Diseño de la jerarquía de memoria

William Stallings, Organización y Arquitectura de Computadores
Andrew S. Tanenbaum, Organización de Computadoras
Linda Null y Julia Lobur, Computer Organization and Architecture

John Hennessy – David Patterson
Arquitectura de Computadores – Un enfoque cuantitativo
(1a edición, capítulo 8) (4th edition, chapter 5)

El principio de localidad

(Locality of reference)

En general, todos los programas de computadora favorecen una parte del espacio de direcciones en un determinado instante de tiempo. Esta hipótesis tiene dos dimensiones:

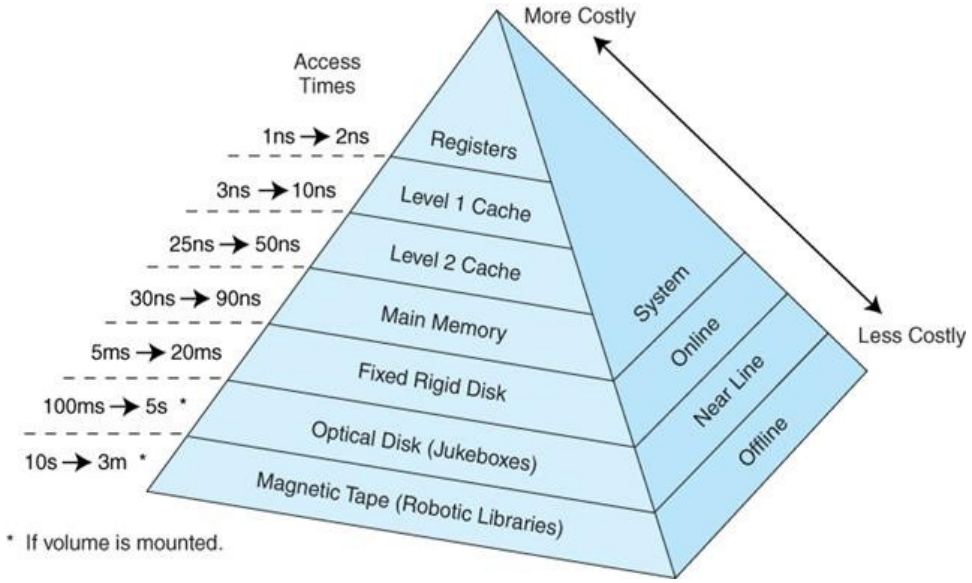
- **Localidad temporal:** Si se referencia un elemento, tenderá a ser nuevamente referenciado pronto.
- **Localidad espacial:** Si se referencia un elemento, los elementos cercanos a él tenderán a ser referenciados pronto. La localidad secuencial es un caso particular de la localidad espacial dado por el acceso a los elementos consecutivos de un arreglo.

La localidad es un tipo de comportamiento predecible que se cumple en los sistemas de computadora. Los sistemas que presentan una localidad fuerte son buenos candidatos a ser optimizados por diferentes técnicas.

Localidad en los programas: consecuencia de la estructura secuencial de los programas.

Localidad en los datos: consecuencia de las estructuras secuenciales de datos.

La jerarquía de memoria



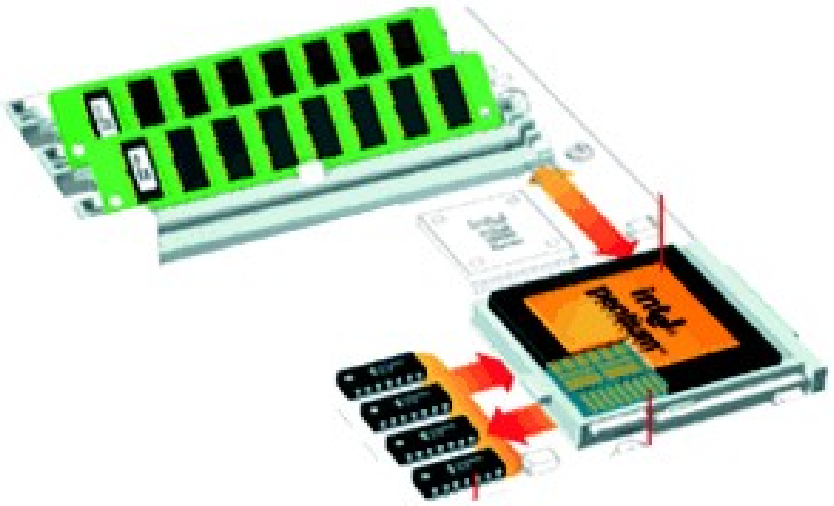
Junto con la expansión del espacio de memoria y el aumento de velocidad de los procesadores aparece el problema de performance de la memoria.

Las memorias grandes de bajo consumo (DRAM semiconductoras, capacitores) tienen desempeños inferiores a las memorias basadas en transistores (SRAM).

Una jerarquía de memoria es una reacción natural a la localidad y la tecnología (el hardware más pequeño es más rápido). Su objetivo es el aumento del rendimiento.

velocidad – costo – volumen - potencia

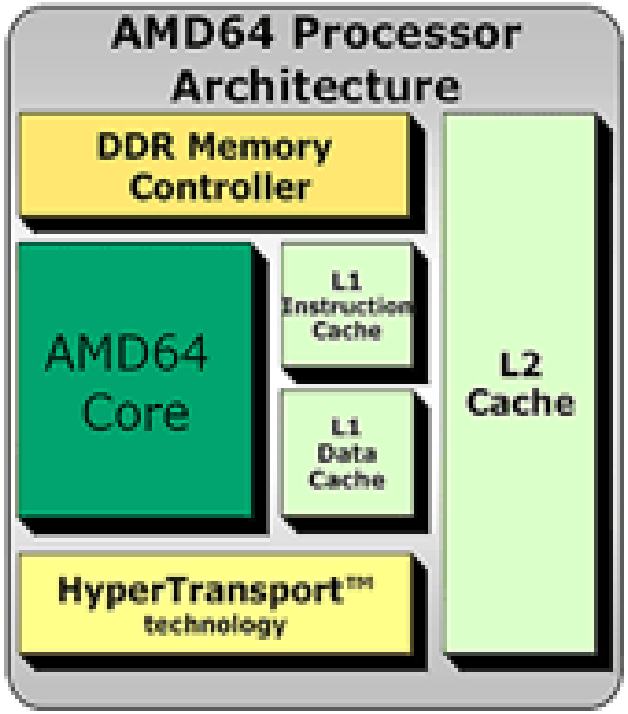
Memoria caché



Para mantener a los microprocesadores funcionando a máxima velocidad se inserta una pequeña cantidad de memoria rápida entre la memoria principal lenta y el microprocesador. El objetivo es albergar las instrucciones que se repiten o los datos que se acceden más fecuentemente (IBM 360, 1968).

De esta manera el procesador pude ejecutar a máxima velocidad mientras las instrucciones y los datos estén en el cache (cache hit). Caso contrario (cache miss) debe detenerse hasta que las instrucciones sean captadas de la memoria principal.

Cuanto más grande sea el cache, más tiempo estará el procesador a máxima velocidad, pero mayor será el consumo de potencia y de transistores.



El cache crece

La idea de caché ha crecido junto con el tamaño y la complejidad de los procesadores. Un Pentium 4 tiene 2 MB de memoria caché interna, lo que representa más del doble del espacio de memoria completo del 8088 que se utilizó para construir las primeras PC.

El cache se divide

Las memorias pequeñas pueden ser direccionadas más rápidamente. Incluso las dimensiones físicas de la oblea pueden enlentecerlas. La solución para aumentar el tamaño es agregarle un caché a la memoria caché (figura) organizado en niveles (L1, L2, L3).

Performance

Debe tenerse en cuenta que el mero tamaño de la memoria caché o el número de niveles disponibles no son buenos indicadores de la performance del procesador. Diferencias en las organizaciones (como por ejemplo entre AMD e Intel) hacen que sea complicado comparar utilizando las especificaciones de caché.

Definitivamente, un aumento en el tamaño del caché no significa necesariamente un aumento proporcional de performance. La única forma de estimar el desempeño de un procesador es mediante el uso de *benchmarks* apropiados.

Memoria Caché

Principales conceptos

Se trata de una memoria más rápida y más pequeña que la memoria principal, que almacena copias de las zonas de memoria más utilizadas.

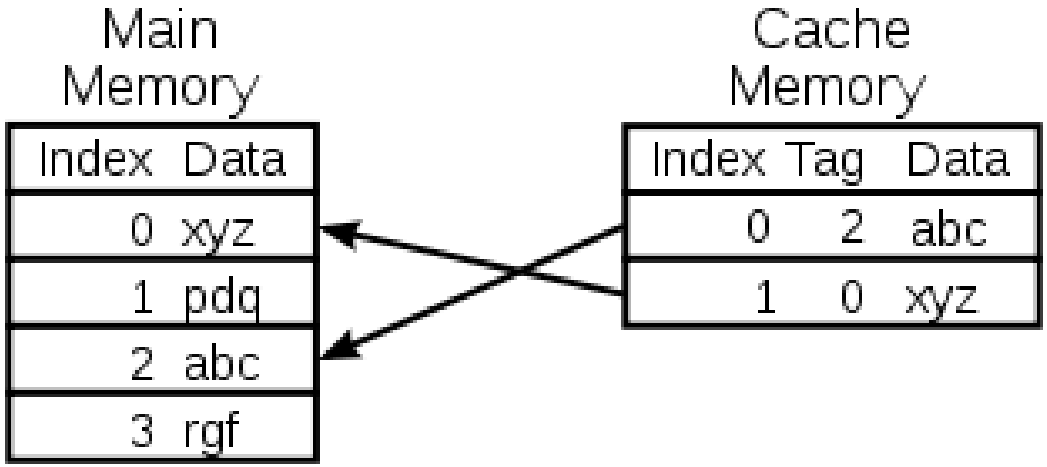
Se diseña tal que es invisible para el programador (ISA). Se trata de una mejora en la **organización** (que no afecta, en principio, el diseño de los programas).

El objetivo es reducir el tiempo PROMEDIO de acceso a memoria.

Cuanto mayor cantidad de accesos a memoria sean abastecidos por el caché, más cerca estará la latencia **promedio** de la latencia de la memoria caché que de la latencia de la memoria principal.

La latencia de un acceso a memoria aislado aumenta.

Funcionamiento



Linea de caché 0
Almacena la dirección de memoria 2
Contiene ‘abc’

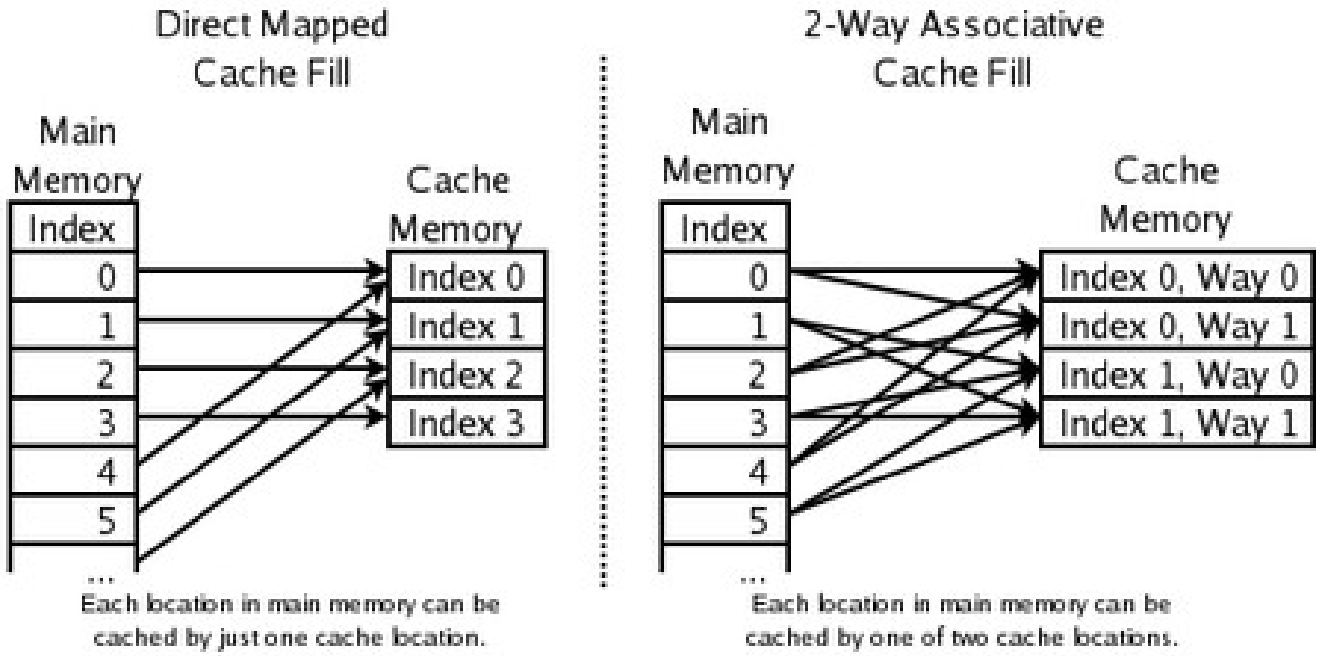
Memoria Caché

Términos usuales

- El principio de localidad.
- Caché de datos y caché de instrucciones. Arquitectura Harvard.
- TLB (*Translation Lookaside Buffer*). Memoria virtual: MMU (*Memory Management Unit*)
- Bloques o lineas de caché (*caché lines/blocks*) (8-512 b).
- Fallo/acierto (*caché miss/hit*).
- Detención (*stall*).
- Tasa de aciertos (*caché hit rate*).
- Política de reemplazo (*replacement policy*). LRU (*Least Recently Used*).
- Política de escritura (*write policy*) cómo actuar ante una escritura del caché de datos.
  - *write-through*: actualizar inmediatamente la linea.
  - *write-back*: esperar a que la linea sea reemplazada. En este caso un fallo produce una doble demora.
- Dirty block.
- Coherencia de caché (*caché coherence*).

Memoria Caché

Asociatividad



4-Way associative caché fill

Fully associative caché

Memoria Caché

Fallos de caché

En orden de importancia:

- **Fallo de lectura de instrucciones:** Detención completa del procesador.
- **Fallo de lectura de datos:** Las instrucciones que no necesitan los datos pueden continuar.
- **Fallo de escritura de datos:** Puede hacerse en background.

Con la tecnología moderna, el procesador puede ejecutar cientos de instrucciones mientras se copia un bloque de memoria al caché.

Fuentes de fallos de caché:

- **Forzosos:** La primera vez que se utiliza un bloque (fallos de arranque en frio o de primera referencia)
- **Capacidad:** Deben descartarse bloques por falta de espacio y luego hay que recuperarlos.
- **Conflicto:** Debido a la ubicación impuesta por la asociatividad (fallos de colisión)

Memoria Caché

Criterios de diseño

Múltiples variables a tener en cuenta:

- Cache dividida (datos e instrucciones) o unificada.
- Tamaño de bloque vs. velocidad media de acceso (hit rate vs. latency). En general, la menor frecuencia de fallos no compensa la mayor penalización de los fallos.
- Mayor asociatividad vs. complejidad y por lo tanto lentitud.
- Tamaño del caché vs. tiempo de acceso, poniendo en peligro la velocidad de clock de la CPU.
- Múltiples niveles (costo vs performance).

Compromiso difícil de resolver. No existe una solución general. Simulaciones con benchmarks y programas reales. Address Traces.

Esfuerzo de diseño - Potencia - Area

Discusión sobre productividad

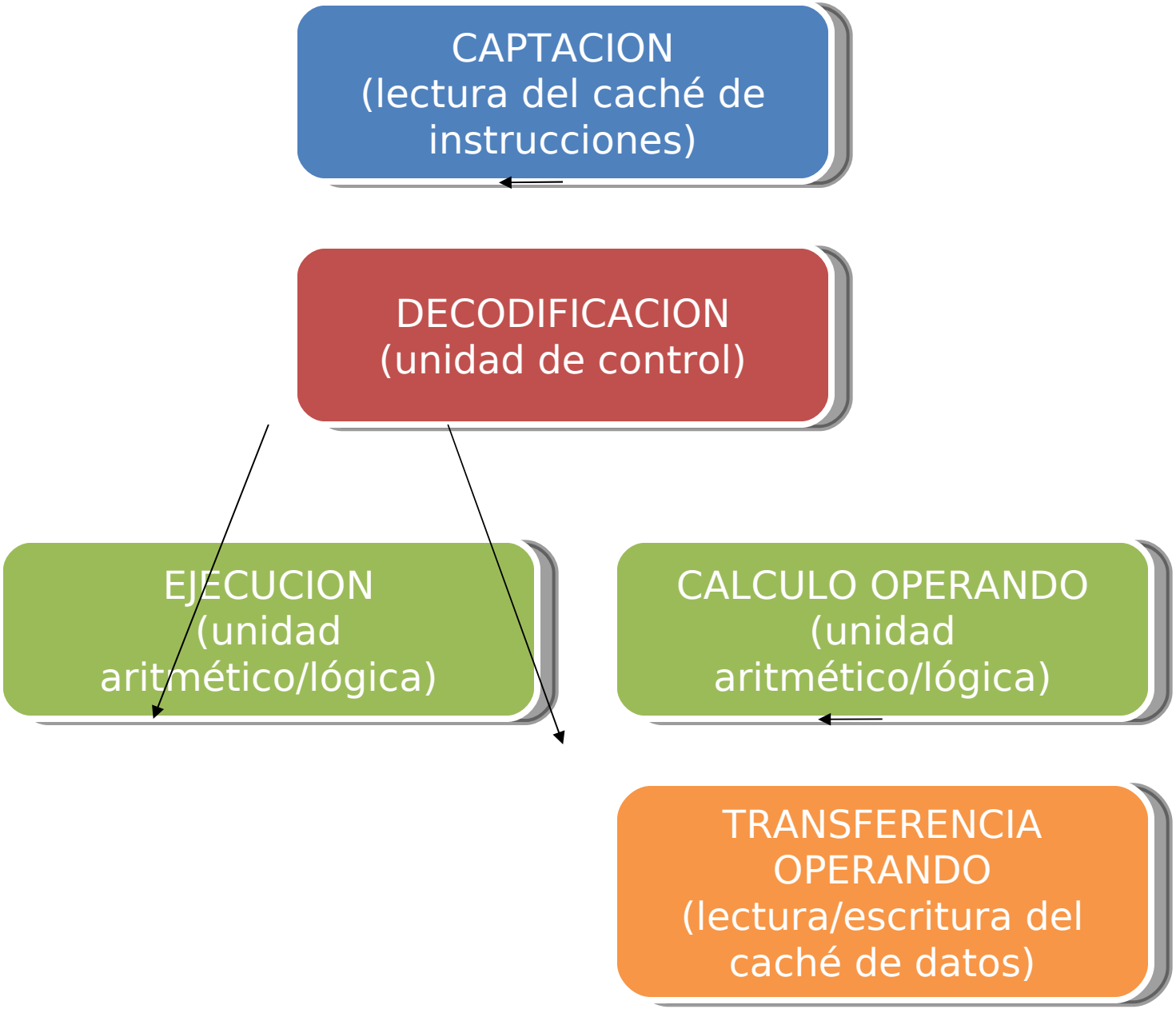
La importancia de la memoria caché en un diseño moderno.

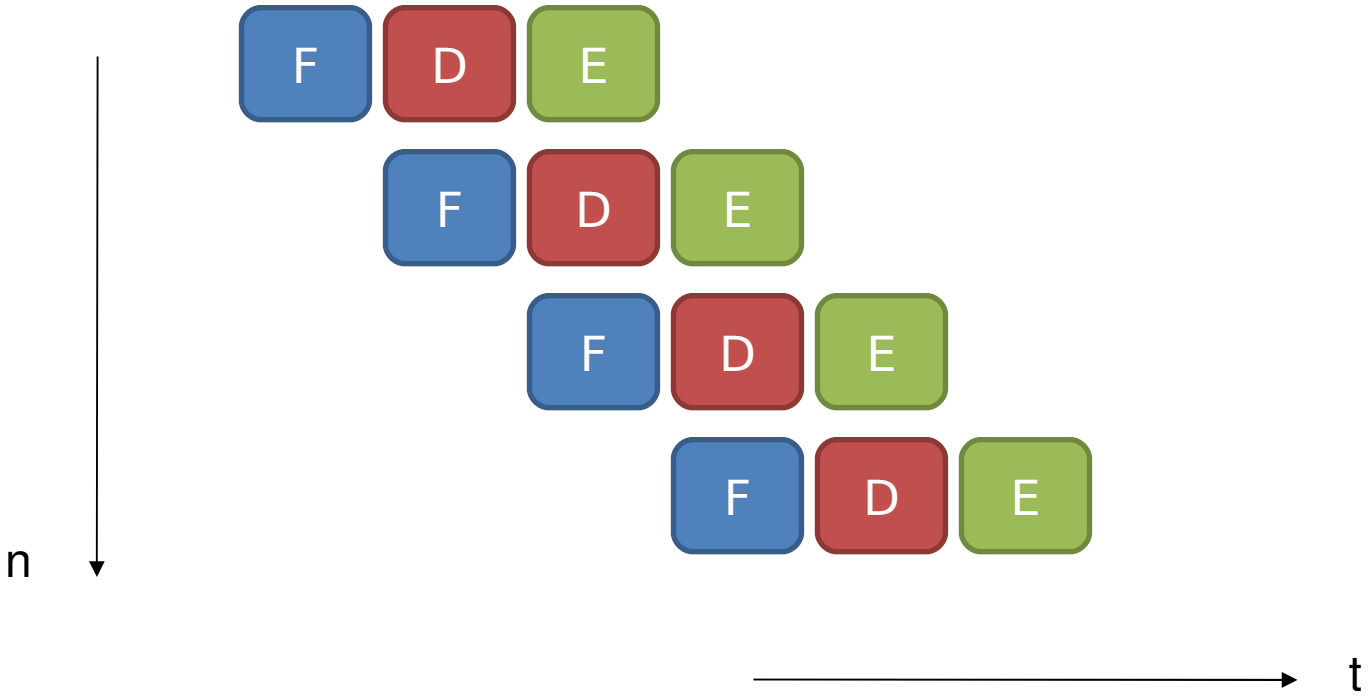
Suponiendo:

- Ciclo de instrucción segmentado: Captación-Decodificación-Ejecución.
- Cache ideal (hit rate 100%, latencia 1 ciclo).
- Caché dividida (datos e instrucciones).

Cada etapa del cauce segmentado requiere un ciclo de reloj y no hay conflictos de hardware.

PRODUCTIVIDAD = 1 instrucción/ciclo


CAPTACION
(lectura del caché de instrucciones)
DECODIFICACION
(unidad de control)
EJECUCION
(unidad aritmético/lógica)
CALCULO OPERANDO
(unidad aritmético/lógica)
TRANSFERENCIA OPERANDO
(lectura/escritura del caché de datos)



Segmentación del cauce de instrucciones

Discusión sobre productividad

Es posible una productividad de más de una instrucción por ciclo de reloj?

Memoria principal entrelazada

ver

Memoria virtual

- TLB (*Translation Lookaside Buffer*).
- MMU (*Memory Management Unit*)