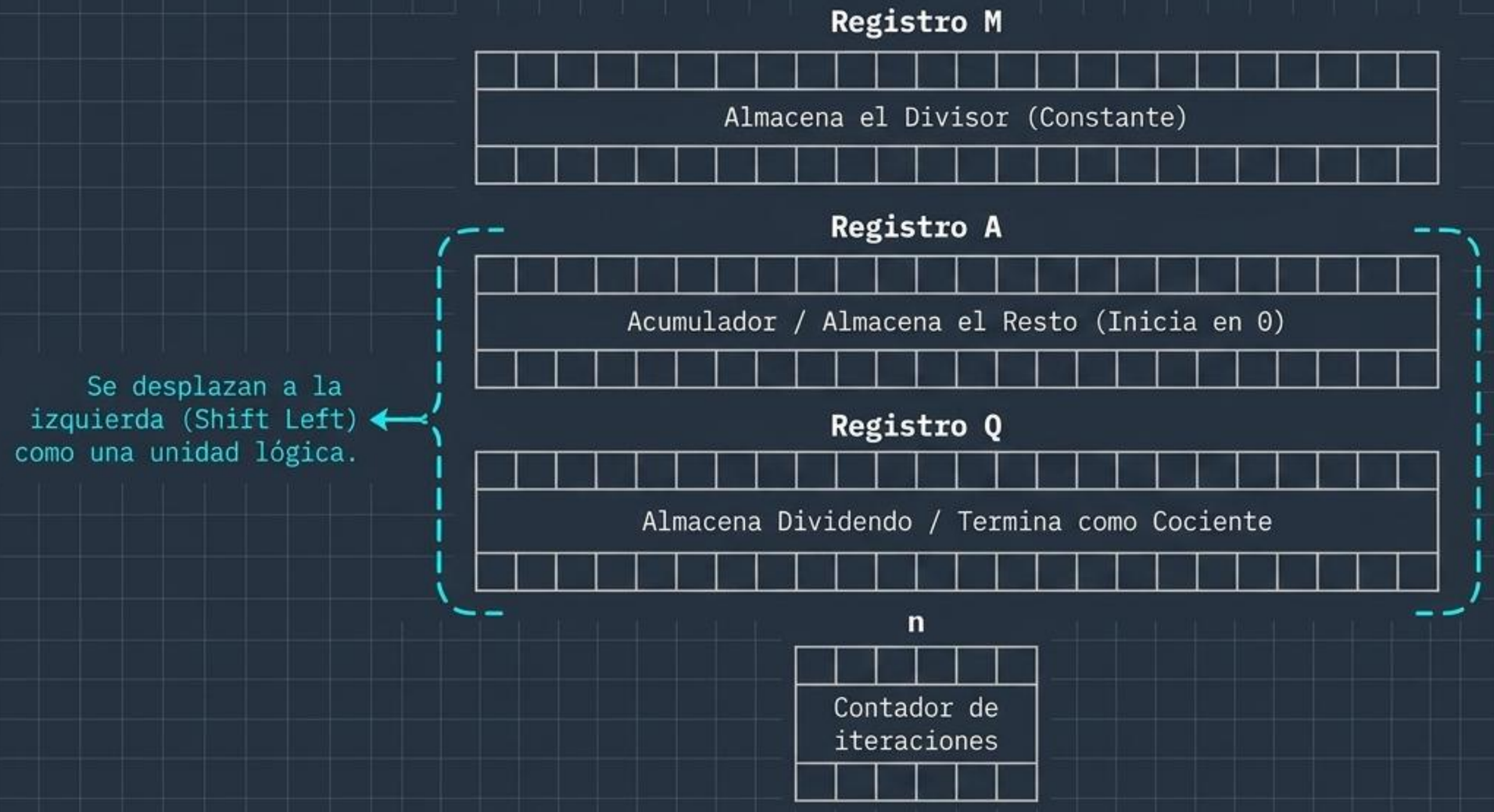
Correr la mantisa un lugar a la derecha y sumar 1 al exponente.

Transición al silicio: El motor de la división

Para resolver la fracción de mantisas, el procesador debe recurrir a algoritmos cíclicos a nivel de registros de hardware.

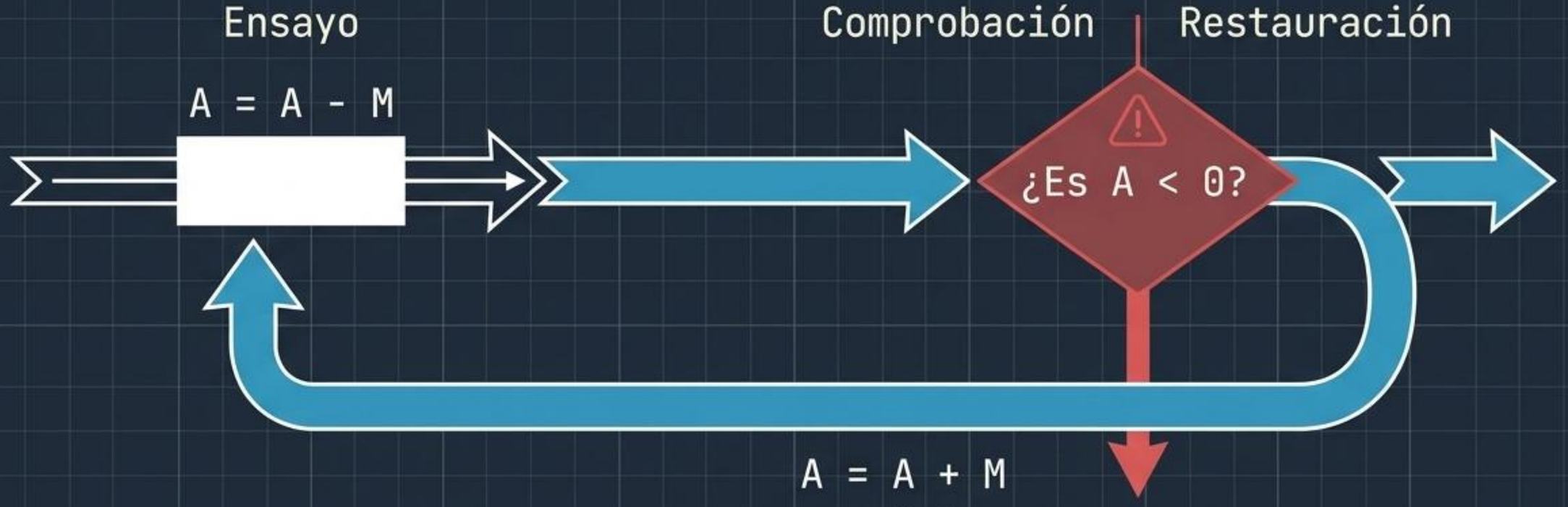
Arquitectura de los registros de división

El proceso iterativo requiere cuatro registros fundamentales trabajando de forma orquestada.



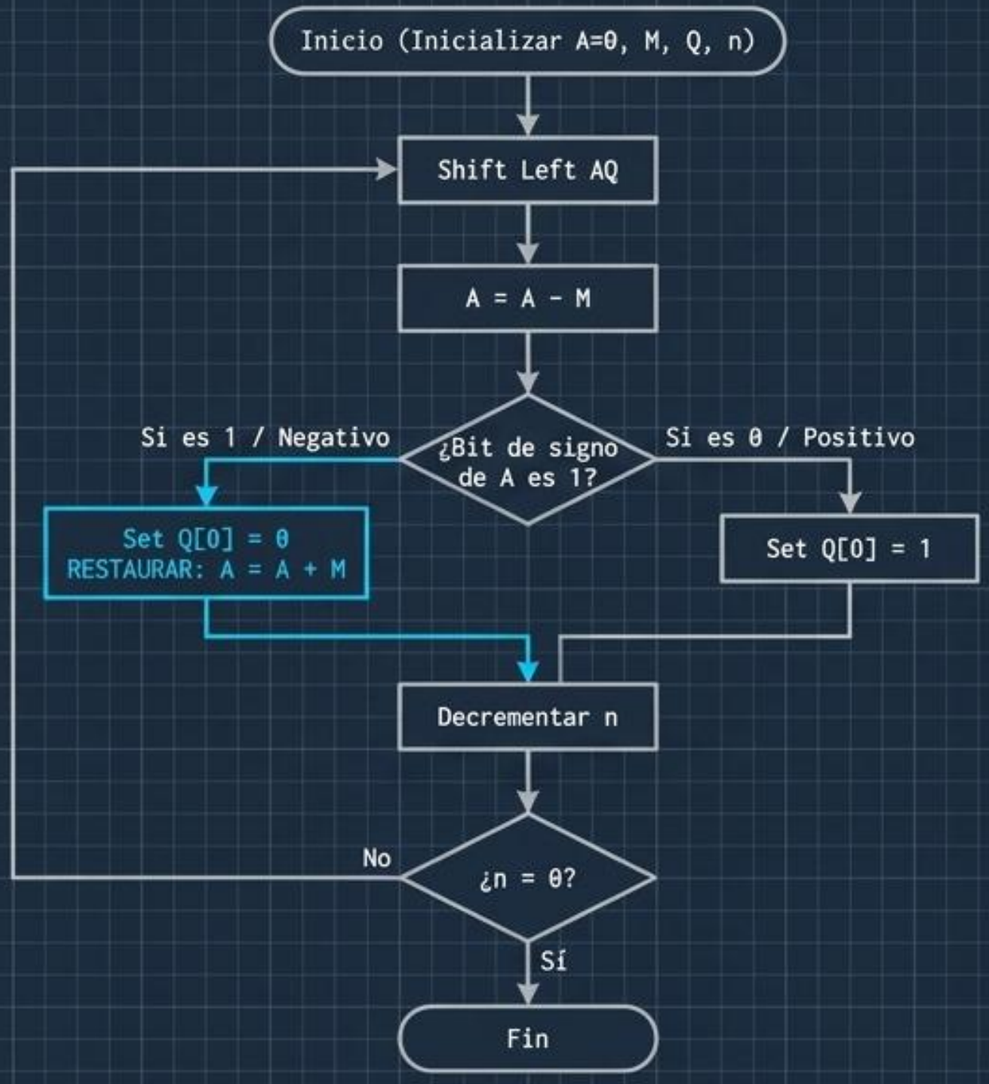
El principio de Restauración (Restoring)

El algoritmo clásico opera bajo la lógica de ensayo y error. Si la resta falla, se deshace el cálculo restaurando el valor original.



Flujo Lógico: Restoring Division

La mecánica del algoritmo fuerza ciclos adicionales en la unidad lógico-aritmética cada vez que el resto temporal resulta negativo.

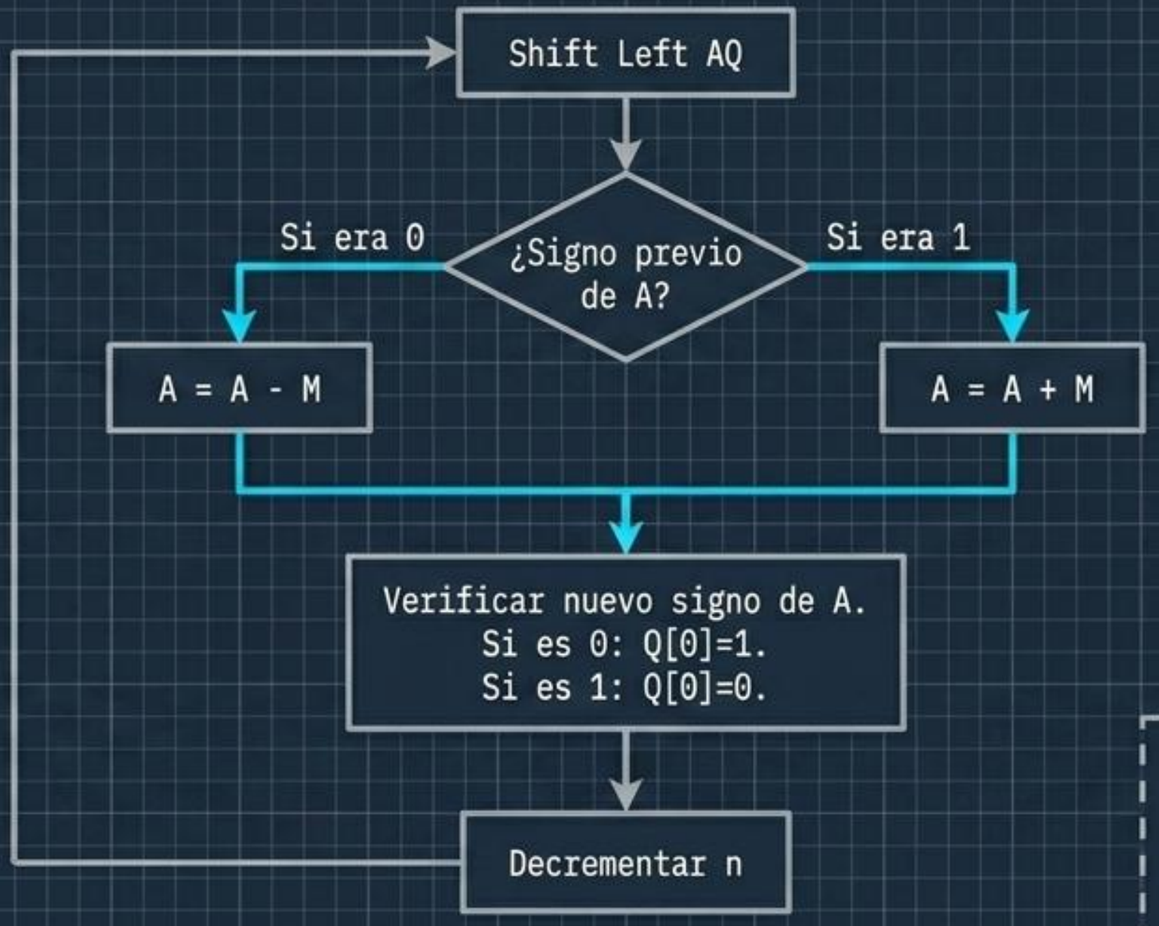


El principio Sin Restauración (Non-Restoring)

Para acelerar el cómputo, elimina el paso de "deshacer". Acepta la deuda temporal y la compensa sumando en el ciclo siguiente.

Flujo Lógico: Non-Restoring Division

Al bifurcar la decisión antes de la operación, el procesador elimina la necesidad de restaurar valores.



Ajuste Final

Al finalizar n=0, si el signo de A es 1, realizar ajuste único final: A = A + M.

Síntesis Arquitectónica: Restoring vs Non-Restoring

El diseño de hardware busca el equilibrio exacto entre la complejidad del control lógico y la velocidad del reloj.

Criterio de Evaluación	Algoritmo Restoring	Algoritmo Non-Restoring
Mecanismo de Error	Deshacer el paso anterior	Compensar en el siguiente paso
Operación Primaria	Solo Resta (A - M)	Suma o Resta (A ± M) dependiente del signo
Velocidad Computacional	Más lento (requiere paso de restauración)	Más rápido (menor número de operaciones por bit)
Complejidad del Control	Simple (flujo lógico único)	Compleja (lógica de ramificación por signo)
Ajuste Final del Resto	Automáticamente correcto al terminar	Puede requerir un ajuste de suma final (A = A + M)