1. **¿Cuáles son los elementos que componen la arquitectura Von Neumann? ¿Qué restricciones nos pone la memoria Von Neumann a los programadores?**

Según la Arquitectura Von Neumann, una computadora está constituida por una CPU (unidad central de procesamiento), una memoria principal y unidades de E/S. Estos componentes se interconectan con un bus del sistema, de modo que puedan llevar a cabo la función básica de la computadora, que es ejecutar programas.

La CPU se compone de dos elementos: un intérprete de instrucciones (la UC, la *unidad de control*) y un módulo de uso general para las funciones aritméticas y lógicas (la ALU, la *unidad aritmético-lógica*).

La arquitectura Von Neumann se basa en tres conceptos claves:

- Los datos y las instrucciones se almacenan en una sola memoria de escritura-lectura.
- Los contenidos de esta memoria se direccionan indicando su posición, sin considerar el tipo de dato contenido en la misma.
- La ejecución se produce siguiendo una secuencia de instrucción tras instrucción (a no ser que dicha secuencia se modifique explícitamente).

Estos conceptos pueden generar restricciones a los programadores. Que los datos y las instrucciones se almacenen en una sola memoria genera limitaciones de espacio y que los datos se accedan indicando su posición, sin considerar el contenido puede generar accesos erróneos; el programador deberá asegurarse que los accesos a memoria sean los adecuados, es decir, no querer leer datos y accidentalmente leer instrucciones, o viceversa.

2. **¿Dónde se encuentran los vectores de interrupción en un sistema de atención vectorizada?**

En un sistema de atención vectorizada, los vectores de interrupción se encuentran en un área de memoria reservada, denominado tabla de vectores de interrupción.

3. **¿Qué es un bus? Describa los tipos, arbitraje y técnicas de sincronización. Mencione diferencias entre bus PCI y bus SCSI.**

Un bus es un camino de comunicación entre dos o mas dispositivos. Una característica clave es que se trata de un medio de transmisión compartido. Al bus se conectan varios dispositivos, y cualquier señal transmitida por uno de esos dispositivos esta disponible para que los otros dispositivos conectados al bus puedan acceder a ella. Solo un dispositivo puede transmitir con éxito en un momento dado.

Usualmente, un bus esta constituido por varios caminos de comunicación, o líneas. Cada línea es capaz de transmitir señales binarias representadas por 1 y por 0.

El bus que conecta los componentes principales del computador (procesador, memoria, E/S) se denomina *bus del sistema*. A cada línea del bus se le asigna un significado o una función particular, y se pueden clasificar en tres grupos:

- **Bus de datos:** proporciona el camino para trasmitir datos entre los módulos del sistema. Puesto que cada línea solo puede transportar un bit a la vez, el número de líneas (anchura del bus) determina cuantos bits se puede transmitir al mismo tiempo.
- **Bus de dirección:** se utiliza para designar la fuente o el destino del dato situado en el bus de datos. La anchura del bus de direcciones determina la máxima capacidad de memoria posible del sistema.
- **Bus de control:** se utiliza para controlar el acceso y el uso de las líneas de datos y de direcciones. Como el bus de datos y direcciones se comparte por todos los componentes, debe existir una forma de controlar su uso. Las señales de control transmiten tanto ordenes como información de temporización entre los módulos del sistema. Las señales de temporización indican la validez de los datos y las direcciones. Las señales de ordenes especifican las operaciones a realizar

Las líneas del bus se pueden dividir en dos tipos genéricos, dando lugar a dos tipos de buses:

- **Dedicadas:** uso de líneas separadas para direcciones y para datos.
- **Multiplexadas:** la información de dirección y datos se transmite a través del mismo conjunto de líneas.

La ventaja del multiplexado es el uso de menos líneas, cosa que ahorra espacio y costes. La desventaja es que necesita una circuitería mas compleja en cada módulo.

Como en un instante dado solo una unidad puede transmitir a través del bus, se requiere algún método de arbitraje. Los métodos se pueden clasificar como:

- **Centralizado:** un único dispositivo hardware, denominado *arbitro*, es responsable de asignar los tiempos del bus.
- **Distribuido:** no existe un árbitro. Cada módulo dispone de lógica para controlar el acceso y los módulos actúan conjuntamente para compartir el bus. Antes de cada transferencia, se debe definir cual modulo sería el árbitro del bus para dicha transferencia.

La forma en la que se coordinan los eventos del bus se llama *temporización*. Los buses utilizan dos tipos de temporización:

- **Sincrónica:** la presencia de un evento en el bus esta determinada por un reloj. El bus incluye una línea de reloj a través de la que se trasmite una secuencia en la que se alternan intervalos regulares de igual duración a uno y a cero (ciclos de reloj). Todos los dispositivos pueden leer la línea de reloj, y todos los eventos empiezan al principio del ciclo de reloj.
- **Asincrónica:** la presencia de un evento del bus es consecuencia y depende de que se produzca un evento previo.

La temporización sincrónica es más fácil de implementar y comprobar. Sin embargo, es menos flexible. Debido a que todos los dispositivos de un bus sincrónico deben utilizar la misma frecuencia de reloj, el sistema no puede aprovechar mejoras en las prestaciones de los dispositivos. Con la temporización asincrónica, pueden compartir el bus una mezcla de dispositivos lentos y rápidos, utilizando tanto las tecnologías más antiguas, como las más recientes.

Si se conecta un gran numero de dispositivos al bus, las prestaciones pueden disminuir. Por esto, la mayoría de los computadores usan varios buses.

Las principales diferencias entre un bus PCI y uno SCSI son:

- Propósito, el bus PCI es de proposito general, mientras que el SCSI es para conectar dispositivos de almacenamiento de datos.
- Velocidad: el PCI es de altas velocidad, mientras que el SCSI tiene velocidades de transferencia de datos mas bajas.
- Topologia: en el PCI todos los dispositivos se conectan directamente al bus principal, en el SCSI se conectan todos los dispositivos en cadena.
- Configuracion: en el PCI los dispositivos son plug-and-play, mientras que en el SCSI tienen que ser configurados a mano.
- Longitud de cableado: el bus PCI tiene una longitud maxima de cableado limitada, mientras que el SCSI puede tener longitud de varios metros.

4. **Describa que se debe tener en cuenta para diseñar una caché.**

Para diseñar una caché se deben tener en cuenta los siguientes criterios:

- **Tamaño:** nos gustaría que el tamaño sea lo suficiente pequeño para que el coste total medio por bit se aproxime al de la memoria principal sola, y que fuera lo suficientemente grande como para que el tiempo de acceso medio total sea próximo al de la caché sola. Las cachés grandes suelen ser ligeramente mas lentas que las pequeñas. El tamaño también esta limitado por las superficies disponibles de chip y de tarjeta. Como las prestaciones de la caché son muy sensibles al tipo de tare, es imposible predecir un tamaño “óptimo”.

- **Política de ubicación:** Ya que hay menos líneas de caché que bloques de memoria principal, se necesita un algoritmo que haga corresponder bloques de memoria principal a líneas de caché. Hay tres tipos de correspondencia:
    - **Correspondencia Directa:** se corresponde cada bloque de memoria a solo una línea posible de caché. La correspondencia se expresa como:

      Nro. línea caché = *Nro. bloque Mem.Princ.* **mod** *Nro. líneas en caché*

      Esta técnica es sencilla y poco costosa de implementar. Su principal desventaja es que hay una posición concreta de caché para cada bloque dado; si un programa referencia repetidas veces a palabras de dos bloques diferentes asignados a la misma línea, dichos bloques estarían intercambiando continuamente en la caché, y la tasa de acierto sería baja.
    - **Correspondencia Asociativa:** cada bloque de memoria puede cargarse en cualquier línea de la caché. Así, hay flexibilidad para que cualquier bloque sea reemplazado cuando se va a escribir uno nuevo en la caché. La principal desventaja es la compleja circuitería necesaria para examinar en paralelo las etiquetas de todas las líneas de caché cuando se quiere buscar algún elemento.
    - **Correspondencia Asociativa por Conjuntos:** la caché se divide en conjuntos, cada bloque de memoria puede cargarse en cualquier línea del conjunto que le corresponde. Las relaciones son:

      Nro. conjunto = *Nro. bloque ref.* **mod** *Nro. conjuntos caché*

      Con la correspondencia asociativa por conjuntos, la etiqueta de una dirección de memoria es mas corta que con la correspondencia totalmente asociativa, y el numero de comparaciones que se debe hacer en un conjunto es mucho menor.

      El uso de dos líneas por conjunto es el caso mas común, mejorando significativamente la tasa de aciertos respecto de la correspondencia directa.
- **Política de reemplazo:** Una vez que se ha llenado la caché, para introducir un nuevo bloque debe sustituirse uno de los bloques existentes. Para el caso de correspondencia directa, solo hay una posible línea para cada bloque particular. Para las técnicas asociativas se requieren algoritmos de sustitución, los cuatro más comunes son:
    - **LRU:** el *"Least-Recentrly Used"*, se sustituye el bloque que se ha mantenido en la caché por mas tiempo sin haber sido referenciado.
    - **FIFO:** el "*First-in-First-Out*", se sustituye aquel bloque que ha estado mas tiempo en la caché.
    - **LFU:** el "*Least-Frequently Used*", se sustituye aquel bloque del conjunto que ha experimentado menos referencias.
    - **Aleatorio:** se elige una línea al azar entre las posibles candidatas.
- **Política de escritura:** Hay dos casos a considerar cuando se tiene que reemplazar un bloque de caché, si el bloque fue modificado o no. Si no fue modificado, no hay problema en sobrescribirlo con un nuevo bloque. Si fue modificado, entonces la memoria principal debe actualizarse.

  Además, más de un dispositivo puede tener acceso a la memoria principal. Si una palabra ha sido modificada solo en la caché, la correspondiente de memoria no es válida para ser leída por otros dispositivos. Si un dispositivo ha alterado la memoria principal, entonces la palabra de caché no es válida.

  Por último, si varios procesadores se conectan al mismo bus y cada uno de ellos tiene su propia caché local, si se modifica una palabra en una de las cachés, podría invalidar una palabra de otras cachés.

  Las políticas de escritura se dan:
    - **En acierto:**
        - **Escritura Inmediata (Write-through):** todas las operaciones de escritura se hacen tanto en la caché como en memoria principal, asegurando que el contenido de la memoria principal siempre es válido. Cualquier otro procesador puede monitorizar el trafico a memoria principal para mantener la coherencia de su propia caché. La principal desventaja es que genera un trafico sustancial con la memoria.
        - **Post-escritura (Write-back):** las actualizaciones se hacen solo en la caché, lo que minimiza las escrituras en memoria. Cuando se tiene una actualización, se activa un bit ACTUALIZAR asociado a la línea. Cuando el bloque es sustituido, es escrito en memoria principal si y sólo si el bit está activo. El problema es que se tienen

porciones de memoria principal que no son válidas, y los accesos por parte de módulos de E/S tienen que hacerse a través de la caché. Esto complica la circuitería y genera un cuello de botella potencial.

- **En fallo:**
  - **Write Allocate:** primero se asigna un bloque de memoria caché para esa ubicación y, a continuación, se escriben los datos en la memoria caché. De esta manera, se evita escribir directamente en la memoria principal, que es más lenta que la caché. Sin embargo, esto también significa que tiene que desalojar otro bloque de caché para dejar espacio para el nuevo, lo que puede causar otra pérdida de caché más adelante.
  - **No-write Allocate:** no se asigna un bloque de caché para esa ubicación y, en su lugar, se escriben los datos directamente en la memoria principal. De esta manera, se evita desalojar un bloque de caché que puede ser útil más adelante. Sin embargo, esto también significa que tienes que acceder a la memoria principal cada vez que escribes en esa ubicación, que es más lenta que la caché.

5. **Describa algoritmos de ubicación y política de escritura en caché.**

Política de ubicación: Ya que hay menos líneas de caché que bloques de memoria principal, se necesita un algoritmo que haga corresponder bloques de memoria principal a líneas de caché. Hay tres tipos de correspondencia:

- **Correspondencia Directa:** se corresponde cada bloque de memoria a solo una línea posible de caché. La correspondencia se expresa como:

  Nro. línea caché = *Nro. bloque Mem.Princ.* **mod** *Nro. líneas en caché*

  Esta técnica es sencilla y poco costosa de implementar. Su principal desventaja es que hay una posición concreta de caché para cada bloque dado; si un programa referencia repetidas veces a palabras de dos bloques diferentes asignados a la misma línea, dichos bloques estarían intercambiando continuamente en la caché, y la tasa de acierto sería baja.

- **Correspondencia Asociativa:** cada bloque de memoria puede cargarse en cualquier línea de la caché. Así, hay flexibilidad para que cualquier bloque sea reemplazado cuando se va a escribir uno nuevo en la caché. La principal desventaja es la compleja circuitería necesaria para examinar en paralelo las etiquetas de todas las líneas de caché cuando se quiere buscar algún elemento.
- **Correspondencia Asociativa por Conjuntos:** la caché se divide en conjuntos, cada bloque de memoria puede cargarse en cualquier línea del conjunto que le corresponde. Las relaciones son:

  Nro. conjunto = *Nro. bloque ref.* **mod** *Nro. conjuntos caché*

  Con la correspondencia asociativa por conjuntos, la etiqueta de una dirección de memoria es más corta que con la correspondencia totalmente asociativa, y el número de comparaciones que se debe hacer en un conjunto es mucho menor.

  El uso de dos líneas por conjunto es el caso más común, mejorando significativamente la tasa de aciertos respecto de la correspondencia directa.

Política de escritura: Hay dos casos a considerar cuando se tiene que reemplazar un bloque de caché, si el bloque fue modificado o no. Si no fue modificado, no hay problema en sobrescribirlo con un nuevo bloque. Si fue modificado, entonces la memoria principal debe actualizarse.

Además, más de un dispositivo puede tener acceso a la memoria principal. Si una palabra ha sido modificada solo en la caché, la correspondiente de memoria no es válida para ser leída por otros dispositivos. Si un dispositivo ha alterado la memoria principal, entonces la palabra de caché no es válida.

Por último, si varios procesadores se conectan al mismo bus y cada uno de ellos tiene su propia caché local, si se modifica una palabra en una de las cachés, podría invalidar una palabra de otras cachés.

Las políticas de escritura se dan:

- **En acierto:**
    - **Escritura Inmediata (Write-through):** todas las operaciones de escritura se hacen tanto en la caché como en memoria principal, asegurando que el contenido de la memoria principal siempre es válido. Cualquier otro procesador puede monitorizar el tráfico a memoria principal para mantener la coherencia de su propia caché. La principal desventaja es que genera un tráfico sustancial con la memoria.
    - **Post-escritura (Write-back):** las actualizaciones se hacen solo en la caché, lo que minimiza las escrituras en memoria. Cuando se tiene una actualización, se activa un bit ACTUALIZAR asociado a la línea. Cuando el bloque es sustituido, es escrito en memoria principal si y sólo si el bit está activo. El problema es que se tienen porciones de memoria principal que no son válidas, y los accesos por parte de módulos de E/S tienen que hacerse a través de la caché. Esto complica la circuitería y genera un cuello de botella potencial.
- **En fallo:**
    - **Write Allocate:** primero se asigna un bloque de memoria caché para esa ubicación y, a continuación, se escriben los datos en la memoria caché. De esta manera, se evita escribir directamente en la memoria principal, que es más lenta que la caché. Sin embargo, esto también significa que tiene que desalojar otro bloque de caché para dejar espacio para el nuevo, lo que puede causar otra pérdida de caché más adelante.
    - **No-write Allocate:** no se asigna un bloque de caché para esa ubicación y, en su lugar, se escriben los datos directamente en la memoria principal. De esta manera, se evita desalojar un bloque de caché que puede ser útil más adelante. Sin embargo, esto también significa que tienes que acceder a la memoria principal cada vez que escribes en esa ubicación, que es más lenta que la caché.

6. **Analice ventajas y desventajas de poseer varios niveles de caché.**

Las ventajas de tener varios niveles de cache pueden ser reducción del tiempo de datos, mayor capacidad de almacenamiento y reducción de la latencia de memoria principal Las desventajas que puede tener son: mayor complejidad del sistema, mayor costo de espacio físico y energía, y problemas de coherencia de cache.

7. **¿Qué características definen un procesador como superescalar? Describa las políticas de emisión de instrucciones en un cauce segmentado.**

Un procesador superescalar es que en el que las instrucciones comunes pueden iniciar su ejecución simultáneamente y ejecutarse de manera independiente. Lo esencial del enfoque es su habilidad para ejecutar instrucciones en diferentes cauces de manera independiente y concurrente. Hay múltiples unidades funcionales, cada una de las cuales esta implementada como un cauce segmentado, que admiten la ejecución en paralelo de varias instrucciones.

El termino emisión de instrucciones se usa para referirse al proceso de iniciar la ejecución de instrucciones en las unidades funcionales del procesador y el termino política de emisión de instrucciones para referirse al protocolo usado para emitir instrucciones. Se puede decir que la emisión de una instrucción tiene lugar cuando esta pasa de la etapa de decodificación del cauce a la primera etapa de ejecución.

Las políticas de emisión de instrucciones se pueden agrupar en tres categorías:

- **Emisión en orden y finalización en orden:** se emiten las instrucciones en el orden exacto en que lo haría una ejecución secuencial y se escriben los resultados en el mismo orden. Para garantizar la finalización en orden, cuando hay una pugna por una unidad funcional o cunado una unidad funcional necesita mas de un ciclo para generar un resultado, la emisión de instrucciones se detiene temporalmente.
- **Emisión en orden y finalización desordenada:** puede haber cualquier numero de instrucciones en la etapa de ejecución en un momento dado, hasta alcanzar el máximo grado de paralelismo en

la maquina ocupando las unidades funcionales. La emisión de instrucciones se para cuando hay una pugna por un recurso, una dependencia de datos o una dependencia de control.
Es más difícil ocuparse de las interrupciones y excepciones. Cuando ocurre una interrupción, la ejecución de instrucciones se suspende en el punto actual, para reanudarse posteriormente. El procesador debe asegurarse que la reanudación tiene en cuenta que, en el momento de interrupción, algunas instrucciones posteriores a la instrucción que provocó dicha interrupción pueden haber finalizado ya.

- **Emisión desordenada y finalización desordenada:** para permitir la emisión desordenada, es necesario desacoplar las etapas del cauce de decodificación y ejecución. Esto se hace mediante un buffer llamado ventana de instrucciones. Cuando el procesador termina de decodificar una instrucción, la coloca en la ventana de instrucciones. Mientras el buffer no se llene, el procesador puede continuar captando y decodificando nuevas instrucciones. Cunado una unidad funcional de la etapa de ejecución queda disponible, se puede emitir una instrucción desde la ventana a la etapa de ejecución. Cualquier instrucción puede emitirse, siempre que (1) necesite la unidad funcional particular que esté disponible y (2) ningún conflicto ni dependencia la bloqueen.
  Así, las instrucciones se emiten desde la ventana de instrucciones sin que se tenga en cuenta su orden original en el programa. Su única restricción es que el programa funcione correctamente.

8. **¿Cuáles son las arquitecturas que pueden encontrarse en la configuración MIMD de la taxonomía de Flynn?**

La configuración MIMD (Múltiples secuencias de Instrucciones y Múltiples secuencias de Datos) es en el que un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes. En un computador MIMD hay múltiples unidades de control, y cada una proporciona una secuencia de instrucciones separada a su propio elemento de proceso. El MIMD puede ser un multiprocesador de memoria compartida o un multiprocesador de memoria distribuida.

Los multiprocesadores de memoria compartida se pueden dividir en tres categorías. En las tres, todos los procesadores tienen acceso a todas las partes de memoria principal usando instrucciones de carga y almacenamiento:

- **Acceso uniforme a memoria (UMA, *Uniform Memory Acess*):** el tiempo de acceso de un procesador a cualquiera otra región de la memoria es el mismo. El tiempo de acceso a memoria por parte de todos los procesadores es el mismo.
- **Acceso no uniforme a memoria (NUMA, *Nonuniform Memory Acess*):** el tiempo de acceso a memoria de un procesador depende de la región a la que se acceda. Para procesadores distintos, las regiones de memoria que son más lentas o más rápidas son diferentes.
- **NUMA con coherencia de caché (CC-NUMA, *Cache-Coherent NUMA*):** un computador NUMA en el que la coherencia de caché se mantiene en todas las cachés de los distintos procesadores.

9. **¿Qué características posee un procesador supersegmentado frente a un superescalar?**

La supersegmentación aprovecha el hecho de que muchas etapas del cauce realizan tareas que utilizan menos de medio ciclo de reloj. De este modo, doblando la velocidad de reloj interna se permite la realización de dos tareas en un ciclo de reloj externo.

Una implementación superescalar de un procesador es que en el que las instrucciones comunes pueden iniciar su ejecución simultáneamente y ejecutarse de manera independiente.

10. **Describa las características que diferencian los SMP respecto a los Clusters.**

Las diferencias entre SMP y Clusters son:

- Compartición de recursos: los SMP comparten recursos como la memoria, los buses de datos y los dispositivos de E/S entre todos los procesadores, los clusteres tienen recursos separados y se conectan mediante una red de interconexión.
- Escalabilidad: los SMP no escalan bien más allá de un número limitado de procesadores, mientras los clusteres de computadora se pueden escalar fácilmente agregando más nodos en la red.
- Latencia de red: en los SMP los procesadores pueden acceder a la memoria compartida de manera muy eficiente, lo que reduce la latencia de acceso, los clusteres deben comunicarse a través de la red, lo que puede generar mayor latencia.
- Costo: los SMP pueden ser más costosos debido a que requieren una gran cantidad de hardware compartido.
- Tolerancia a fallas: los clusteres de computadora tienen mayor tolerancia a fallas que los sistemas SMP, ya que la pérdida de un nodo no afectara al funcionamiento de los demás.

11. Describa las limitaciones existentes al paralelismo a nivel instrucciones.

La expresión paralelismo de las instrucciones se refiere al grado en el que, en promedio, las instrucciones de un programa se pueden ejecutar en paralelo. Las limitaciones del paralelismo en las instrucciones son:

- **Dependencia de datos verdadera:** una instrucción genera un dato que lee otra posterior. Si no hay dependencia, se puede captar y ejecutar dos instrucciones en paralelo. En caso de que exista dependencia de datos entre la primera y la segunda instrucción, se retrasa la segunda instrucción tantos ciclos de reloj como sea necesario para eliminar la dependencia.
- **Dependencia de control, o relativa al procedimiento:** ocurre cuando la ejecución de una instrucción depende de cómo se ejecute otra. Las instrucciones que siguen a una bifurcación tienen una dependencia relativa al procedimiento en esa bifurcación y no pueden ejecutarse hasta que se ejecute el salto.
- **Conflicto en los recursos:** un conflicto en los recursos es una pugna de dos o mas instrucciones por el mismo recurso al mismo tiempo. Los conflictos en los recursos pueden superarse duplicando estos. Además, cuando una operación tarda mucho tiempo en finalizar, los conflictos en los recursos se pueden minimizar segmentando la unidad funcional apropiada.
- **Dependencia de salida:** una instrucción escribe un dato después que otra posterior. Solo se da si se deja que las instrucciones adelanten unas a otras (finalización desordenada). La segunda instrucción debe finalizar después de la primera para producir el valor correcto de salida. Para asegurar esto, la emisión de la segunda debe detenerse si su resultado puede ser sobrescrito más tarde por una instrucción anterior que tarda mas en finalizar.
- **Antidependencia:** una instrucción modifica un valor antes que otra anterior que lo tiene que leer, lo lea. Se da por la emisión desordenada. Es similar a la dependencia verdadera, pero a la inversa: en lugar de que la primera instrucción produzca un valor que usa la segunda instrucción, la segunda instrucción destruye un valor que usa la primera instrucción.

12. ¿Cómo se segmenta el cauce de instrucciones en el MIPS 64?

Las 5 etapas de la segmentación son:

- Búsqueda (IF): Se accede a memoria por la instrucción. Se incrementa el PC.
- Decodificación/búsqueda de operandos (ID): Se decodifica la instrucción. Se accede al banco de registros por los operandos. Se calcula el valor del operando inmediato. Si es un salto, se calcula el destino y si se toma o no.
- Ejecución/Dirección efectiva (EX): Si es una instrucción de procesamiento, se ejecuta en la ALU. Si es un acceso a memoria, se calcula la dirección efectiva. Si es un salto, se almacena el nuevo PC.
- Acceso a memoria/terminación del salto (MEM): Si es un acceso a memoria, se accede.
- Almacenamiento (WEB): Se almacena el resultado en el banco de registros.

13. ¿Cómo funciona el Acceso Directo a Memoria? ¿Qué es un robo de ciclo?

El DMA es una técnica de gestión de E/S en el cual el módulo de E/S y la memoria principal intercambian datos directamente, sin la intervención del procesador. Esta técnica requiere un módulo adicional en el bus del sistema. El controlador de DMA es capaz de imitar al procesador y de recibir el control del bus del sistema cedido por el procesador. Necesita dicho control para transferir datos a, y desde memoria, a través del bus del sistema.

El DMAC debe actuar como maestro del bus durante la transferencia DMA y debe ser capaz de:

- Solicitar el uso del bus mediante las señales y la lógica de arbitraje necesarias.
- Especificar la dirección de memoria sobre la que se deba realizar la transferencia.
- Generar las señales de control del bus.

Cuando el procesador desea leer o escribir un bloque de datos, envía una orden al modulo de DMA, incluyendo la siguiente información:

- Si se solicita una lectura o escritura, utilizando la línea de control de lectura o escritura entre el procesador y el DMAC.
- La dirección del dispositivo de E/S en cuestión, indicada a través del bus de datos.
- La posición inicial de memoria a partir de donde se lee o escribe, indicada a través del bus de datos y almacenada por el modulo de DMA en su registro de direcciones.
- El numero de palabras a leer o escribir, también indicado a través de las líneas de datos y almacenado en el registro de cuenta de datos.

Después, el procesador continúa con otro trabajo.

Cuando el periférico está listo para realizar la transferencia se lo indica al DMAC. El DMAC pide el control del bus y realiza la transferencia entre el periférico y la memoria. Después de cada palabra transferida, se actualizan los registros del DMAC (número de palabras a transferir y dirección de memoria).

Una vez terminada la transferencia, el DMAC libera el bus y le envía una señal de interrupción al procesador. Así, el procesador interviene al comienzo y al final de la transferencia.

Un problema de esta técnica es que la CPU no puede acceder a memoria para leer instrucciones o datos mientras el bus está ocupado con una transferencia DMA. Esto puede degradar el rendimiento de la CPU.

La solución es el tipo de transferencia robo de ciclo: el DMAC solicita el control del bus a la CPU. Cuando la CPU concede el bus al DMAC, se realiza la transferencia de una sola palabra y después el DMAC libera el bus. El DMAC solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo. Así, no se degrada el rendimiento del sistema, pero la transferencia tarda más tiempo en llevarse a cabo.

Para la CPU esto no es una interrupción, así que no debe guardar el contexto. Si bien el trabajo de la CPU es lento, no será tanto como si ella realizara la transferencia.

14. Describir un ciclo de instrucción.

El ciclo de instrucción, en su forma más simple, puede describirse con dos etapas: el ciclo de captación (el procesador capta la instrucción de memoria) y el ciclo de ejecución (el procesador ejecuta la instrucción). La ejecución de un programa consiste en la repetición del proceso de captación de instrucción y ejecución de un programa. La ejecución se para solo si la maquina se desconecta, se produce algún tipo de error irrecuperable o ejecuta una instrucción del programa que detiene al computador.

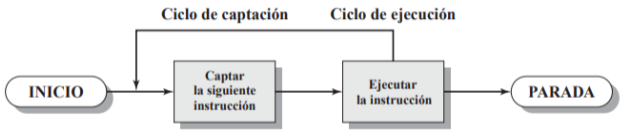


Figura 3.3. Ciclo de instrucción básico.

En el ciclo de captación, la CPU capta una instrucción de memoria. La CPU utiliza un registro llamado contador de programa (PC) para seguir la pista de la instrucción que debe captarse a continuación. A no ser que se indique otra cosa, la CPU siempre incrementa el PC después de captar una instrucción, de forma que captará la siguiente instrucción de la secuencia. La instrucción captada se almacena en un registro de la CPU conocido como registro de instrucción (IR). La CPU interpreta la instrucción y lleva a cabo la acción requerida.

En el ciclo de ejecución, la acción realizada depende del tipo de instrucción captada:

- **Procesador-Memoria:** deben transferirse datos desde la CPU a memoria, o viceversa.
- **Procesador-E/S:** deben transferirse datos a o dese el exterior mediante transferencias entre la CPU y un modulo de E/S.
- **Procesamiento de Datos:** la CPU ha de realizar una operación aritmética o lógica con los datos.
- **Control:** una instrucción puede especificar que la secuencia de ejecución se altere.

Una visión mas detallada del ciclo de instrucción es la de un diagrama de estados. Para un ciclo de instrucción dado, algunos estados pueden no darse y otros pueden visitarse más de una vez. La máquina siempre está en algunos de estos estados si hay un programa en ejecución:

- **Cálculo de la dirección de la instrucción (IAC, Instruction Adress Calculation):** determina la dirección de la siguiente instrucción a ejecutar. Normalmente, esto implica añadir un número fijo a la dirección de la instrucción previa.
- **Captación de instrucción (IF, Instruction Fetch):** la CPU lee la instrucción desde su posición de memoria.
- **Decodificación de la operación indicada en la instrucción (IOD, Instruction Operation Decoding):** analiza la instrucción para determinar el tipo de operación a realizar y el (los) operando(s) a utilizar.
- **Cálculo de la dirección del operando (OAC, Operand Adress Calculation):** si la instrucción implica una referencia a un operando en memoria o disponible mediante E/S, determina la dirección del operando.
- **Captación del operando (OF, Operand Fetch):** capta el operando desde memoria o se lee desde el dispositivo de E/S.
- **Operación con los datos (DO, Data Operation):** realiza la operación indicada en la instrucción.
- **Almacenamiento de operando (OS, Operand Store):** escribe el resultado en memoria o lo saca a través de un dispositivo de E/S.

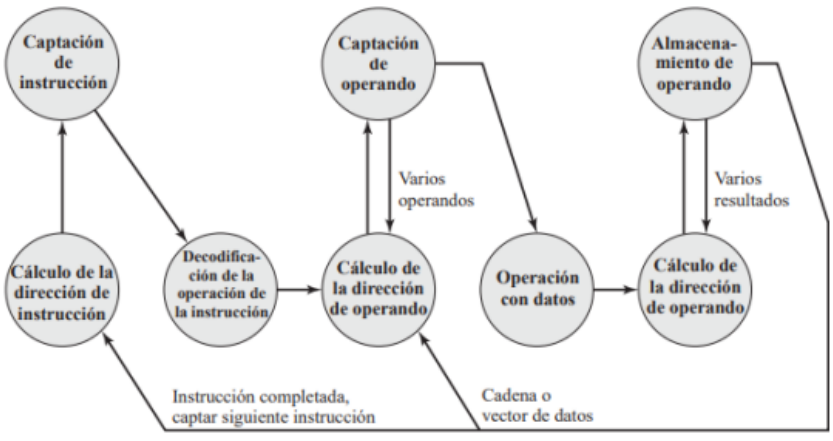


Figura 3.6. Diagrama de estados del ciclo de intrucción.

Si se habilitan las interrupciones, se agrega al ciclo de instrucción un tercer ciclo, el ciclo de interrupción. En este, el procesador comprueba si se ha generado alguna interrupción, indicada por la presencia de una flag de interrupción. Si no hay señales de interrupción presentes, el procesador continúa con el ciclo de captación y accede a la siguiente instrucción del programa en uso. Si hay alguna interrupción pendiente, se gestiona:

- El procesador suspende la ejecución del programa en curso y guarda su contexto (flags, PC, etc).
- Carga el contador de programa con la dirección de comienzo de una gestión de interrupción.

A continuación, el procesador prosigue con el ciclo de captación y accede a la primera instrucción del programa de interrupciones, que dará servicio a la interrupción. Cuando la rutina de gestión termina, el procesador prosigue con la ejecución del programa de usuario en el punto que se interrumpió.

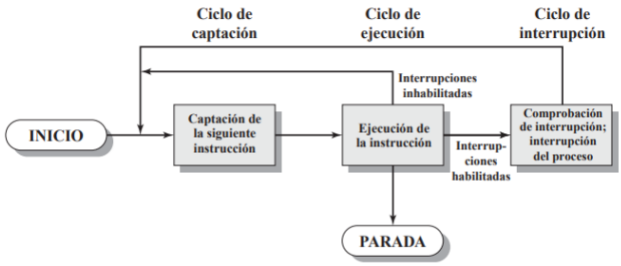


Figura 3.9. Ciclo de instrucción con interrupciones.

Diagrama de estados de un ciclo de instrucción con interrupciones

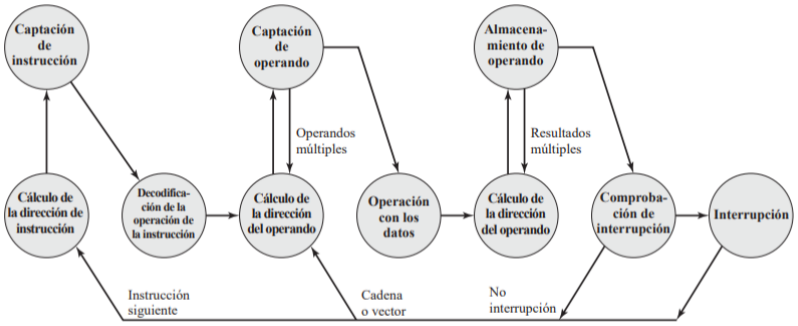


Figura 3.12. Diagrama de estados de un ciclo de instrucción, con interrupciones.

15. Describir los tratamientos de salto.

Los riesgos de control se dan porque hasta que la instrucción de salto no se ejecute realmente, es imposible determinar si el salto se producirá o no. Llamamos penalización por salto a los ciclos perdidos esperando que se resuelva el salto.

Con saltos incondicionales, la dirección del destino de salto se debe determinar lo antes posible para reducir la penalización. Como la dirección del destino se puede obtener en la etapa X del cauce, solo se pierde un ciclo.

En los saltos condicionales, en cambio, hay que resolver la condición de salto, por lo que se pierden un mínimo de 3 ciclos. Lo optimo entonces, es resolver la condición lo antes posible, para solo perder 1 ciclo de reloj (lo que pierden los saltos incondicionales). Para esto, las condiciones de los saltos se deben resolver en la etapa D, en donde:

- Se decodifica y se sabe que la instrucción es un salto (lo que ya se hacía).
- Se puede evaluar la condición de salto, es decir, comparar los valores de los registros. Esto se puede realizar añadiendo un restador a la etapa.
- Se puede calcular la dirección del salto, es decir, si se debe seguir secuencialmente o saltar. En este caso, se necesita un sumador para sumar el valor del PC con lo correspondiente.

Si queremos reducir todavía mas la penalización, podemos utilizar tratamientos de saltos. Hay de dos tipos:

- **Técnica Hardware:** predicción de saltos para evitar la parada.
- **Técnica Software:** salto retardado o relleno de ranura de retardo.

En las predicciones de salto, puede haber:

- **Técnicas estáticas:** no dependen de la historia de la ejecución que haya tenido lugar antes del salto condicional.
    - **Predecir que nunca se salta:** supone que el salto no se producirá y continuará captando instrucciones secuencialmente.
    - **Predecir que siempre se salta:** supone que el salto se producirá y siempre captará la instrucción destino del salto.

  Si la predicción falla, se fue a buscar la instrucción errónea. Esta se descarta y se capta la instrucción correcta. Solo se pierde el ciclo de penalización por salto, por lo que, en el peor de los casos, esperamos lo mismo que si no se predice.

- **Técnicas dinámicas:** dependen de la historia de la ejecución.
    - **Conmutador saltar/no saltar:** a cada instrucción de salto pueden asociarse uno o mas bits que reflejen su historia reciente. Estos bits son consultados a modo de conmutador saltar/no saltar que dirige al procesador a tomar una determinada decisión la próxima vez que encuentre la instrucción.
    - **Tabla de historial de saltos (BTB):** es una pequeña memoria caché asociada a la etapa de captación de instrucción del cauce (F). Cada elemento de la tabla tiene tres campos: la dirección de una instrucción de salto, un determinado numero de bits de historia que guardan el estado de esa instrucción e información sobre la instrucción destino.

El salto retardado se basa en realizar trabajo útil mientras el salto se resuelve. Una ranura de retardo del salto es el periodo de penalización luego de una instrucción de salto. El compilador trata de situar instrucciones útiles (que no dependan del salto) en los huecos de retardo. Esto requiere una reordenación de las instrucciones.

16. Nombrar los modos de direccionamiento.

Los modos de direccionamiento son la forma en el que programador le describe a la máquina el lugar donde se encuentran los operandos o la siguiente instrucción a ejecutar. Los más comunes son:

- **Inmediato:** el operando esta presente en la propia instrucción. Ej: MOV AX, 10
- **Directo:** también llamado absoluto. El campo de direcciones tiene la dirección efectiva del operando. Solo requiere una referencia a memoria y no necesita ningún calculo especial. Ej: MOV AX, 1000H
- **Indirecto:** el campo de direcciones referencia la dirección de una palabra a memoria, la cual contenga la dirección completa del operando.
- **Directo de Registro:** el campo de direcciones referencia un registro, en donde se encuentra el operando.
- **Indirecto de Registro:** el campo de direcciones referencia un registro, la cual tiene la dirección de memoria del operando.
- **Con Desplazamiento:** las instrucciones tienen dos campos de direcciones. Uno de los campos tiene una referencia a un registro cuyo contenido se le suma al valor contenido en el otro campo para generar la dirección efectiva.

17. Describir el proceso de E/S mediante interrupciones. ¿Qué ventaja presenta ante la E/S programada?

Con E/S mediante interrupciones, el procesador proporciona la orden de E/S a un módulo, y continúa realizando algún trabajo útil. El módulo de E/S trabaja con el periférico para realizar la tarea e interrumpirá al procesador para solicitar su servicio cuando esté preparado para intercambiar datos con él. El procesador entonces ejecuta la transferencia de datos, como antes, y después continúa el procesamiento previo.

Desde el punto de vista del procesador, las acciones son las siguientes:

- El procesador envía una orden al módulo de E/S. Pasa a realizar otro trabajo.
- Al final de cada ciclo de instrucción, el procesador comprueba las interrupciones.
- Cuando se pide la interrupción desde el módulo de E/S, el procesador guarda el contexto del programa en curso y procesa la interrupción.
- Luego de procesar la interrupción, recupera el contexto del programa que estaba ejecutando y continúa su ejecución.

Con la E/S programada, el procesador debe esperar a que el módulo de E/S y el periférico terminen la tarea, y debe comprobar repetidamente el estado del módulo de E/S. Esto hace que se degrade el nivel de prestaciones de todo el sistema. Con la E/S mediante interrupciones, el procesador no debe quedarse esperando, y puede seguir trabajando en otra cosa; simplemente debe atender la interrupción cuando esta llegue.

18. ¿Cuáles son los tres tipos de periféricos?

Un dispositivo externo se conecta al computador mediante un enlace a un módulo de E/S. El enlace se utiliza para intercambiar señales de control, estado, y datos entre el módulo de E/S y el dispositivo externo. Un dispositivo externo conectado a un módulo de E/S frecuentemente se denomina periférico. En sentido amplio, los dispositivos externos se pueden clasificar en tres categorías:

- De *interacción con humanos*: permiten la comunicación con el usuario del computador.
- De *interacción con máquinas*: permiten la comunicación con elementos del equipo.
- De *comunicación*: permiten la comunicación con dispositivos remotos.

19. ¿Cuántos tipos de registros tiene la CPU? ¿Qué registros componen cada tipo?

La CPU tiene dos tipos de registro:

- Los visibles para el usuario: compuestos por registros de uso general, datos, direcciones y códigos de condición.
- Los de control y estado: compuestos por:
    - **PC:** el *Program Counter*, indica la posición donde está el procesador en su secuencia de instrucciones.
    - **IR:** el *Instruction Register*, contiene la instrucción que se está ejecutando.
    - **MAR:** el *Memory Adress Register*, especifica la dirección en memoria de la próxima lectura o escritura.
    - **MBR:** el *Memory Buffer Register*, contiene el dato que se va a escribir en memoria o donde se escribe el dato que se va a leer de memoria.

20. Nombrar y describir las clases de interrupciones.

Las clases de interrupciones más comunes son:

- **Programa:** generadas por alguna condición que se produce como resultado de la ejecución de una instrucción, tal como el overflow, división por cero, intento de ejecutar una instrucción inexistente e intento de acceder fuera del espacio de memoria permitido para el usuario.
- **Temporización:** generadas por un temporizador interno al procesador. Esto permite al sistema operativo realizar ciertas funciones de manera regular.
- **E/S:** generadas por un controlador de E/S, para indicar la finalización sin problemas de una operación o para avisar de ciertas condiciones de error.
- **Fallo de hardware:** generadas por un fallo tal como la falta de potencia de alimentación o un error de paridad en la memoria.

21. **Describir el proceso de una llamada a procedimiento.**

Una subrutina es un programa con entidad propia que se incorpora en un programa más grande. En cualquier punto del programa se puede invocar o *llamar* al procedimiento.

En la llamada a un procedimiento es necesario pasar o transferir ciertos argumentos. Esto se puede hacer:

- Por **valor**: se pasa una copia del dato.
- Por **referencia**: se pasa la dirección de memoria del dato.

Además, también es necesario el lugar en donde se encuentran estos datos:

- Por **registro**: se copia el valor del dato o su dirección en un registro de la CPU. La principal limitación es el número de registros disponibles.
- Por **memoria**: se define un área de memoria especifica para el pasaje de argumentos. Esto hace difícil transferir un número variable de argumentos.
- Por **pila:** se apilan el valor del dato o su dirección en la pila antes de la llamada.

Cuando se realiza la llamada al procedimiento desde un programa (puede ser el programa principal u otra subrutina), se utiliza una instrucción que realiza la bifurcación o salto al procedimiento. Al momento de la llamada, la CPU debe guardar el contexto del programa que estaba ejecutando, es decir, los flags de estado y el valor del PC para poder retornar al punto en el que estaba cuando vuelva de la subrutina. Estos valores se guardan en la pila (apilan) junto con la llamada al procedimiento. Además, se actualiza el PC con el valor de la primera instrucción del procedimiento.

Luego, la CPU ejecuta las instrucciones del procedimiento. La última instrucción va a ser una instrucción de retorno, la cual "saca" el primer valor de la pila (desapila), el cual debería ser el estado del programa antes de haber saltado a la subrutina. Así, se actualiza el PC con la próxima instrucción (la siguiente instrucción luego del llamado a la subrutina) y se sigue la ejecución del programa anterior.

22. **Explicar los diferentes tipos de operaciones de instrucción.**

Una clasificación típica es la siguiente:

- **Transferencia de datos:** debe especificarse las ubicaciones del operando fuente y del operando destino, el tamaño de los datos a ser transferidos y el método de direccionamiento. Para diferentes movimientos puede haber diferentes instrucciones o una sola instrucción y diferentes direcciones.
- **Aritméticas:** proporcionan las operaciones básicas de suma, resta, multiplicación y división.
- **Lógicas:** operaciones que manipulan bits individualmente.
- **De Conversión:** operaciones para cambiar formatos de datos.
- **De E/S:** pocas instrucciones, pero de especificaciones específicas. Se pueden realizar utilizando instrucciones de movimiento de datos o a través de un controlador aparte (DMA).
- **De Control de Flujo:** modifican el valor contenido en el registro PC. Pueden ser de:
    - **Bifurcación:** también llamada de salto, tiene como uno de sus operandos la dirección de la siguiente instrucción a ejecutar. Las más frecuentes son las de salto condicional, es decir, solo se efectúa la bifurcación si se cumple una condición dada. Una instrucción de salto en que se produce siempre la bifurcación es un salto incondicional.
    - **De llamada a procedimiento:** requiere de dos instrucciones básicas: una de llamada (CALL), que produce una bifurcación desde la posición actual al procedimiento; y una instrucción de retorno del procedimiento (RET) al lugar dese el que se llamó.

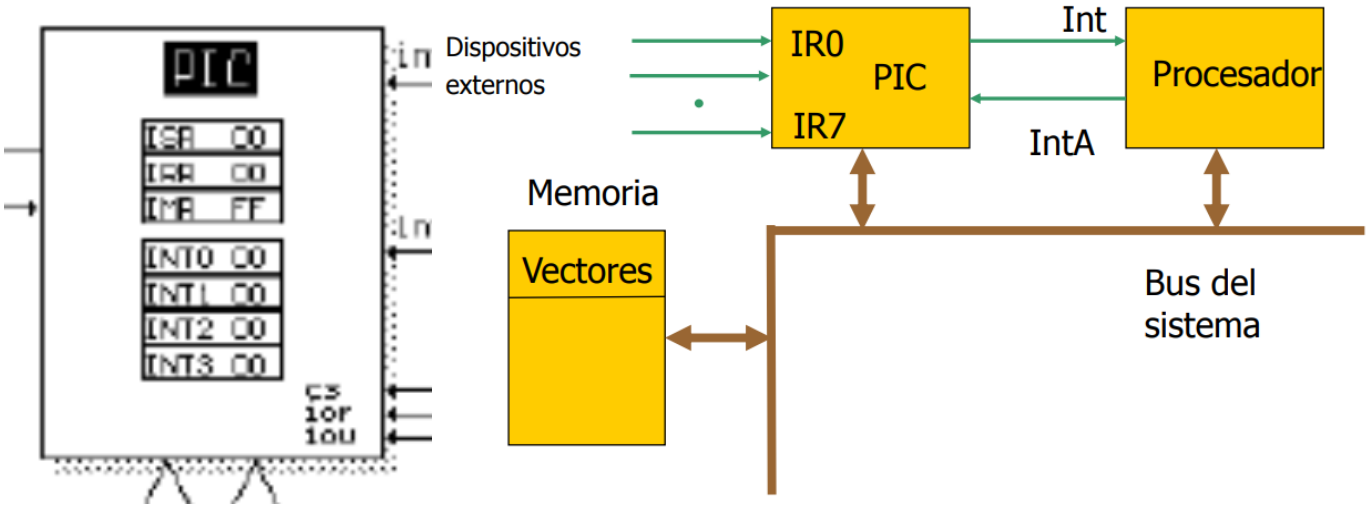
23. **¿Cuáles son las diferencias en la terminación de una subrutina y un gestor de interrupción?**

Las subrutinas son programas con entidad propia que se incorporan en un programa mas grande. En cualquier punto del programa se pueden invocar o *llamar*. Una subrutina termina con una instrucción de retorno (en el MSX88, por ejemplo, es la instrucción RET), la cual, en la mayoría de los casos, desapila el estado del programa (el cual se apila al momento de la llamada).

Un gestor de interrupción es una subrutina de servicio de interrupción. Una interrupción, sea por hardware o por software, es atendida por la CPU mediante su gestor. Este no forma parte del programa, sino que suele ser parte del sistema operativo. El gestor termina con una instrucción especifica de terminación de la interrupción (en el MSX88, por ejemplo, es la instrucción IRET), que desapila el estado del programa (que fue apilado al momento de la llamada). Luego, se vuelve a la ejecución normal del programa de usuario.

En resumen, las subrutinas son parte del flujo de ejecución normal del programa y se llaman explícitamente cuando se necesita su funcionalidad. Por otro lado, los gestores de interrupción son rutinas especiales que se activan en respuesta a eventos externos y se encargan de manejar esos eventos sin intervención directa del programa principal.

24. Esquematizar y describir la estructura de un Controlador Programable de Interrupciones.



El Controlador Programable de Instrucciones (PIC) tiene una sola responsabilidad, la gestión de interrupciones. El PIC acepta las solicitudes de interrupción de los dispositivos conectados a él, determina que interrupciones tiene la prioridad mas alta, y se lo indica entonces al procesador activando la señal INTR. El procesador reconoce la solicitud mediante la línea INTA. Esto hace que el PIC sitúe el vector apropiado en el bus de datos, para que el procesador pueda iniciar el procesamiento de la interrupción.

El PIC tiene 2 líneas para comunicarse con la CPU:

- INTR: por donde le manda el pedido de interrupción.
- INTA: por donde la CPU le avisa que esta disponible para atender el pedido.

Tiene también registros internos, que cumplen funciones de control y administración de la información:

- EOI (20H): para comandos. Cuando la CPU termina la subrutina de gestión de interrupción, escribe el valor 20H en el EOI para que termine la interrupción.
- IMR (21H): marcara de interrupciones. Se pueden enmascaras dispositivos poniendo en ‘1’ el bit que le correspondería. El LSB corresponde a INT0.
- IRR (22H): petición de interrupción. Si el dispositivo no está enmascarado, pone en ‘1’ su bit correspondiente al pedir una interrupción. Si hay algún bit en ‘1’, el PIC le avisa a la CPU que hay un pedido de interrupción.
- ISR (23H): interrupción en servicio. Cuando la CPU responde que puede atender la interrupción, se pone en ‘1’ el bit correspondiente al dispositivo del cual se está atendiendo la interrupción.
- INT0…INT7 (24H..32H): almacenan el número de vector asociado con el dispositivo correspondiente. Esto hay que programarlo. El INT0 es el de mayor prioridad.

El PIC es programable. El procesador determina el esquema de prioridad que se va a utilizar cargando una palabra de control en el PIC.

25. **Describir cómo funciona la gestión de E/S programada con respuesta de espera.**

En la gestión de E/S programada con respuesta de espera, los datos se intercambian entre el procesador y el módulo de E/S. El procesador ejecuta un programa que controla directamente la operación de E/S, incluyendo la comprobación del estado del dispositivo, el envío de una orden de lectura o escritura y la transferencia del dato.

Cuando el procesador está ejecutando un programa y encuentra una instrucción relacionada con una E/S, ejecuta dicha instrucción mandando una orden al módulo de E/S apropiado. El módulo de E/S realizará la acción solicitada y después activará los bits apropiados de estado de E/S. El módulo no realiza ninguna otra acción para avisar al procesador, es decir, no interrumpe, se queda esperando. El procesador comprueba periódicamente el estado del modulo de E/S hasta que encuentra que la operación ha terminado.

26. **¿La coherencia de datos en un sistema jerárquico de memoria se ve afectado por el uso del DMA?**

En un sistema jerárquico de memoria, la coherencia de datos hace referencia al hecho de que las copias de la misma información en los distintos niveles deben contener los mismos valores.

El Acceso Directo a Memoria (DMA) es una técnica de E/S que permite que periféricos transmitan datos entre ellos y la memoria sin la intervención directa de la CPU. Cuando un dispositivo DMA realiza una transferencia de datos, la CPU no tiene control inmediato sobre esos datos, lo que puede dar lugar a problemas de coherencia. Además, mientras sucede la transferencia, la CPU está trabajando en otra cosa, por lo que puede estar modificando datos de la caché que no se verán reflejados en memoria (ya que no puede acceder al bus, el actual dueño es el DMAC).

En conclusión, la coherencia de datos sí se puede ver afectada por el uso del DMA.

27. **¿De qué depende el paralelismo de una maquina superescalar?**

El paralelismo de la maquina es una medida de la capacidad del procesador para sacar partido al paralelismo en las instrucciones. Depende del numero de instrucciones que pueden captarse y ejecutarse al mismo tiempo y de la velocidad y sofisticación de los mecanismos que usa el procesador para localizar instrucciones independientes.

Un programa puede no tener el suficiente nivel de paralelismo en las instrucciones como para sacar el máximo partido al paralelismo de máquina.

28. **¿Cuál es el objetivo de utilizar la técnica de renombre de registros en un procesador superescalar?**

El objetivo principal de utilizar la técnica de renombre de registros en un procesador superescalar es mejorar el paralelismo y la eficiencia de la ejecución de instrucciones.

En un procesador superescalar, se ejecutan múltiples instrucciones en paralelo mediante el uso de múltiples unidades de ejecución y un pipeline ancho. Sin embargo, para lograr este nivel de paralelismo, es necesario que las instrucciones sean independientes y no tengan dependencias entre sí.

Las dependencias de salida y antidependencias surgen porque los valores de los registros no pueden reflejar la secuencia de valores dictada por el flujo del programa. Valores entran en conflicto por el uso de registros, y el procesador debe resolver tales conflictos deteniendo ocasionalmente alguna etapa del cauce.

El renombramiento de registros es una técnica donde el hardware del procesador asgina dinámicamente los registros que están asociado con los valores que necesitan las instrucciones ne diversos instantes del tiempo. Cuando se crea un nuevo valor de registro (es decir, cuando se ejecuta una instrucción que tiene

un registro como operando destino) se asigna un nuevo registro para este valor. Las instrucciones posteriores que acceden con ese valor como operando fuente en ese registro tienen que sufrir un proceso de renombramiento: las referencias a registros de esas instrucciones han de revisarse para referenciar el registro que contiene el valor que se necesita. De este modo, las referencias a un mismo registro original en diferentes instrucciones pueden referirse a distintos registros reales, suponiendo diferentes valores.

Así, las dependencias en salida y antidependencias se eliminan y quedan sólo las dependencias verdaderas, mejorando el paralelismo y la eficiencia de la ejecución de instrucciones.

29. ¿Qué es la predicación de saltos en un procesador VLW?

Es una técnica de compilación (para generar código con alto grado de paralelismo). Se basa en la ejecución con predicados de la IA-64. Se deja que ambas ramas de un salto condicional se ejecuten en paralelo (para explotar todo el potencial de paralelismo). Se eliminan los saltos y reemplazan por ejecución condicional (para lo que se requiere soporte de hardware).

30. Esquematizar y describir un sistema NUMA con coherencia de cache

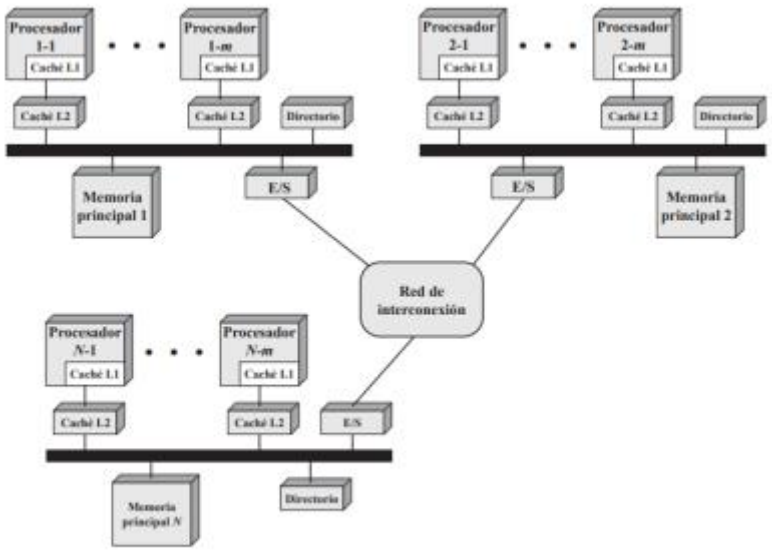


Figura 18.12. Organización CC-NUMA.

Un sistema NUMA es un sistema de Múltiples secuencias de Instrucciones y Múltiples secuencias de Datos (MIMD) de memoria compartida y acceso no uniforme a memoria.

Que el acceso a memoria sea no uniforme significa que todos los procesadores tienen acceso a todas las partes de memoria principal usando instrucciones de carga y almacenamiento, el tiempo de acceso a memoria de un procesador depende de la región a la que acceda y para procesadores distintos, las regiones de memoria que son más rápidas o más lentas son diferentes.

Un sistema CC-NUMA es un computador NUMA en el que la coherencia de caché se mantiene en todas las cachés de los distintos procesadores.

En el sistema hay varios nodos independientes, cada uno de los cuales es un SMP (multiprocesador simpetrico). Cada nodo contiene varios procesadores, cada uno con sus cachés L1 y L2, mas memoria principal. Los nodos se interconectan mediante un medio comunicación, que podría ser algún tipo de red.

Cada nodo del sistema incluye cierta cantidad de memoria principal, sin embargo, desde el punto de vista de los procesadores, existe un único espacio de memoria direccionable en el que a cada posición se asocia una única dirección válida para todo el sistema. Cuando un procesador inicia un acceso a memoria, si la posición solicitada no se encuentra en la caché del procesador, entonces la caché L2 inicia una operación de captación. Si la línea deseada está en una posición remota de la memoria principal, la línea se capta a través del bus local. Si la línea solicitada está en una porción remota de la memoria, entonces

se envía una petición automática para captar dicha línea a través de la red de interconexión, se proporcional a través del bus local a la caché que lo solicitaba en dicho bus. Toda esta actividad es automática y transparente al procesador y a su caché.

La coherencia de caché es una cuestión central. En términos generales podemos decir que cada nodo debe mantener algún tipo de orden de directorios que dé alguna indicación de la situación de varias partes de la memoria y también de la información de estado de la caché.

31. ¿Cuáles son los elementos a tener en cuenta para el diseño de un conjunto de instrucciones?

El repertorio de instrucciones define muchas de las funciones realizadas por el procesador, y tiene pues un efecto significativo sobre la implementación del mismo. El repertorio de instrucciones es el medio que tiene el programador para controlar el procesador. Los aspectos fundamentales de diseño son:

- El **repertorio de operaciones**: cuántas y qué operaciones considerar y cuán complejas deben ser.
- Los **tipos de datos**: los distintos tipos de datos con los que se efectúan operaciones.
- Los **formatos de instrucciones**: longitud de la instrucción (en bits), número de direcciones, tamaño de los distintos campos, etc.
- Los **registros**: número de registros del procesador que pueden ser referenciados por las instrucciones, y su uso.
- El **direccionamiento**: el modo o modos de direccionamiento mediante los cuales puede especificarse la dirección de un operando.

32. Describa las distintas maquinas que pueden cumplir instrucciones.

Algunas de las máquinas comunes que pueden cumplir instrucciones incluyen:

- **Unidad Aritmético-Lógica (ALU):** Esta máquina está diseñada para llevar a cabo operaciones aritméticas (suma, resta, multiplicación, división, etc.) y operaciones lógicas (AND, OR, NOT, etc.) sobre datos numéricos.
- **Unidad de Carga/Almacenamiento (Load/Store Unit):** Esta máquina se encarga de las operaciones de lectura (carga) y escritura (almacenamiento) de datos entre la memoria principal y los registros internos del procesador.
- **Unidad de Coma Flotante (FPU):** Esta máquina se especializa en realizar operaciones aritméticas en números de punto flotante, como sumas, restas, multiplicaciones y divisiones.
- **Unidad de Desplazamiento (Shift Unit):** Esta máquina realiza operaciones de desplazamiento de bits en los datos, como desplazamiento hacia la izquierda o hacia la derecha.
- **Unidad de Control de Flujo (Control Flow Unit):** Esta máquina se encarga de ejecutar las instrucciones de control de flujo, como saltos incondicionales (jump), saltos condicionales (branch), llamadas a subrutinas (call), retornos (return), etc.
- **Unidad de Punto Fijo (Fixed-Point Unit):** Esta máquina se especializa en operaciones de punto fijo en números enteros, como multiplicaciones y divisiones enteras.
- **Unidad de Predicción de Saltos (Branch Prediction Unit):** Esta máquina se encarga de predecir el resultado de saltos condicionales, lo que permite una ejecución más eficiente de las instrucciones de control de flujo.

33. Explicar el mecanismo de interrupción vectorizada

Cuando se utilizan interrupciones vectorizadas, el dispositivo que quiere interrumpir además de la señal de pedido de interrupción debe colocar en el bus de datos un identificador (vector). Este lo coloca el periférico directamente o un controlador de interrupciones que se ocupa de todo.

Este vector identifica el gestor de interrupciones que le corresponde a la interrupción pedida. El gestor de interrupciones es una subrutina de servicio de interrupción. Una interrupción, sea por hardware o por software, es atendida por la CPU mediante su gestor.

Los vectores de interrupción se ubican en una parte de memoria reservada, llamada tabla de vectores de interrupción. Cuando la CPU acepta el pedido de interrupción, el módulo que había pedido la interrupción envía su vector por el bus de datos. El elemento en el vector es la dirección del gestor de interrupción. Por lo que la CPU apila el estado del programa y el PC, y salta indirectamente vía memoria al gestor, utilizando el dato del vector. Una vez se termina la subrutina de gestión, la CPU desapila el estado del programa y el PC para seguir con el programa que estaba realizando antes de ser interrumpido.

34. ¿Porque 2 niveles de cache “normales” son mejores que uno solo “mejorado”?

Aquí hay algunas razones para preferir dos niveles de caché "normales" en lugar de uno solo "mejorado":

- **Jerarquía de almacenamiento:** La utilización de dos niveles de caché permite crear una jerarquía de almacenamiento más eficiente. Los niveles de caché más cercanos al procesador (L1) son más pequeños y rápidos, mientras que los niveles más alejados (L2) son más grandes y más lentos.
- **Latencia y ancho de banda:** Los dos niveles de caché pueden optimizarse individualmente para minimizar la latencia y aprovechar el ancho de banda disponible en el procesador. Un solo nivel "mejorado" podría tener limitaciones en términos de latencia o ancho de banda, lo que podría afectar negativamente el rendimiento general.
- **Acceso simultáneo:** Con dos niveles de caché, el procesador puede acceder simultáneamente a ambos niveles de caché, lo que permite una mayor capacidad de paralelismo y reducción de tiempos de espera. En un solo nivel "mejorado", si una instrucción requiere un acceso a la caché que está siendo utilizada por otra instrucción, puede generarse un cuello de botella.
- **Complejidad del diseño:** Implementar un solo nivel de caché "mejorado" puede requerir un diseño más complejo y costoso en términos de recursos de hardware. Por otro lado, dos niveles de caché "normales" pueden ser más simples de diseñar y mantener, lo que puede resultar en un mejor equilibrio entre rendimiento y costo.
- **Adaptabilidad:** La utilización de dos niveles de caché permite una mayor adaptabilidad a diferentes configuraciones de hardware y necesidades de rendimiento. Cada nivel de caché puede ajustarse individualmente para optimizar el rendimiento para una determinada aplicación o escenario de uso.

35. ¿Qué es y que mejora la técnica BTB?

La tabla de historia de saltos (BTB) es una técnica de predicción de saltos por hardware dinámica, es decir, toma en cuenta la historia de la ejecución para predecir si la instrucción que se debe captar luego del salto es la que sigue secuencialmente (no se salta) u otra (se salta).

La BTB una pequeña memoria caché asociada a la etapa de captación de instrucción del cauce. Cada elemento de la tabla tiene tres campos:

- La dirección de una instrucción de salto,
- Un determinado numero de bits de historia que guardan el estado de uso de esa instrucción,
- Información sobre la dirección destino.

Cuando se ejecuta la instrucción de salto, la etapa de ejecución comunica el resultado a la lógica de la tabla e historia de saltos. El estado de la instrucción se actualiza para reflejar una predicción correcta o incorrecta. Si la predicción es incorrecta, la lógica de selección se desvía hacia la dirección correcta de la siguiente captación.

36. ¿Cómo se calcula la línea de cache en la que se ubica un bloque de memoria por correspondencia directa?

Nro linea caché = numero de bloque de memoria principal mod numero de líneas en caché.

37. ¿Qué elementos distintivos poseen los procesadores de VLIW?

Los procesadores VLIW presentan varios elementos distintivos que los diferencian de otras arquitecturas de procesadores:

- **Longitud de la instrucción:** En un procesador VLIW, las instrucciones son muy largas, lo que significa que contienen múltiples operaciones que se ejecutarán en paralelo. Cada instrucción VLIW puede contener operaciones aritméticas, operaciones lógicas, operaciones de carga y almacenamiento, entre otras, que se ejecutarán simultáneamente.
- **Paralelismo explícito:** A diferencia de otros procesadores superscalares, donde el hardware es responsable de encontrar y ejecutar instrucciones independientes en paralelo, en un procesador VLIW, el paralelismo se establece explícitamente en el compilador. El compilador debe analizar el código y agrupar las instrucciones que pueden ejecutarse en paralelo en una sola instrucción VLIW.
- **Ventana de instrucciones:** Los procesadores VLIW utilizan una ventana de instrucciones fija y amplia, que contiene un conjunto de instrucciones que se ejecutarán en paralelo en un ciclo de reloj dado. Esta ventana es estática y se define durante el diseño del procesador.
- **Instrucciones independientes:** Para que el procesador VLIW funcione de manera eficiente, las instrucciones dentro de la ventana deben ser independientes, lo que significa que no deben tener dependencias de datos o control entre sí. Esto permite que todas las operaciones se ejecuten en paralelo sin conflictos.
- **Alto rendimiento y eficiencia energética:** Debido a su naturaleza altamente paralela y a la capacidad de ejecutar múltiples instrucciones en paralelo, los procesadores VLIW pueden lograr un alto rendimiento y una alta eficiencia energética en ciertas aplicaciones específicas que se pueden adaptar adecuadamente para aprovechar el paralelismo.
- **Limitaciones de diseño:** Los procesadores VLIW pueden ser más difíciles de diseñar y programar en comparación con otros diseños de procesadores, ya que el compilador debe ser capaz de identificar y agrupar instrucciones independientes de manera eficiente para generar instrucciones VLIW óptimas.

38. ¿Qué características describen a un cluster?

Un cluster es un grupo de computadores complejos interconectados que trabajan conjuntamente como un único recurso de cómputo creándose la ilusión de que se trata de una sola máquina.

- Cada computador del clúster se denomina nodo, y cada nodo puede funcionar por sí solo, independientemente del cluster.
- Los clusters constituyen la alternativa a los multiprocesadores simétricos para disponer de prestaciones y disponibilidad elevadas.
- Escalabilidad absoluta: es posible configurar clusters grandes que incluso superan las prestaciones de los computadores independientes más potentes. Un cluster puede tener decenas de máquinas, cada una de las cuales puede ser un multiprocesador.
- Escalabilidad incremental: un cluster se configura de forma que sea posible añadir nuevos sistemas al cluster en ampliaciones sucesivas. Así, un usuario puede comenzar con un sistema modesto y ampliarlo a medida que lo necesite, sin tener que sustituir el sistema de que dispone por uno nuevo que proporcione mayores prestaciones.
- Alta disponibilidad: puesto que cada nodo del cluster es un computador autónomo, el fallo de uno de los nodos no significa la pérdida del servicio. En muchos casos, es el software el que proporciona automáticamente la tolerancia a fallos.
- Mejor relación precio-prestaciones: al utilizar elementos estandarizados, es posible configurar un cluster con mayor o igual potencia de cómputo que un computador independiente mayor, a mucho menos costo.

39. ¿Cuáles son los tres conceptos clave de la arquitectura Von-Neumann?

La arquitectura Von Neumann se basa en tres conceptos claves:

- Los datos y las instrucciones se almacenan en una sola memoria de escritura-lectura.
- Los contenidos de esta memoria se direccionan indicando su posición, sin considerar el tipo de dato contenido en la misma.
- La ejecución se produce siguiendo una secuencia de instrucción tras instrucción (a no ser que dicha secuencia se modifique explícitamente).

40. ¿Cuáles son los elementos de una instrucción maquina?

Cada instrucción debe contener la información que necesita el procesador para su ejecución. Los elementos constitutivos de una instrucción de maquina son:

- **Código de operación:** especifica la operación a realizar. La operación se indica mediante un código binario denominado código de operación (codop).
- **Referencia a operandos fuente u origen:** la operación puede implicar a uno o más operandos origen, es decir operandos que son entradas para la instrucción.
- **Referencia al operando de destino o resultado:** la operación puede producir un resultado.
- **Referencia a la siguiente instrucción:** dice al procesador donde captar la siguiente instrucción tras completarse la ejecución de la instrucción actual.

La siguiente instrucción a captar está en memoria. En la mayoría de los casos, es la instrucción que sigue inmediatamente a la instrucción en ejecución. En tales casos no hay referencia explícita a la siguiente instrucción. Cuando sea necesaria una referencia explícita, debe suministrarse de memoria principal o virtual.

41. ¿Qué funciones cumple el módulo de E/S?

Las principales funciones y requisitos de un módulo de E/S se encuentran dentro de las siguientes categorías:

- **Control y temporización:** para poder coordinar el tráfico entre los recursos internos y los dispositivos externos.
- **Comunicación con el procesador:**
  - **Decodificación de órdenes:** el módulo de E/S acepta órdenes del procesador. Estas órdenes generalmente se envían utilizando líneas del bus de control.
  - **Datos:** el procesador y el módulo de E/S intercambian datos a través del bus de datos.
  - **Información de Estado:** puesto que los periféricos son lentos, es importante conocer el estado del módulo de E/S.
  - **Reconocimiento de Dirección:** cada dispositivo de E/S tiene una dirección. Por lo que un módulo de E/S puede reconocer una única dirección para cada uno de los periféricos que controla.
- **Comunicación con los dispositivos:** intercambiar ordenes, información del estado y datos.
- **Almacenamiento temporal de datos:** los datos se almacenan temporalmente en el módulo de E/S y después se envían al periférico a la velocidad de este. En el sentido contrario, los datos se almacenan para no mantener a la memoria ocupada en una operación de transferencia lenta.
- **Detección de errores:** un módulo de E/S es a menudo responsable de la detección de errores y de informar de estos al procesador.

42. ¿Qué clases de atascos en el cauce se pueden encontrar en MIPS?

Existen tres tipos de atascos:

- **Estructurales:** causados por conflictos en los recursos.

- **Por dependencia de datos:** ocurren cuando dos instrucciones se comunican por medio de un dato.
- **Por dependencia de control:** ocurren cuando la ejecución de una instrucción depende de cómo se ejecute otra.

43. ¿Qué tipos de dependencia existen?

- **Dependencia de datos verdadera:** una instrucción genera un dato que lee otra posterior. Si no hay dependencia, se puede captar y ejecutar dos instrucciones en paralelo. En caso de que exista dependencia de datos entre la primera y la segunda instrucción, se retrasa la segunda instrucción tantos ciclos de reloj como sea necesario para eliminar la dependencia.
- **Dependencia de control, o relativa al procedimiento:** ocurre cuando la ejecución de una instrucción depende de cómo se ejecute otra. Las instrucciones que siguen a una bifurcación tienen una dependencia relativa al procedimiento en esa bifurcación y no pueden ejecutarse hasta que se ejecute el salto.
- **Dependencia de salida:** una instrucción escribe un dato después que otra posterior. Solo se da si se deja que las instrucciones adelanten unas a otras (finalización desordenada). La segunda instrucción debe finalizar después de la primera para producir el valor correcto de salida. Para asegurar esto, la emisión de la segunda debe detenerse si su resultado puede ser sobrescrito más tarde por una instrucción anterior que tarda mas en finalizar.
- **Antidependencia:** una instrucción modifica un valor antes que otra anterior que lo tiene que leer, lo lea. Se da por la emisión desordenada. Es similar a la dependencia verdadera, pero a la inversa: en lugar de que la primera instrucción produzca un valor que usa la segunda instrucción, la segunda instrucción destruye un valor que usa la primera instrucción.

44. Describa la Taxonomía de Flynn

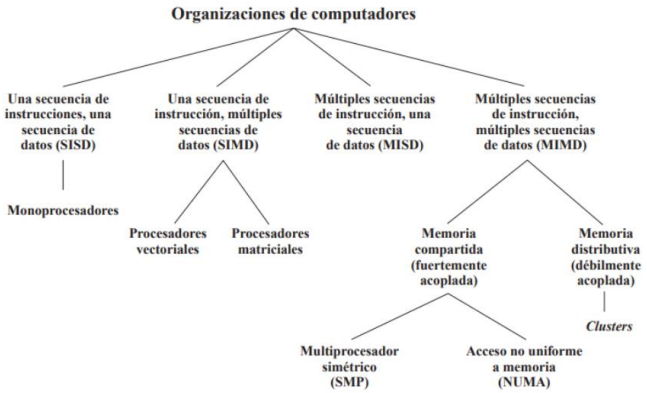


Figura 18.1. Taxonomía de las arquitecturas paralelas.

La taxonomía introducida primeramente por Flynn es todavía la forma más común de clasificar a los sistemas según sus capacidades de procesamiento paralelo. Flynn propuso las siguientes categorías o clases de computadores:

- **Una secuencia de instrucciones y una secuencia de datos (SIDS, Single Instruction Single Data):** un único procesador interpreta una única secuencia de instrucciones para operar con los datos almacenados en una única memoria. Los computadores monoprocesador caen dentro de esta categoría.
- **Una secuencia de instrucciones y múltiples secuencias de datos (SIMD, Single Instruction Multiple Data):** una única instrucción máquina controla paso a paso la ejecución simultanea y sincronizada de un cierto número de elementos de proceso. Cada elemento de proceso tiene una memoria asociada, de forma que cada instrucción es ejecutada por cada procesador con un conjunto de datos diferentes. Los procesadores vectoriales y matriciales pertenecen a esta categoría.
- **Múltiples secuencias de instrucciones y una secuencia de datos (MISD):** se transmite una secuencia de datos a un conjunto de procesadores, cada uno de los cuales ejecuta una secuencia de instrucciones diferente. Esta estructura nunca ha sido implementada.

- **Múltiples secuencias de instrucciones y múltiples secuencias de datos (MIMD):** un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes. Los SMP, los clusters y los sistemas NUMA son ejemplos de esta categoría. Los procesadores son de uso general: cada uno es capaz de procesar todas las instrucciones necesarias para realizar las transformaciones apropiadas de los datos.

45. ¿Qué se entiende por segmentación de cauce? ¿Qué ventajas proporciona su implementación?

La segmentación de cauce es una forma de organizar el hardware de la CPU para realizar más de una operación al mismo tiempo. Consiste en descomponer el proceso de ejecución de las instrucciones en fases o etapas que permitan una ejecución simultánea. Explota el paralelismo entre las instrucciones de un flujo secuencial.

La segmentación mejora las prestaciones a nivel de diseño de hardware. Es invisible al programador.

Teóricamente, el máximo rendimiento posible es completar una instrucción con cada ciclo de reloj. El incremento potencial de la segmentación del cauce es proporcional al numero de etapas del cauce. Incrementa la productividad, pero no reduce el tiempo de ejecución de la instrucción.

46. ¿Qué es la localidad de las referencias?

El termino localidad de las referencias hace referencia a dos términos:

- Localidad Temporal: los elementos de memoria referenciados recientemente tienen alta probabilidad de volver a ser referenciados en un futuro próximo.
- Localidad Espacial: los elementos de memoria cuyas direcciones están próximas a los últimos referenciados van a ser referenciados.