PREGUNTAS DE FINAL – ARQUITECTURA DE COMPUTADORAS

- ***¿QUÉ ES LA SEGMENTACIÓN DEL CAUCE DE INSTRUCCIÓN?***

La segmentación de cauce o "pipelining" es cuando en un extremo se aceptan nuevas entradas cuando algunas entradas anteriores todavía no aparecieron como salidas en el otro extremo.
Para aplicar este concepto a la ejecución de instrucciones, debe mirarse al ciclo de instrucción como una serie de pasos secuenciales.
Si consideramos que el procesamiento de una instrucción tiene dos etapas, captación y ejecución, habrá periodos en la ejecución de una instrucción en los que no se accede a memoria principal. Este tiempo podría utilizarse en captar la siguiente instrucción en paralelo con la ejecución de la actual. Si las etapas de captación y ejecución fueran de la misma duración, el tiempo de ciclo se reduciría a la mitad.
Es una técnica de mejora de prestaciones a nivel de diseño hardware, invisible al programador.
Para conseguir mayor aceleración, el cauce debe partir las instrucciones en más etapas de duración similar.

- ***TRES MOTIVOS DE RETARDO DE CAUCE EN SEGMENTACIÓN DE CAUCE.***

Estructurales: Provocados por conflictos por los recursos. Dos o más instrucciones necesitan utilizar el mismo recurso de hardware en el mismo ciclo. Una solución es un ciclo de parada en una de las instrucciones.
Por **dependencia de datos**: Ocurren cuando dos instrucciones se comunican por medio de un dato (ej.: una lo produce y la otra lo usa)
Por **dependencia de control**: Ocurren cuando la ejecución de una instrucción depende de cómo se ejecute otra (ej.: un salto y los 2 posibles caminos).

- ***SEGMENTACION DE CAUCE: DESCRIBA LOS METODOS Y TECNICAS UTILIZADAS PARA DISMINUIR O EVITAR LAS PARADAS QUE AFECTARAN AL FUNCIONAMIENTO DE LOS CAUCES.***

Para los **atascos estructurales**: replicar, segmentar o realizar turnos para el acceso a las unidades funcionales en conflicto

- Duplicar los recursos de hardware necesarios: duplicar los recursos de la ALU.
- Separar la memoria de instrucciones de la de datos.
- Turnar el acceso al banco de registros: ej. las escrituras en la primera mitad del ciclo y lectura en la segunda mitad del ciclo.

Para los atascos de **dependencia de datos**: para los riesgos RAW de debe determinar cómo y cuándo aparece los riesgos. Sera necesario una unidad de detección de riesgos o un compilador más complejo. Se puede plantear dos soluciones:

- Por hardware: conocida como técnica de adelantamiento, forwarding o cortocircuito. Consiste en pasar el dato obtenido de la instrucción a las instrucciones que lo necesiten como operando. Si el dato está disponible a la salida de la ALU se lleva a la entrada de la etapa correspondiente sin esperar la escritura. Es fácil de implementar si se detectan todos los adelantamientos y se comunican los riesgos de segmentación correspondientes.
- Por software: consiste reordenando las instrucciones del código sin afectar los resultados. Es realizada por el compilador. Hay dos formas:
  - Insertar una instrucción NOP, el problema de esta es que genera retardo.
  - Reordenar las instrucciones, máxima separación con las instrucciones RAW, hay que tener cuidado con la ejecución fuera de orden.

Para los atascos de **dependencia de control**: aquí hay que tener en cuenta los dos tipos de saltos

- Incondicional: la dirección de destino se debe determinar lo más pronto posible, dentro del cauce, para reducir la penalización. Se puede adelantar la resolución del salto en la etapa de decodificación, en ella se decodifica y se sabe que es un salto.
- Condicional: introduce riesgo adicional por la dependencia entre la condición de salto y el resultado de una instrucción previa.

Para tratamiento de saltos hay:

- Técnicas de hardware: consiste en predicciones de salto para evitar la parada. Hay dos tipos las estáticas y dinámicas.

Las estáticas son:
- Predecir que nunca se salta: asume que el salto no se producirá. Siempre capta la siguiente instrucción.
- Predecir que siempre se salta: asume que el salto se producirá. Siempre capta la instrucción destino del salto.
- Predecir según el código de operación: hay instrucciones con más probabilidades de saltar

Las dinámicas son:
- Conmutador saltar/no saltar: basado en la historia de las instrucciones. Es eficaz para los bucles.
- Tabla de historia de saltos: pequeña cache asociada a la etapa de búsqueda. Tres campos: dirección de una instrucción de bifurcación, información de la instrucción destino (dirección del destino o instrucción destino), N bits de estado (historia de uso).

Otras técnicas son:
- Varios cauces (uno por cada opción de salto): precaptan cada salto en diferentes cauces. Se debe utilizar el cauce correcto. La desventaja es que provoca retardos en el acceso al bus y a los registros, y si hay múltiples saltos, se necesita un mayor número de cauces.
- Pre captación del destino de salto: se precapta la instrucción destino del salto, además de las instrucciones siguientes a la bifurcación. La instrucción se guarda hasta que se ejecute la instrucción de bifurcación.
- Buffer de bucles: Memoria muy rápida. Gestionada por la etapa de captación de instrucción del cauce. Comprueba el buffer antes de hacer la captación de memoria. Es muy eficaz para pequeños bucles y saltos.
- Técnicas de software: salto retardado o de relleno de ranura de retardo. El compilador trata de situar instrucciones útiles (que no dependan del salto) en los huecos de retardo. Si no es posible, se utilizan instrucciones NOP. Las instrucciones en los huecos de retardo de salto se captan siempre. La desventaja es que requiere reordenar las instrucciones.

- ***SEGMENTACION DE CAUCE. ¿QUE VENTAJAS PROPORCIONA SU IMPLEMENTACION?***

La ventaja de su implementación es la aceleración de la ejecución de los programas, y las mejoras de rendimiento explicadas en el próximo ítem.

- ***SEGMENTACION DE CAUCE. ¿CUANTO MEJORA SU RENDIMIENTO? DESCRIBA LAS DEPENDENCIAS DE LOS DATOS QUE PUEDAN AFECTAR UN CAUCE SEGMENTADO.***

Teóricamente, la mejora ideal es completar una instrucción con cada ciclo de reloj.
El incremento potencial de la segmentación del cauce es proporcional al número de etapas del cauce.
Si K es el número de etapas del cauce, entonces la velocidad del procesador segmentado equivale a la velocidad secuencial multiplicado por K.
La segmentación de cauce aumenta la productividad total, pero no acelera las instrucciones individualmente.

Los atascos pueden ser por dependencia de datos, dependencia de control y estructurales.
Dependencia de datos es la condición en la que los operandos fuente o destino de una instrucción no están disponibles en el momento en que se necesitan en una etapa determinada del cauce.
Los tipos de dependencias de datos pueden ser:
- Lectura después de Escritura (RAW, dependencia verdadera): una instrucción genera un dato que lee otra posterior.
- Escritura después de Escritura (WAW, dependencia en salida): una instrucción escribe un dato después que otra posterior. Sólo se da si se deja que las instrucciones se adelanten unas a otras.
- Escritura después de Lectura (WAR, anti dependencia): una instrucción modifica un valor antes de que otra anterior que lo tiene que leer, lo lea. No se puede dar en nuestro cauce simple.

- ***EXPLIQUE LOS METODOS DE PASAJE DE ARGUMENTOS A PROCEDIMIENTOS O FUNCIONES. DESCRIBA EL COMPORTAMIENTO CON ANIDAMIENTO DE MULTIPLES PROCEDIMIENTOS/FUNCIONES.***

El pasaje de parámetros se puede realizar a través de:
- Vía registros: en este caso el número de registros es la principal limitación. Además, es importante documentar que registros se usan.
  Normalmente, un procedimiento típico usa solo unos pocos parámetros, y la profundidad de activación de procedimientos también está dentro de un rango relativamente pequeño. Para explotar esas propiedades se

usan múltiples conjuntos pequeños de registros, cada uno asignado a un procedimiento distinto. Una llamada a un procedimiento hace que el procesador conmute automáticamente a una ventana de registro distinta de tamaño fijo en lugar de salvaguardar los registros en memoria. Las ventanas de procedimientos adyacentes están parcialmente solapadas para permitir el paso de parámetros.
La ventana se divide en tres áreas de tamaño fijo. Los registros de parámetros contienen parámetros pasados al procedimiento actual desde el procedimiento que lo llamó, y los resultados a devolver a este. Los resultados locales se usan para variables locales según la asignación que realiza el compilador. Los registros temporales se usan para intercambiar parámetros y resultados con el siguiente nivel más bajo (el procedimiento llamado por el procedimiento en curso). Los registros temporales de un nivel son físicamente los mismos que los registros de parámetros del nivel más abajo adyacente. Este solapamiento posibilita que los parámetros se pasen sin que exista una transferencia de datos real.
Dado que las arquitecturas RISC se implementan con un gran banco de registros, se utiliza la técnica de ventana de registros descripta anteriormente.

- Vía memoria: aquí se usa un área definida de memoria (RAM). Difícil de estandarizar.
- Vía pila (stack): es el método más ampliamente usado, considerado el verdadero "pasaje de parámetros". Es independiente de memoria y registros. Hay que comprender bien cómo funciona porque la pila (stack) es usada por el usuario y por el sistema.
  El manejo de la pila para anidamiento de subrutinas es el siguiente:
  Antes de llamar a la subrutina se debe apilar los parámetros a pasar y la dirección de retorno, al llamar a la subrutina de debe:
  1.Salvar el estado de BP (viejo BP), es decir apilar el valor del BP
  2.Cargar el valor del BP el del SP
  3.Reservar espacio para datos locales, si los hay
  4.Salvar valores de otros registros, si es que se van a modificar los registros
  5.Acceder a parámetros, para acceder a los parámetros se le debe sumar un desplazamiento al BP para acceder a la posición de la pila en la que están los parámetros. En general el desplazamiento es: 2 (es el tamaño de BP apilado) + tamaño de dirección de retorno + total de tamaño de parámetros entre el buscado y BP Aquí se puede llamar a otra subrutina o regresar a la anterior, si se vuelve a hacer una llamada se debe apilar los parámetros a pasar a la subrutina y la dirección de retorno. La subrutina llamada debe repetir los pasos anteriores más los que siguen. Si no se hace otra llamada a subrutina también se deberá seguir los siguientes pasos:
  7.Retornar parámetro, si se tiene que retornar datos
  8.Regresar correctamente del procedimiento (desapilar todo lo que apilo, volver a cargar a cargar el valor del BP que tenía antes de entrar a la subrutina

- ***¿QUE ES UN BUS? TIPOS DE BUSES, TEMPORIZACION Y METODOS DE ARBITRAJES.***

Un bus es un camino de interconexión entre dos o más dispositivos, es un medio de transmisión compartido. Cada señal que se comparte en un bus, está disponible para todos los dispositivos conectados. Usualmente, un bus está constituido por líneas de comunicación. Cada línea es capaz de transmitir señales binarias, las transmisiones pueden ser una serie de bits en un intervalo de tiempo, o usando varias líneas se pueden transmitir los bits en paralelo. (Acá se podría hablar sobre la estructura, líneas de datos, dirección y control)

Elementos de diseño de un bus:

- Tipos de buses: Las líneas del bus pueden ser de dos tipos, dedicadas o multiplexadas. Las dedicadas se usan para una sola función, como por ejemplo una línea de direcciones. Las multiplexadas permiten ahorrar espacio y costo, ya que se usa la misma línea para varias funciones, como por ejemplo para direcciones y datos. Usando otra línea como control, primero se coloca una dirección en el bus, y luego de un tiempo se usa para transmitir los datos.
- Método de arbitraje: Es el método con que se supervisa el uso del bus, ya que normalmente son varios los dispositivos que necesitan usarlo: puede ser centralizado o distribuido. Con el esquema centralizado, hay un único dispositivo llamado controlador o árbitro del bus, que se encarga de asignar los tiempos de uso del bus. En el esquema distribuido, no hay un controlador central, sino que cada módulo contiene la lógica necesaria para controlar el acceso, y los módulos trabajan en conjunto para coordinarse y compartir el bus.

- Temporización: Es la forma en la que se coordinan los eventos en el bus. Puede ser síncrona o asíncrona. Con la temporización síncrona, el bus incluye una línea de reloj, por la que se alternan en intervalos regulares una secuencia de unos y ceros. Cada intervalo es un ciclo de reloj. Todos los dispositivos pueden leer la línea de reloj, y todos los eventos empiezan al principio del ciclo de reloj. En la temporización asíncrona, la presencia de un evento en el bus es consecuencia de un evento previo. Por ejemplo, el procesador puede poner las señales de dirección y lectura en el bus, y activar la señal de "master sync". Entonces el módulo de memoria proporciona el dato y activa la señal "slave sync".
- Anchura del bus: Es la cantidad de líneas distintas del bus de datos y de direcciones. Cada línea transporta 1 bit, entonces la anchura determina la cantidad de bits que se pueden transmitir simultáneamente, afectando de manera directa al rendimiento. Pueden ser 32, 64, 128, o más.
- Tipo de transferencia de datos: Puede ser de lectura, escritura, y en algunos casos una combinación de ambas. Por ejemplo, lectura-modificación-escritura, es simplemente una lectura seguida de escritura inmediatamente, de manera indivisible.

- ***DESCRIBA LAS LIMITACIONES EXISTENTES AL PARALELISMO AL NIVEL DE INSTRUCCIONES.***

En el paralelismo existen las siguientes limitaciones:
- Dependencia de datos verdadera: la segunda instrucción se puede captar y decodificar, pero no se puede ejecutar hasta que finalice la ejecución de la primera instrucción. El motivo es que la segunda instrucción necesita un dato producido por la primera instrucción.
- Dependencia relativa al procedimiento: la presencia de saltos en una secuencia de instrucciones complica el funcionamiento del cauce. Las instrucciones que siguen a una bifurcación tienen una dependencia relativa a procedimiento en esa bifurcación y no puede ejecutarse hasta que se ejecute el salto. Las consecuencias en un cause superescalar son más graves, ya que se pierde un mayor número de oportunidades de comenzar a ejecutar instrucciones en cada retardo.
- Conflicto con los recursos: es una pugna entre dos o más instrucciones por el mismo recurso al mismo tiempo.
- Dependencia de salida: esta dependencia se da en una ejecución de finalización de instrucciones desordenada. En una escritura después de una escritura, con la finalización desordenada puede que una instrucción posterior que modifica en dato termine antes que la que modifica el dato primero, de esta forma el dato quedaría con un valor erróneo.
- Antidependencia: esta se da en la ejecución de instrucciones con emisión desordenada. Es similar a la dependencia de datos verdadera, pero a la inversa, en lugar de que la primera instrucción produzca un valor que la segunda usa, la segunda instrucción destruye un valor que usa la primera instrucción.

- ***DESCRIBA LAS CARACTERISTICAS DEL CONTROLADOR DE INTERRUPCIONES PIC.***

Un PIC es un dispositivo controlador programable de instrucciones. Es un dispositivo usado para combinar varias fuentes de interrupciones sobre una o más líneas del CPU, mientras que permite que los niveles de prioridad sean asignados a sus salidas de interrupción. Cuando el dispositivo tiene múltiples salidas de interrupción a imponer, las impondrá en orden de su prioridad relativa. Los modos comunes de un PIC incluyen prioridades duras, prioridades rotativas, y prioridades en cascada. Los PICs a menudo permiten la conexión en cascada de sus salidas a las entradas entre uno y otro.
Los PICs típicamente tienen un conjunto común de registros: Interrupt Request Register (IRR), In-Service Register (ISR), Interrupt Mask Register (IMR). El IRR especifica qué interrupciones están pendientes de reconocimiento, y es típicamente un registro interno que no puede ser accedido directamente. El registro ISR especifica qué interrupciones han sido reconocidas, pero todavía están esperando por un final de interrupción (EOI). El IMR especifica qué interrupciones deben ser ignoradas y no ser reconocidas. Un esquema simple de registros como este, permite hasta dos distintas peticiones de interrupción estén pendientes a un tiempo, una esperando por reconocimiento, y una esperando por EOI.
/// Además tiene los 8 registros para las interrupciones donde carga el valor del vector de interrupción correspondiente
En los PICs hay un número de esquemas de prioridad comunes, incluyendo prioridades duras, prioridades específicas, y prioridades rotativas.
Las interrupciones pueden ser disparadas por borde o por nivel.

Hay un número de formas comunes de reconocer que una interrupción ha terminado cuando es emitido un EOI. Éstas incluyen especificar qué interrupción se terminó, usando una interrupción implícita que ha terminado (usualmente la prioridad más elevada está pendiente en el ISR), y tratar el reconocimiento de la interrupción como el EOI.

- ***DESCRIBIR EL MECANISMO DE INTERRUPCION. MENCIONAR CUALES SON LAS FUENTES DE INTERRUPCION (TIPOS). DESCRIBIR EL TRATAMIENTO DE MULTIPLES INTERRUPCIONES.***

Fuentes de interrupción:

- Interrupciones por hardware: Son las generadas por dispositivos de E/S.
    - Son las "verdaderas" interrupciones.
    - El sistema de cómputo tiene que manejar estos eventos externos "no planeados" ó "asincrónicos".
    - No están relacionadas con el proceso en ejecución en ese momento.
    - Son conocidas como interrupt request.
- Traps/excepciones
    - Interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos como:
    - Condiciones excepcionales: overflow en ALU de punto flotante.
    - Falla de programa: tratar de ejecutar una instrucción no definida.
    - Fallas de hardware: error de paridad de memoria.
    - Accesos no alineados o a zonas de memoria protegidos
- Interrupciones por software

  Muchos procesadores tienen instrucciones explícitas que afectan al procesador de la misma manera que las interrupciones por hardware.
    - Generalmente usadas para hacer llamadas a funciones del SO.
    - Esta característica permite que las subrutinas del sistema se carguen en cualquier lugar.
    - No requieren conocer la dirección de la rutina en tiempo de ejecución.

Si hay múltiples fuentes que pueden solicitar interrupción se establece cuales son más importantes.
Se consideran:

- No Enmascarables: las que NO pueden ignorarse. Indican eventos peligrosos o de alta prioridad.
- Enmascarables: pueden ser ignoradas. Con instrucciones podemos inhibir la posible solicitud.

- ***INTERRUPCIONES MULTIPLES.***

Un programa puede estar recibiendo datos a través de una línea de comunicación e imprimir los resultados, por lo que tendría múltiples interrupciones (cada vez que llegue un dato y cada vez que se termine de escribir). Por lo tanto, para tratar las interrupciones múltiples se pueden seguir dos alternativas:

- Desactivar las interrupciones mientras se está procesando una interrupción, por lo que el procesador ignorará la interrupción y la dejará pendiente para cuando termine de atender la interrupción actual (no puede decidir entre interrupciones más importantes que otras). Las interrupciones se manejan en un orden secuencial estricto.
- Definir prioridades para las interrupciones, permitiendo que una segunda interrupción de prioridad más alta pueda interrumpir a un gestor de interrupción de prioridad menor. Cuando se ha gestionado la interrupción de prioridad más alta, el procesador vuelve a las interrupciones previas (de menor prioridad). Terminadas todas las rutinas de gestión de interrupciones se retoma el programa del usuario.

- ***EXPLIQUE EL MECANISMO DE INTERRUPCION. DESCRIBA LAS CARACTERISTICAS Y EL FUNCIONAMIENTO DE UN PIC.***

La entrada-salida por interrupción es un mecanismo en el cual un dispositivo "interrumpe" al procesador para solicitar su servicio cuando esté preparado para intercambiar datos con él.
Esto permite que el procesador no tenga que esperar, comprobando repetidamente el estado del módulo, hasta que el módulo pueda transmitir o recibir los datos. Dado que los dispositivos son mucho más lentos que el procesador, no usar interrupciones haría caer el nivel de prestaciones de todo el sistema.
Desde el punto de vista del módulo de E/S, se recibe un read del procesador, entonces el módulo procede a leer los datos solicitados desde el periférico. Una vez que tiene el dato en el registro de datos del módulo, envía una interrupción al procesador a través de una línea de control. Espera a que el procesador solicite su dato, y cuando

esto sucede, sitúa el dato en el bus de datos.
Desde el punto de vista del procesador, por ej. para una lectura, envía un read al módulo E/S, y pasa a realizar otro trabajo. Al final de cada ciclo de instrucción, comprueba si hay interrupción. Cuando hay solicitud de interrupción, el procesador guarda el contexto del programa en curso, y procesa la interrupción, en este caso sería leer el dato desde el módulo de E/S y guardarlo en memoria. Luego recupera el contexto del programa que estaba ejecutando, y continúa la ejecución.

- ***DESCRIBA LAS CARACTERISTICAS QUE DIFERENCIAN A LOS PROCESADORES RISC RESPECETO DE LOS CISC.***

RISC: Computadoras de repertorio reducido de instrucciones (Reduced Instruction Set Computer). Tienen un gran número de registros de uso general, el repertorio de instrucciones es limitado y sencillo ("reducido"), y se hace un gran énfasis en la optimización de la segmentación de instrucciones.

CISC (Complex Instruction Set Computer) en cambio, tiene un repertorio de instrucciones más amplio, con la intención de facilitar el trabajo de los escritores de compiladores. También pretenden mejorar la eficiencia de la ejecución, a través de secuencias complejas de operaciones en micro código, y dar soporte a lenguajes de alto nivel (HLL) más complejos.
El razonamiento es que, si hay instrucciones de máquina que se parezcan a las de HLL, la tarea se simplifica. El problema con esto es que las instrucciones de máquina complejas suelen ser difíciles de aprovechar, ya que el compilador debe descubrir los casos que se ajusten perfectamente.
La tarea de optimizar el código generado para minimizar su tamaño, reducir el número de instrucciones ejecutadas y mejorar la segmentación es mucho más difícil con un repertorio complejo de instrucciones.

Características de RISC:
- Una instrucción por ciclo: Se ejecuta una instrucción máquina cada ciclo máquina. Con instrucciones sencillas y de un ciclo, hay poca o ninguna necesidad de microcódigo, las instrucciones de maquina pueden estar cableadas, lo que significa que no hay que acceder a la memoria de control de microprograma durante la ejecución de la instrucción.
- Operaciones registro a registro: La mayoría de las operaciones deben ser de esta manera, lo que simplifica el repertorio de instrucciones y fomenta la optimización del uso de los registros, ya que los operandos accedidos frecuentemente, permanecen en el almacenamiento de alta velocidad.
- Modos de direccionamiento sencillos: se usa principalmente el sencillo direccionamiento a registro
- Formatos de instrucción sencillos: Sólo se usa un formato, o unos pocos. La longitud de las instrucciones es fija, y alineada en los límites de una palabra. Las posiciones de los campos, especialmente la del código de operación, son fijas. Todo esto simplifica la unidad de control.

- ***ESTRUCTURA DE UN MODULO DE E/S.***

Un módulo de entrada/salida permite que el procesador vea a una amplia gama de dispositivos externos de manera simplificada. El módulo debe ocultar los detalles de temporización, electromecánica y formatos de los dispositivos, para que el procesador pueda funcionar en términos de órdenes de lectura y escritura.
El módulo se conecta con el resto de la máquina a través de líneas, como pueden ser las líneas del bus del sistema. Los datos que se transfieren a y desde el módulo, se almacenan temporalmente en registros. Además, puede haber registros de estado que proporcionan información del estado actual. Un registro de estado también puede funcionar como información de control, para recibir información de control del procesador. La lógica que hay en el módulo interactúa con el procesador a través de una serie de líneas de control, usadas por el procesador para proporcionar las órdenes al módulo de E/S. Algunas de las líneas de control pueden ser usadas por el módulo E/S, por ejemplo, para señales de arbitraje y estado.
El módulo también debe ser capaz que reconocer y generar las direcciones asociadas a los dispositivos que controla. Cada módulo E/S tiene una dirección única, y si controla más de un dispositivo, tiene un conjunto de direcciones únicas.
Cada módulo posee la lógica específica para cada una de las interfaces de los dispositivos que controla.

- ***CARACTERISTICAS FUNCIONALES DE DMA.***

DMA (Direct Memory Access) es un módulo adicional en el bus de sistema, que tiene la capacidad de imitar al

procesador y tomar el control del sistema, con la función de realizar transferencias de E/S de datos, de manera directa entre los periféricos y la memoria, a través del bus del sistema. Para hacerlo, el módulo de DMA debe utilizarlo solo cuando el procesador no lo necesita, o debe forzar al procesador a que suspenda temporalmente su funcionamiento. Esta última técnica es la más común, y se denomina robo de ciclo, puesto que en efecto, el módulo de DMA roba un ciclo de bus.
Cuando el procesador desea leer o escribir un bloque de datos, envía una orden al módulo de DMA, incluyendo la siguiente información:

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

Después el procesador continúa con otro trabajo. Ha delegado la operación de E/S al módulo de DMA que se encargará de ella. EL módulo de DMA transfiere el bloque completo de datos, palabra a palabra directamente desde o hacia la memoria, sin que tenga que pasar a través del procesador. Cuando la transferencia ha terminado, el módulo de DMA envía una señal de interrupción al procesador.

- ***DESCRIBA LOS ELEMENTOS A TENER EN CUENTA EN EL DISEÑO DE UNA MEMORIA CACHE. ANALICE LAS VENTAJAS Y DESVENTAJAS DE POSEER VARIOS NIVELES DE CACHE.***

Los principales elementos a tener en cuenta son:

- Tamaño de caché: Se necesita hacer un balance entre el tamaño (mientras más grande es más costoso) y la performance, que lo ideal es tener mucho tamaño, pero si son muy grandes tienden a ser ligeramente más lentas. También está limitado por la superficie disponible de chip y de tarjeta.
- Función de correspondencia: Son directa, asociativa y asociativa por conjunto. Directa es la correspondencia más sencilla. Consiste en hacer corresponder cada bloque de la memoria principal a solo una línea posible de la cache. i = j mod m; donde i es el número de línea de caché, j el número de bloque de memoria principal, m el número de líneas de la caché. La ventaja es que es muy sencilla de implementar. La desventaja es que hay una posición concreta de chache para cada bloque dado. Por ello, si un programa referencia repetidas veces a palabras de los bloques diferentes asignados en la misma línea dichos bloquean se estarían intercambiando continuamente en la caché, y la tasa de aciertos sería baja.

  La correspondencia asociativa supera la desventaja de la directa, permitiendo que cada bloque de memoria principal pueda cargarse en cualquier línea de la cache. La lógica de control de la cache interpreta una dirección de memoria simplemente como una etiqueta y un campo de palabra. El campo de etiqueta identifica inequívocamente un bloque de memoria principal. Para determinar si un bloque está en la cache, su lógica de control debe examinar simultáneamente todas las etiquetas de líneas para buscar una coincidencia. La desventaja es la compleja circuitería necesaria para examinar en paralelo las etiquetas de todas las líneas de cache.

  La correspondencia asociativa por conjuntos intenta recoger lo positivo de las dos técnicas anteriores. La cache se divide en v conjuntos, cada uno de k líneas. Las relaciones que se tienen son:
  m = v*k
  i=j mod v;
  i = número de conjunto de cache
  j = número de bloque de memoria principal
  m = número de líneas de la cache
- Algoritmos de sustitución: LRU; FIFO; LFU, aleatorio.
- Política de escritura:
  - Escritura inmediata: todas las operaciones de escritura se hacen tanto en la caché como en la memoria principal, para evitar inconsistencias.

    - Post-escritura: La información sólo se actualiza en la caché. Se marca como actualizada. bit de "sucio". La memoria principal se actualiza en el reemplazo y puede contener información errónea en algún momento.
- Tamaño de línea: A medida que aumenta el tamaño del bloque, la tasa de aciertos primero aumenta debido al principio de localidad, al aumentar el tamaño del bloque, más datos útiles son llevados a la memoria caché. Sin embargo, la tasa de aciertos comenzará a decrecer cuando el tamaño de bloque se haga aún mayor. Hay dos efectos: bloques más grandes reducen el número de bloques que caben en la caché, y a medida que un bloque se hace más grande, cada palabra adicional está más lejos de la requerida, y por lo tanto es más improbable que sea necesaria a corto plazo.
- Número de cachés: Con el aumento de densidad de integración ha sido posible tener una cache en el mismo chip del procesador. Esto reduce la actividad del bus externo del procesador y por lo tanto reduce los tiempos de ejecución, e incrementa las prestaciones globales del sistema. Los diseños más actuales incluyen tanto cache on-chip como internos. La estructura más sencilla se denomina cache de dos niveles, siendo la cache interna el nivel 1 (L1) y la externa el nivel 2 (L2). En general, el uso de cache multinivel mejora las prestaciones, pero aumenta la complejidad del diseño.
- ***CACHE: POLITICAS DE ESCRITURA DESDE EL PUNTO DE VISTA DE LA COHERENCIA DE DATOS.***

En una estructura en la que más de un procesador tiene un a cache y la memoria principal es compartida, si se modifican los datos de una cache, se invalida no solamente la palabra correspondiente de memoria principal, sino también la misma palabra en otras caches. Un sistema que evite este problema se dice que mantiene la coherencia de caché. Entre las posibles aproximaciones a la coherencia de cache se incluyen:

- Vigilancia del bus con escritura inmediata: cada controlador de cache monitoriza las líneas de direcciones para detectar operaciones de escritura en memoria, por parte de otros maestros en el bus. Si otro maestro escribe en una posición de memoria compartida que también reside en la memoria cache, el controlador de cache invalida el elemento de la cache. Esta estrategia depende del uso de una política de escritura inmediata por parte de todos los controladores de cache.
- Transparencia hardware: se utiliza hardware adicional para asegurar que todas las actualizaciones de memoria principal vía cache quedan reflejadas en todas las cache.
- Memoria excluida de cache: solo una porción de memoria principal se comparte con más de un procesador y esta se diseña como no transferible a cache. En un sistema de este tipo todos los accesos a la memoria compartida son fallos de cache porque la memoria compartida nunca se copia a cache.

- ***CACHE: DESCRIBIR LAS POLITICAS DE ESCRITURA (EN ACIERTO Y FALLO).***

Escritura y escritura inmediata son políticas de escritura en acierto.
En fallo se usan:

- Write allocate: La información se lleva de la memoria principal a la caché. Se sobrescribe en la caché. Es habitual con write-back (post escritura).
- No-write allocate: El bloque no se lleva a la memoria caché. Se escribe directamente en la memoria principal. Habitual con write-through (escritura inmediata).

- ***¿QUE CARACTERISTICAS POSEE UN PROCESADOR SUPERESCALAR?***

Un procesador superescalar es aquel que usa múltiples cauces de instrucciones independientes. Cada cauce consta de múltiples etapas, de modo que puede tratar varias instrucciones a la vez. El hecho de que haya varios cauces introduce un nuevo nivel de paralelismo, permitiendo que varios flujos de instrucciones se procesen simultáneamente. Un procesador superescalar saca provecho de lo que se conoce como "paralelismo al nivel de instrucciones", que hace referencia al grado en que la instrucción de un programa puede ejecutarse en paralelo.
Típicamente, un procesador superescalar capta varias instrucciones a la vez, y a continuación, intenta encontrar instrucciones cercanas que sean independientes entre sí, y puedan, por consiguiente, ejecutarse en paralelo. Si la entrada de una instrucción depende de la salida de una instrucción precedente, la segunda instrucción no puede completar su ejecución al mismo tiempo ni antes que la primera. Una vez que se han identificado tales dependencias, el procesador puede emitir y completar instrucciones en un orden diferente al del código de máquina original.
El procesador puede eliminar algunas dependencias innecesarias mediante el uso de registros adicionales y el

renombramiento de las referencias a registros en el código original.

- ***CARACTERISTICAS DE LOS CLUSTERS.***

Un cluster es un grupo de computadores completos interconectados, que trabajan conjuntamente como un único recurso de cómputo, creándose la ilusión de que se trata de una sola máquina. Un "computador completo" es un sistema que puede funcionar por sí solo, independientemente del cluster.
Las principales ventajas que se pueden conseguir con un cluster son:

- Escalabilidad absoluta: Es posible configurar clusters grandes, que incluso superen a las computadoras independientes más potentes. Se pueden tener decenas de máquinas.
- Escalabilidad incremental: Se configura de manera tal que se pueden añadir nuevos sistemas al cluster en ampliaciones sucesivas.
- Alta disponibilidad: Dado que cada computadora del cluster es un sistema autónomo, el fallo de un nodo no implica que deje de funcionar todo el cluster. Generalmente, el software incorpora el manejamiento de fallos.
- Buena relación precio/prestaciones: Se usan elementos estandarizados, por lo que se puede configurar un cluster con mayor rendimiento de cómputo que un computador independiente mayor, a menor costo.

- ***DESCRIBAS LAS CARACTERISTICAS QUE DIFERENCIAN LOS SMP RESPECTO DE LOS CLUSTERS.***

SMP (Symetric Multi Processors) es un término que se refiere a la arquitectura hardware del computador, y también al comportamiento del sistema operativo que utiliza dicha arquitectura. Un SMP puede definirse como un computador con las siguientes características:

- Hay dos o más procesadores similares con capacidades comparables
- Estos procesadores comparten la memoria principal y las E/S, y están interconectados mediante un bus u otro tipo de sistema de interconexión de forma que el tiempo de acceso a memoria es aproximadamente el mismo para todos los procesadores.
- Todos los procesadores comparten los dispositivos de E/S, bien a través de los mismo canales o mediante canales distintos que proporcionan caminos de acceso al mismo dispositivo.
- Todos los procesadores pueden desempeñar las mismas funciones, de ahí el término "simétrico".
- El sistema está controlado por un SO integrado que proporciona las interacciones entre los procesadores y sus programas a los niveles de trabajo, tarea, fichero y dato.

En los clusters la unidad de interacción física es normalmente un mensaje o un fichero completo. En un SMP la interacción se puede producir a través de elementos de datos individuales y puede existir un elevado nivel de cooperación entre procesadores.
La principal ventaja de un SMP es que resulta más fácil de gestionar y configurar que un cluster. El SMP es mucho más cerca del modelo de computador de un solo procesador para el que están disponibles casi todas las aplicaciones. El principal cambio que se necesita para pasar de un computador monoprocesador a un SMP se refiere al funcionamiento del planificador. Otra ventaja de un SMP es que necesita menos espacio físico, y consume menos energía que un cluster comparable. Los SMP son plataformas estables y bien establecidas.

- ***QUE CARACTERISTICAS POSEE UN PROCESADOR SUPERSEGMENTADO FRENTE A UN SUPERESCALAR.***

La supersegmentación aprovecha el hecho de que muchas etapas del cauce realizan tareas que requieren menos de medio ciclo de reloj. De este modo, doblando la velocidad del reloj interno, se permite la realización de dos tareas en un ciclo de reloj externo.
El superescalar, en cambio, tiene múltiples cauces de instrucciones independientes. Cada cauce consta de múltiples etapas, de modo que puede tratar varias instrucciones a la vez. El hecho de que haya varios cauces introduce un nuevo nivel de paralelismo, permitiendo que varios flujos de instrucciones se procesen simultáneamente. Un procesador superescalar saca provecho de lo que se conoce como paralelismo en las instrucciones, que hace referencia al grado en que las instrucciones de un programa pueden ejecutarse en paralelo.
Ambas técnicas, si trabajan ininterrumpidamente, pueden procesar la misma cantidad de instrucciones en el mismo tiempo. El procesador supersegmentado se queda atrás con respecto al procesador superescalar al comienzo del programa y en cada destino de un salto.