Pasaje de argumentos

1. Explique los métodos de pasaje de argumentos a procedimientos o funciones.

   El pasaje de argumentos a procedimientos o funciones pueden ser:

   - Vía Registros: El número de registros es la principal limitación. Es importante documentar qué registros se usan.
   - Vía Memoria: Se usa un área definida de memoria (RAM). Difícil de estandarizar
   - Vía Pila (Stack): Es el método más ampliamente usado. El verdadero "pasaje de parámetros". Independientemente de memoria y registros. Hay que comprender bien cómo funciona porque la pila es usada tanto por el usuario como por el sistema.

2. ¿Qué es una pila? Describir el comportamiento con anidamiento de múltiples procedimientos/funciones utilizando pila.

   La pila o stack es una estructura de datos que sigue el principio LIFO y se utiliza para gestionar la ejecución de programas y el uso de memoria de manera eficiente. Es un bloque de direcciones de memoria reservado para esta funcionalidad.

   Consta de dos operaciones básicas:

   - PUSH → Agregar (apilar) un elemento en la parte superior de la pila.
   - POP → Retirar (desapilar) el elemento superior de la pila.

   Cuando anidamos múltiples procedimientos o funciones utilizando una pila, se está agregando llamadas a funciones dentro de otras funciones. Cada vez que se llama a una nueva función, se agrega un nuevo "marco de pila", que contiene variables locales, parámetros, la dirección de retorno, y cualquier otra información necesaria para la ejecución de esa función.

   A medida que las funciones anidadas se completan, sus marcos de pila se retiran de la pila, y el control regresa a la función que las llamó.

   Se debe:

   1- Salvar el estado del BP
   2- Salvar el estado de SP

3- Reservar espacio para datos locales (opcional)
4- Salvar valores de otros registros (opcional)
5- Acceder a parámetros

Luego, para retornar:

- Los registros salvados deben ser descargados en orden inverso
- Si se reservó espacio para variables locales, se debe reponer SP con el valor de BP que no cambió durante el procedimiento
- Reponer BP
- Volver al programa que llamó al procedimiento con RET.

Mecanismo de interrupción

1. ¿Qué es una interrupción? ¿Cuál es la función de un controlador de interrupciones?

   Una interrupción es el mecanismo mediante el cual se puede interrumpir el procesamiento normal de la CPU. Pueden ser origen interno o externo a la misma.

   La función de un controlador de interrupciones es gestionar y coordinar las interrupciones. Se encarga de determinar qué dispositivo o evento generó la interrupción, priorizarlas en caso de múltiples interrupciones simultáneas y luego redirigir el flujo de ejecución del procesador hacia un programa de manejo de interrupciones específico.

2. Describa el mecanismo de interrupción.

   El mecanismo de interrupción funciona de la siguiente manera:

   1- Generar la interrupción: Un dispositivo o componente del sistema detecta un evento que requiere atención, por lo que genera una señal de interrupción.
   2- Interrupción del procesador: Cuando se genera una interrupción, el procesador suspende temporalmente la ejecución del programa actual y cambia su enfoque hacia el controlador de interrupciones.
   3- Guarda el estado: Antes de manejar la interrupción, el procesador guarda el estado actual del programa en ejecución. Esto incluye la posición del programa en la que se detuvo, los registros de la CPU y otros datos relevantes.

4- Rutina de manejo de interrupciones: El controlador de interrupciones determina la naturaleza de la interrupción y consulta una tabla de vectores de interrupción que contiene direcciones de memoria de rutinas específicas de manejo de interrupciones. Estas rutinas son porciones de código que se encargan de gestionar el evento específico asociado con la interrupción.
5- Ejecución de la rutina de manejo: El procesador comienza a ejecutar la rutina de manejo de interrupciones correspondiente. Esta rutina se encarga de procesar el evento que generó la interrupción. Una vez que la rutina ha sido ejecutada, el control es devuelto al punto en el programa donde se detuvo inicialmente.
6- Restauración del estado: Después de que se maneja la interrupción, el procesador restaura el estado previo guardado, incluyendo los registros y la posición del programa, permitiendo que el programa original se reanude como si la interrupción nunca hubiera ocurrido.
7- Continuación de la ejecución: Con el estado restaurado, el programa original continúa su ejecución desde el punto en el que se interrumpió.

El mecanismo de interrupción permite a los sistemas manejar eventos en tiempo real, realizar tares de gestión de dispositivos y responder eficazmente a situaciones imprevistas sin detener por completo la operación del sistema.

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

La subrutina regresa al programa principal utilizando la instrucción de retorno RET. En cambio, un gestor de interrupción lo hace después de manejar dicha interrupción, o con la instrucción de retorno IRET.

La instrucción IRET es similar a una instrucción RET, por utilizar la pila, pero recupera una copia del registro de estado y la dirección de retorno

4. Explique características y tratamiento de interrupciones múltiples.

   Las interrupciones múltiples se refiere a la situación en la que ocurren varias interrupciones mientras se está manejando una interrupción previa. Las características a tener en cuenta son:

   - Prioridad de interrupciones: Las interrupciones suelen tener un nivel de prioridad asignado. Cuando se producen múltiples interrupciones, el sistema debe determinar cuál de ellas se manejará primero en función de su prioridad.
   - Máscara de interrupciones: En algunos sistemas, es posible desactivar temporalmente ciertas interrupciones (de menor prioridad) utilizando máscaras de interrupción.
   - Nivel de anidación: El sistema debe ser capaz de manejar múltiples niveles de anidación de interrupciones. Si se está manejando una interrupción y ocurre otra de mayor prioridad, el sistema debe ser capaz de suspender temporalmente la primera interrupción, manejar la segunda y luego regresar al manejo de la primera.
   - Protección del estado del sistema: Durante un manejo de una interrupción, se deben tomar medidas para proteger el estado actual del sistema. Esto generalmente implica guardar los registros y datos críticos antes de comenzar a manejar la interrupción y restaurarlos después de que se haya completado el manejo de la interrupción.

   El tratamiento de interrupciones múltiples puede variar, pero en general sigue un enfoque de manejo de interrupciones en cascada. Los pasos que suele seguir son los siguientes:

   1- El sistema determina la prioridad de las interrupciones en cola y selecciona la de mayor prioridad para manejarla primero.
   2- Se guarda el estado actual del sistema, incluidos los registros y otros datos relevantes, para que pueda restaurarse más tarde.
   3- Se ejecuta la rutina de manejo de la interrupción correspondiente a la interrupción seleccionada.
   4- Después de que se complete el manejo de la interrupción, se restaura el estado del sistema que se guardó previamente.

5- Si aún quedan interrupciones en cola, el sistema selecciona la siguiente interrupción de mayor prioridad y repite el proceso.
6- Cuando no quedan interrupciones en cola o cuando se han manejado todas las interrupciones necesarias, se reanuda la ejecución normal del programa.

5. Describa las características y el funcionamiento de un PIC.

Un PIC (Controlador Programable de Interrupciones) 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. Los PICs tienen un conjunto común de registros:

- IRR, petición de interrupciones, especifica qué interrupciones están pendientes de reconocimiento, suele ser un registro interno que no puede ser accedido directamente.
- ISR, interrupción en servicio, especifica qué interrupciones han sido reconocidas, pero todavía están esperando por un final de interrupción (EOI).
- IMR, máscara de interrupciones, especifica qué interrupciones deben ser ignoradas y no reconocidas.
- INT0...INT7, cada uno de estos registros se relaciona con un vector de interrupción.

El funcionamiento de un PIC implica varios pasos:

1- Detección de interrupciones: El PIC monitorea constantemente las líneas de interrupción de los dispositivos periféricos y otros eventos en busca de cambios en su estado. Cuando se detecta una interrupción, se genera una solicitud de interrupción hacia el procesador.
2- Priorización: Si hay múltiples interrupciones pendientes, el PIC selecciona la de mayor prioridad según su configuración. Esto se basa en las prioridades asignadas a las líneas de interrupción.
3- Enmascaramiento: Antes de notificar al procesador sobre una interrupción, verifica si la línea de interrupción está habilitada o enmascarada. Si la línea está enmascarada, la interrupción se ignora.

4- Vectorización: Una vez que se determina la interrupción que debe atenderse, el PIC envía un vector de interrupción a la CPU. Este vector contiene información sobre la fuente de la interrupción y apunta a la dirección de memoria de la rutina de manejo de interrupciones correspondiente.
5- Procesamiento de interrupción: La CPU recibe el vector de interrupción y ejecuta la rutina de manejo de interrupciones asociada. Esta rutina es la encargada de gestionar la interrupción, guardar el estado necesario y realizar cualquier acción requerida.
6- Fin de interrupción (EOI): Después de manejar la interrupción, la CPU envía una señal de Fin de Interrupción (EOI) al PIC para indicar que ha completado el manejo de la interrupción actual. Esto permite que el PIC se prepare para la siguiente interrupción si la hubiera.

6. Describa las distintas fuentes de interrupción (tipos de interrupción) que conozca y el tratamiento a realizar cuando hay múltiples interrupciones.
   Las interrupciones son el mecanismo mediante el cual se puede interrumpir el procesamiento normal de la CPU. Estas pueden ser de origen interno o externo a la CPU. Las interrupciones pueden ser:
   - Por Hardware: Son las llamadas “verdaderas” interrupciones. Son generadas por dispositivos de E/S (como teclado o mouse) para señalar eventos (como la pulsación de una tecla). El tratamiento implica determinar la fuente de la interrupción, ejecutar la rutina de manejo correspondiente y, si hay múltiples interrupciones de hardware, priorizarlas según su importancia y manejarlas secuencialmente en función de su prioridad.
   - Traps/Excepciones: Son interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos como: condiciones excepcionales (overflow en la ALU), falla de programa (tratar de ejecutar una instrucción no definida), fallas de hardware (error de paridad de memoria) o, accesos no alineados o a zonas de memoria restringidas.

- Por Software: Son generadas por instrucciones explícitas que afectan al procesador de la misma manera que las interrupciones por hardware, generalmente usadas para hacer llamadas a funciones del sistema operativo

Cuando hay múltiples interrupciones puede que al atender una se inhabiliten las otras, por lo que las interrupciones se manejan en orden secuencial, o que se hayan definido prioridades, por lo que una interrupción de prioridad más alta puede interrumpir a un gestor de interrupción de prioridad menor, cuando se ha gestionado la de mayor prioridad el procesador vuelve a las interrupciones previas, y terminadas todas las rutinas de interrupción se retorna a la ejecución normal.

7. Describa las limitaciones existentes al paralelismo a nivel de instrucciones.

   El paralelismo a nivel de instrucciones es una técnica que permite que múltiples instrucciones se ejecuten en paralelo dentro de un procesador. Algunas de sus limitaciones son:

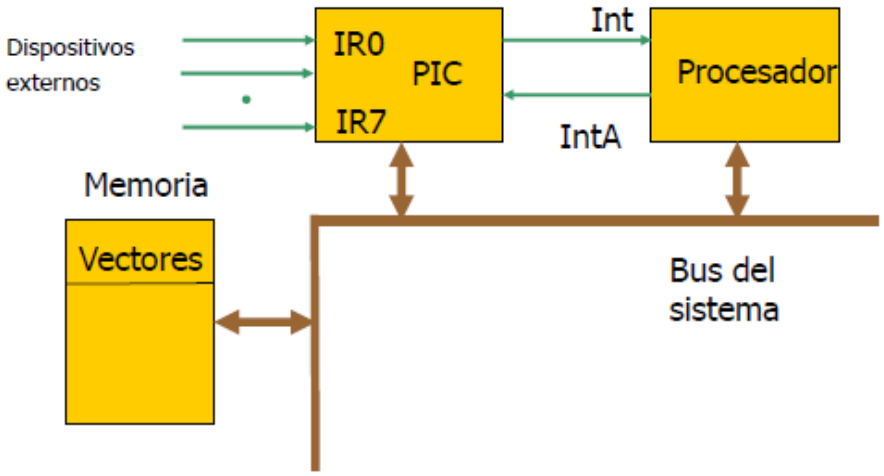
   - Dependencias de Datos: Las dependencias de datos ocurren cuando una instrucción depende de los resultados de una o más instrucciones anteriores. Esto puede limitar la cantidad de instrucciones que pueden ejecutarse en paralelo, ya que algunas de ellas deben esperar a que se completen las instrucciones precedentes.
   - Dependencias de Control: Las dependencias de control se refieren a las instrucciones condicionales, como las instrucciones de salto condicional. El procesador no puede prever de antemano cuál será la próxima instrucción a ejecutar hasta que se resuelva la condición.
   - Limitaciones de Ancho de Bande de Memoria: El acceso a memoria principal es una operación que puede restringir el paralelismo a nivel de instrucción. Si el procesador necesita acceder repetidamente a la memoria, es posible que deba esperar a que se completen las operaciones de

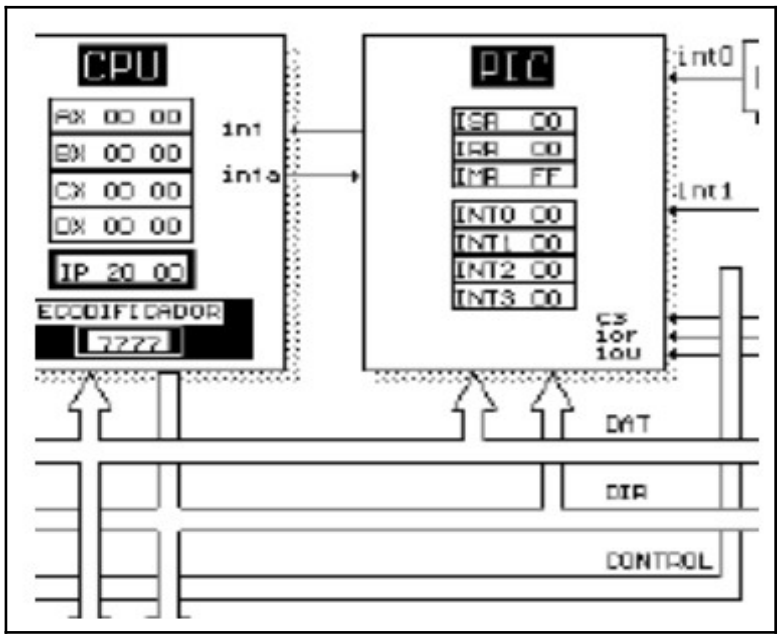
lectura/escritura antes de poder avanzar con otras instrucciones.

- Tamaño de Ventana de Ejecución: La cantidad máxima de instrucciones que se pueden emitir y ejecutar en paralelo puede ser limitada por cuestiones de diseño de hardware o por la arquitectura del procesador.
- Saltos y Llamadas a Subrutinas: Las instrucciones de salto y las llamadas a subrutinas cambian el flujo de ejecución y pueden requerir la cancelación de instrucciones en vuelo que ya no son válidas debido al salto. Esto puede llevar a penalizaciones de tiempo significativas.

8. ¿A que método de atención lo conocemos como de interrupciones vectorizadas?¿Cuándo, por qué, para qué y cómo utiliza una de las denominadas interrupciones por software?
   En el método de interrupciones vectorizadas, cuando se produce una interrupción, se utiliza un vector de interrupción para direccionar directamente la ejecución del procesador a la ubicación de memoria de la rutina de manejo de interrupciones correspondiente. Se utiliza el número de identificación de la interrupción como índice para buscar en la tabla de vectores, luego el procesador salta directamente a esa dirección de memoria y comienza a ejecutar la rutina de manejo de la interrupción.

9. Esquematice y describa la estructura interna de un controlador programable de interrupciones y describa cómo funciona la gestión E/S programada con espera de respuesta





Componentes de la estructura interna de un controlador programable de interrupciones:

-IRR (Interrupt Request Register): Registro de petición de interrupción, indica con bit en 1 las interrupciones demandadas hasta el momento.

-ISR (In Service Register): Registro de interrupción en servicio, indica con bit en 1 cuál es la interrupción que está siendo atendida.

-IMR (Interrupt Mask Register): Registro de máscara de interrupciones, permite el enmascaramiento selectivo de cada una de las entradas de interrupción, indicando con bit en 1 (Indica cuáles deben ser ignoradas). Tras un reset los bits de este registro quedarán en 0.

-EOI (End of Interruption): Fin de interrupción. Como consecuencia, se pone en 0 el bit del ISR correspondiente. Sirve

para indicarle al PIC que ya fue atendida la interrupción y la CPU vuelve a la ejecución normal del programa.
-INT0 ... INT7: 8 registros, donde se cargan los id de interrupciones.

La gestión de E/S programada con espera de respuesta se refiere a un método en el que un programa solicita una operación de E/S y luego espera activamente hasta que se complete la operación. Esto se utiliza en situaciones en las que el programa necesita el resultado de la operación antes de continuar su ejecución. El funcionamiento de este proceso es el siguiente:

1. Solicitud de E/S: El programa emite una solicitud de E/S al dispositivo periférico o al controlador correspondiente.
2. Inicio de la operación: El controlador de E/S inicia la operación y se pone en marcha para realizar la tarea solicitada.
3. Espera activa: Mientras la operación de E/S está en curso, el programa entra en un bucle de espera activa (verifica continuamente el estado de la operación).
4. Finalización de la operación: Cuando la operación de E/S se completa, el controlador de E/S indica su finalización al programa. Esto podría ser a través de una interrupción por hardware o mediante el establecimiento de una bandera de estado.
5. Procesamiento de datos: Una vez que el programa recibe la notificación de que la operación de E/S ha finalizado, puede procesar los datos obtenidos de la operación, si es necesario, y luego continuar con su ejecución normal.

Segmentación de Cauce

1. ¿Qué es segmentación de cauce?
   La segmentación de cauce (pipelining) 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.

2. ¿Qué ventajas proporciona su implementación?
   La implementación de la segmentación de cauce proporciona varias ventajas, entre ellas:
   - Mejora el rendimiento → La ejecución en paralelo acelera la velocidad de ejecución de las instrucciones.
   - Aprovechamiento de recursos → Mientras una unidad de ejecución realiza una operación, las etapas anteriores y posteriores pueden estar ocupadas con otras instrucciones, lo que permite un uso más completo de las unidades funcionales.
   - Mayor paralelismo → Al permitir que múltiples instrucciones se ejecuten al mismo tiempo, la segmentación de cauce aumenta el nivel de paralelismo a nivel de instrucciones en un procesador.
   - Reducción del ciclo de reloj por instrucción → Reduce el tiempo necesario para ejecutar una instrucción completa al dividirla en etapas más pequeñas.

- Mejora de la predicción de saltos → Puede combinarse con técnicas de predicción de saltos para reducir las penalizaciones por fallos de predicción de saltos, lo que resulta en una ejecución más eficiente en las instrucciones.

3. Tres motivos de retardo de cauce en segmentación de cauce

   Hay tres motivos comunes de retardo de cauce en segmentación de cauce:

   1- Dependencia de Datos: Ocurre cuando una instrucción depende de los resultados de una instrucción anterior que aún no ha completado su etapa de ejecución. Los tipos de dependencia de datos son:
      a. RAW – Read after Write
      b. WAR – Write after Read
      c. WAW – Write after Write

   2- Dependencia de Control: Ocurren debido a las decisiones de salto condicional en el flujo de control del programa. Cuando una instrucción de salto condicional se encuentra en el cauce, el procesador puede no conocer la dirección de la próxima instrucción hasta que se resuelva la condición.

   3- Dependencias Estructurales: Ocurren cuando dos o más instrucciones compiten por el mismo recurso funcional en una etapa específica del cauce.

   Para mitigar estos retrasos y mejorar el rendimiento en la segmentación de cauce, se utilizan técnicas como la ejecución fuera de orden, el renombramiento de registros y la predicción de saltos avanzada.

4. Describa las dependencias de los datos que pueden afectar un cauce segmentado.

   Los tipos de dependencias de datos son:

   - Read After Write (RAW): Ocurre cuando una instrucción en el cauce necesita leer un registro o dato que está siendo escrito por una instrucción anterior en el mismo cauce.
   - Write After Read (WAR): Ocurre cuando una instrucción en el cauce necesita escribir un registro o dato que está siendo

leído por una instrucción anterior en el mismo cauce. A veces se denominan “antidependencias”.

- Write After Write (WAW): Ocurre cuando dos instrucciones en el cauce necesitan escribir en el mismo registro o dato. A veces se denominan “dependencias de salida”.

5. Explique los atascos producidos por saltos

   Los atascos producidos por saltos son situaciones en las que la ejecución de instrucciones en un cauce segmentado se ve afectado negativamente debido a la presencia de instrucciones de salto condicional o incondicional. Hay tres tipos principales de atascos de control causados por saltos:

   - Salto condicional tomado, cuando la condición se resuelve como verdadera, lo que significa que se debe tomar el salto. Las instrucciones que siguen al salto condicional y ya han avanzado deben descartarse, lo que conduce a un desperdicio de ciclos de reloj y una penalización en el rendimiento.
   - Salto condicional no tomado, cuando la condición se resuelve como falsa, lo que significa que no se toma el salto. Las instrucciones que siguen al salto deben continuar su ejecución.
   - Salto incondicional, ocurre cuando se encuentra una instrucción de salto incondicional. Las instrucciones que siguen ya han avanzado en el cauce y se encuentra en diferentes etapas de ejecución. Cuando se toma el salto, estas instrucciones en etapas posteriores deben ser descartadas, lo que resulta en un desperdicio de ciclos de reloj y una penalización en el rendimiento.

6. Describa el problema y posibles soluciones ante riesgos por transferencia de control de programa.

   Los riesgos por transferencia de control de programa se refiere a situaciones en las que la ejecución de instrucciones en un procesador segmentado se ve afectada negativamente debido a saltos de programa, como instrucciones de salto condicional o incondicional. Estos riesgos pueden dar lugar a retrasos en la

ejecución de instrucciones y pueden afectar el rendimiento general del procesador.
Los riesgos por transferencia de control pueden ser:

- Salto condicional tomado
- Salto condicional no tomado
- Salto incondicional

Las posibles soluciones:

- Adelantar la resolución de los saltos a la etapa de decodificación
    - En ella se decodifican y se sabe que es un salto
    - Se puede evaluar la condición de salto (con restador)
    - Se puede calcular la dirección de salto (con sumador)
- Para tratamiento de saltos hay:
    - Técnica Hardware → Predicción de saltos:
        - Técnicas estáticas:
            - Predecir que nunca se salta: Asume que el salto no se producirá y siempre capta la siguiente instrucción.
            - Predecir que siempre se salta: Asume que el salto se producirá y siempre capta la instrucción destino del salto.
        - Técnicas dinámicas:
            - Conmutador saltar/no saltar: Basado en la historia de las instrucciones. Es eficaz en bucles.
            - Tabla de historia de saltos (Branch-target buffer): Pequeña cache asociada a la etapa de búsqueda (F). Tiene 3 campos, dirección de una instrucción de bifurcación, información de la instrucción destino y N bits de estado (historia de uso).
        - Predecir según el código de operación:
            - Hay instrucciones con más probabilidades de saltar.
        - Flujos múltiples:

        - Varios cauces (uno por cada opción de salto). Precaptan cada salto en diferentes cauces. Se debe utilizar el cauce correcto.
    - Precaptar el 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:
        - Es una memoria muy rápida, gestionada por la etapa de captación de instrucción del cauce. Comprueba el buffer antes de hacer la captación de memoria.

- Técnica Software → Salto retardado o de relleno de ranura de retardo:
    - El compilador introduce instrucciones que se ejecutarán en cualquier caso después de la instrucción de salto, y de no ser posible se utilizan instrucciones NOP.
    - Requiere reordenar las instrucciones.

DMA y E/S

1. Describa las características funcionales del Acceso Directo a Memoria (DMA) (Etapas de transferencia).

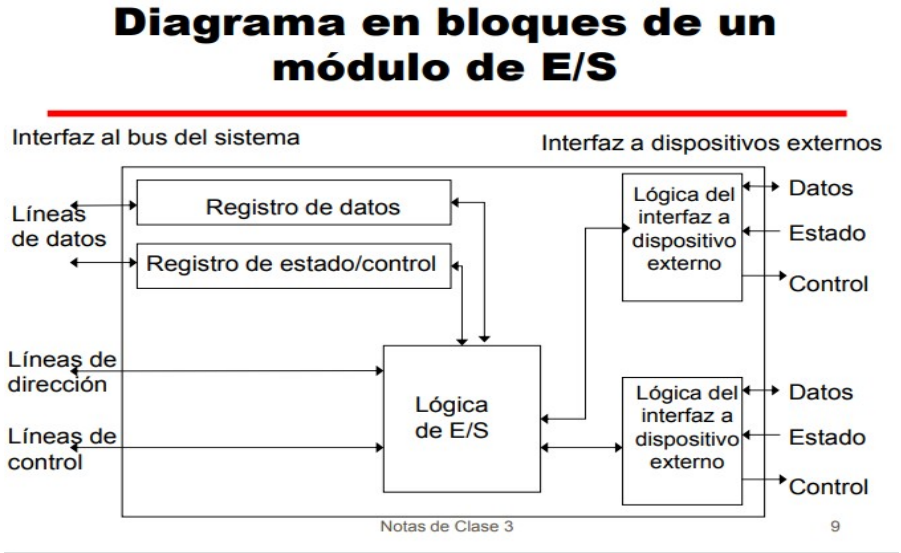
   El Acceso Directo a Memoria permite que un dispositivo periférico acceda a memoria principal (RAM) directamente sin la intervención de la CPU. El DMA acelera la transferencia de datos entre la memoria y los dispositivos periféricos, lo que libera recursos de la CPU para otras tareas. Las características funcionales del DMA incluyen varias etapas de transferencia:

   - Solicitud: La primera etapa implica que un dispositivo periférico envíe una solicitud de acceso a la memoria al controlador DMA. Esta solicitud incluye información sobre la dirección de memoria de origen y destino, la cantidad de datos a transferir y el sentido de la transferencia.
   - Selección del Canal DMA: Los sistemas informáticos pueden tener múltiples canales, cada uno dedicado a un tipo específico de dispositivo o función. En esta etapa, el sistema debe asignar el canal DMA a la solicitud entrante.
   - Configuración: Una vez seleccionado el canal DMA, se configura para que coincida con los requisitos de la transferencia de datos. Esto incluye la configuración de las direcciones de inicio y finalización en la memoria, el tamaño de la transferencia y otras características relevantes.
   - Acceso Directo: El controlador DMA se comunica directamente con la memoria principal y el dispositivo periférico para iniciar la transferencia de datos. La CPU no participa en la transferencia en sí, lo que permite que la CPU realice otras tareas mientras se lleva a cabo la transferencia.
   - Transferencia de Datos: El controlador DMA transfiere datos entre la memoria y el dispositivo periférico utilizando el canal DMA configurado previamente. La transferencia puede ser en una dirección o bidireccional.

- Finalización: Cuando se completa la transferencia, el controlador DMA notifica al dispositivo periférico y actualiza cualquier estado relevante. La CPU puede ser notificada de la finalización de la transferencia mediante una interrupción o un mecanismo similar.
- Liberación del Canal DMA: Después de completar la transferencia, el canal DMA se libera para su uso posterior. Esto permite que otros dispositivos o solicitudes utilicen el canal DMA según sea necesario.

2. ¿Cómo es la estructura de un módulo de E/S? (Esquematice y describa)

   Un módulo de E/S es un componente clave que permite la comunicación entre la CPU y dispositivos periféricos. El funcionamiento de un módulo de E/S permite que el procesador vea una amplia gama de dispositivos de una forma simplificada y oculta los detalles de temporización, formatos y electromecánica propios de los dispositivos para que el procesador pueda funcionar únicamente con órdenes de lectura y escritura o abrir y cerrar fichero.



Estructura de un módulo de E/S: El módulo se conecta al resto del computador a través de un conjunto de líneas (por ejemplo, líneas del bus del sistema). Los datos que se transfieren a y desde el módulo se almacenan temporalmente en uno o más registros de datos. Además, puede haber uno o más registros de estado que proporcionan información del estado presente. Un registro de

estado también puede funcionar como un registro de control, para recibir la 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. Éstas son las que utiliza el procesador para proporcionar órdenes al módulo de E/S. El módulo también debe ser capaz de reconocer y generar las direcciones asociadas a los dispositivos que controla. Cada módulo de E/S tiene una dirección única o, si controla más de un dispositivo externo, un conjunto único de direcciones. Por último, el módulo de E/S posee la lógica específica para la interfaz con cada uno de los dispositivos que controla.

3. Describa las posibles técnicas que puede utilizar una CPU para realizar operaciones de E/S.
   Técnicas de gestión de E/S:
   - E/S programada con espera de respuesta: Se produce un intercambio de datos entre la CPU y el módulo. La CPU tiene control directo sobre la operación y debe esperar a que el módulo termine, permaneciendo ociosa.
   - E/S con interrupciones: La CPU no debe esperar a que la tarea finalice, puede seguir procesando, y el módulo de E/S le envía un pedido de interrupción a la CPU cuando está listo. En ese momento, el procesador ejecuta la transferencia de datos y continúa con el procesamiento previo.
   - E/S con acceso directo a memoria (DMA): La transferencia de datos se realiza entre un periférico y la memoria, sin intervención de la CPU, aumentando así el rendimiento de la aplicación. El controlador de DMA (DMAC) actúa como maestro de bus para transferir datos a y desde memoria a través del bus del sistema.

4. Desarrolle cómo es el funcionamiento del DMA y los usos que de él se hacen.
   El DMA requiere un módulo adicional en el bus del sistema. El controlador de DMA recibe el control del sistema cedido por el procesador, para transferir datos a y desde memoria a través del bus del sistema. Para hacerlo, el DMAC debe utilizarlo sólo cuando el procesador no lo necesita, o forzar al procesador a que

suspenda temporalmente su funcionamiento, a esto se lo conoce como robo de ciclo.
Cuando el procesador desea leer o escribir un bloque de datos, envía una orden al módulo de DMA, incluyendo:

- Si se solicita una lectura o una escritura.
- 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 del bus de datos y almacenada por el DMAC en su registro de direcciones.
- El número de palabras a leer o escribir, indicado a través de las líneas de datos y almacenado en el registro de cuenta de datos.

El procesador continúa su trabajo y el DMAC transfiere el bloque completo de datos, palabra a palabra, directamente desde o hacia la memoria. Cuando la transferencia ha terminado, el DMAC envía una señal de interrupción al procesador.

5. Describa cómo funciona la gestión de E/S programada con espera de respuesta.
   Con la E/S programada, 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, y debe esperar hasta que la operación concluya. Si el procesador es más rápido que el módulo de E/S, el procesador desperdicia este tiempo.
   El procesador es el responsable de extraer los datos de la memoria principal en una salida y de almacenarlos en una entrada.
   Cuando el procesador está ejecutando un programa y encuentra una instrucción de E/S, la ejecuta proporcionando una dirección, especificando el módulo de E/S particular y el dispositivo externo, y una orden de E/S. El módulo realizará la acción solicitada y después activará los bit apropiados en el registro de estado de E/S. El procesador es responsable de comprobar periódicamente el estado del módulo de E/S hasta que encuentra que la operación ha terminado.

6. ¿La coherencia de datos de un sistema jerárquico de memoria se ve afectado por el uso de DMA?
   La coherencia de datos puede verse afectada por el uso de DMA, ya que si se realizan operaciones sobre un dato, éste se actualizará en caché y si no se vacía en memoria antes de que un dispositivo intente accederlo, se podría estar utilizando un valor erróneo, y viceversa.

7. ¿Cuáles son los modos de ubicación de los módulos de entrada salida? Describir.
   Los módulos de E/S pueden ser ubicados en direcciones asociadas a la E/S mapeada en memoria y la E/S aislada
   - E/S asignada en memoria (memory-mapped): Los dispositivos de E/S y memoria comparten un único espacio de direcciones. Tendremos una cierta cantidad de direcciones de memoria asignada a las operaciones de E/S, por lo tanto no habrá necesidad de manejar instrucciones distintas.
   - E/S aislada: Tendremos espacios de direcciones separados, es decir que tendremos direcciones de memoria principal y por otro lado tendremos direcciones asociadas a las E/S. Necesitaremos líneas especiales de E/S y de memoria, y órdenes específicas para E/S (conjunto limitado de instrucciones).

Memoria Caché

1. Memoria Caché. Describa el mapeo asociativo por conjuntos.
   El mapeo asociativo por conjuntos hace referencia a que un bloque puede almacenarse en un conjunto restringido de lugares en la caché.
   La caché se divide en v conjuntos, cada uno de k líneas, donde el producto de estos dos valores da el número de líneas de la caché (m), y el módulo entre el número de bloque de memoria principal (j) y los v conjuntos de la memoria caché da como resultado el número de conjunto de la caché (i).
   El número de bloque de memoria depende de la cantidad de palabras que tienen los bloques, y se obtiene con el valor representado por los bits más significativos de la dirección, que son comunes a todas las palabras del bloque.
   La lógica de control de la caché interpreta una dirección de memoria como tres campos: etiqueta, conjunto y palabra.

2. Analice las políticas de escritura desde el punto de vista de la coherencia de datos.
   Las políticas de escritura varían de acuerdo a si el dato está en la caché (acierto) o no (fallo).

   En acierto:
   - Escritura inmediata (write-through): Se actualizan simultáneamente la posición de la caché y de la memoria principal, por lo que se mantiene la coherencia de datos en todo momento. Suele combinarse con la técnica no-write allocate.
   - Post-escritura (write-back): La información sólo se actualiza en la caché y se marca con un bit de “actualizar” o “sucio”. Cuando el bloque es desalojado de la caché este bit se comprueba y, si está activo, se escribe el bloque en memoria principal. Esto puede producir que la memoria principal contenga información errónea durante un tiempo. Suele combinarse con la técnica write allocate.

En fallo:

- Write Allocate: La información se lleva de memoria principal a la caché y se sobrescribe en ella, por lo que se puede alterar la coherencia de datos hasta que haya un reemplazo en memoria principal.
- No-Write Allocate: El bloque no es llevado a memoria caché, se escribe directamente en memoria principal.

3. Describa los elementos a tener en cuenta en el diseño de una memoria caché. Analice ventajas y desventajas de poseer varios niveles de caché.

   Los elementos básicos a tener en cuenta en el diseño de una memoria caché son:

   - El tamaño de la caché → Es imposible predecir un tamaño óptimo, pero se debe tener en cuenta que: cuanto más grande es, mayor es el número de puertas implicadas en direccionar la caché; cachés grandes tienden a ser ligeramente más lentas; el tamaño de caché está limitado por las superficies disponibles de chip y tarjetas.
   - Función de correspondencia → Se necesita un algoritmo que haga corresponder bloques de memoria principal a líneas de caché. La elección de la función de correspondencia determina cómo se organiza la caché. Pueden utilizarse tres técnicas:
     - Directa: consiste en hacer corresponder cada bloque de memoria principal a sólo una línea posible de caché,
     - Asociativa: cada bloque de memoria principal puede cargarse en cualquier línea de la caché, y
     - Asociativa por conjuntos: un bloque puede almacenarse en un conjunto restringido de lugares en la caché.
   - Algoritmo de sustitución → Una vez que se ha llenado la caché, para introducir un nuevo bloque debe sustituirse uno de los bloques existentes. Para las correspondencias asociativas se requieren algoritmos de sustitución:

    - LRU (last-recently used): Se sustituye el bloque que se ha mantenido en la caché por más tiempo sin haber sido referenciado.
    - FIFO (first-in first-out): Se sustituye el bloque del conjunto que ha estado más tiempo en la caché (el primero que ingresó).
    - LFU (least-frequently used): Se sustituye aquel bloque del conjunto que ha experimentado menos referencias.
    - Otra alternativa es la sustitución aleatoria, elegir una línea al azar para ser sustituida.
- Política de escritura → Cuando se debe reemplazar un bloque de la caché, si se ha realizado al menos una operación de escritura sobre una palabra de la línea correspondiente, la memoria principal debe actualizarse rescribiendo la línea de caché en el bloque de memoria.
    - Escritura inmediata: Todas las operaciones de escritura se hacen tanto en caché como en memoria principal.
    - Postescritura: Las actualizaciones se hacen sólo en caché, y se sobrescriben en memoria principal cuando el bloque es sustituido.
- Tamaño de línea → Cuando se recupera y ubica en caché un bloque de datos, se recuperan no sólo la palabra sino algunas adyacentes. Se producen dos efectos:
    - Bloques más grandes reducen el número de bloques que caben en la caché. Dado que cada bloque captado se escribe sobre contenidos anteriores de la caché, un número reducido de bloques da lugar a que se sobrescriban datos poco después de haber sido captados.
    - A medida que un bloque se hace más grande, cada palabra adicional está más lejos de la requerida y por tanto es más improbable que sea necesaria a corto plazo.
- Número de cachés → Hay dos aspectos relacionados con este tema, el número de niveles de caché, y el uso de caché unificada frente al de cachés separadas.

- Cachés multinivel: En general, el uso de un segundo nivel de caché mejora las prestaciones. No obstante, el uso de cachés multinivel complica todos los aspectos de diseño de la caché.
- Caché unificada frente a cachés separadas: Muchos diseños contenían una sola caché para almacenar las referencias tanto a datos como a instrucciones. Más recientemente, se ha hecho normal separar la caché en dos: una dedicada a instrucciones y otra a datos. La ventaja clave del diseño de una caché partida es que elimina la competición por la caché entre el procesador de instrucciones y la unidad de ejecución. Esto es importante en diseños que cuentan con segmentación de cauce de instrucciones.

4. Describa los algoritmos de reemplazo de bloque y políticas de escritura en la cache.

   Algoritmo de sustitución → Una vez que se ha llenado la caché, para introducir un nuevo bloque debe sustituirse uno de los bloques existentes. Para las correspondencias asociativas se requieren algoritmos de sustitución:

   a. LRU (last-recently used): Se sustituye el bloque que se ha mantenido en la caché por más tiempo sin haber sido referenciado.
   b. FIFO (first-in first-out): Se sustituye el bloque del conjunto que ha estado más tiempo en la caché (el primero que ingresó).
   c. LFU (least-frequently used): Se sustituye aquel bloque del conjunto que ha experimentado menos referencias.
   d. Otra alternativa es la sustitución aleatoria, elegir una línea al azar para ser sustituida.

   Política de escritura → Cuando se debe reemplazar un bloque de la caché, si se ha realizado al menos una operación de escritura sobre una palabra de la línea correspondiente, la memoria principal debe actualizarse rescribiendo la línea de caché en el bloque de memoria.

e. Escritura inmediata: Todas las operaciones de escritura se hacen tanto en caché como en memoria principal.
f. Postescritura: Las actualizaciones se hacen sólo en caché, y se sobrescriben en memoria principal cuando el bloque es sustituido.

5. Si se pretende mejorar el tiempo de acceso medio a memoria cache. ¿Sobre qué parámetros será necesario trabajar y propone como medida para hacerlo?

   Para mejorar el tiempo de acceso a la memoria caché es importante trabajar en varios parámetros y considerar diferentes medidas:

   1- Tamaño de la caché (capacidad) → Aumentar el tamaño de la memoria caché permite que más datos se almacenen en ella, lo que reduce la necesidad de acceder a la memoria principal con frecuencia.
   2- Política de reemplazo → Utilizar una política de reemplazo adecuada puede ayudar a mantener los datos más relevantes en la caché y reducir los fallos de caché.
   3- Localidad espacial y temporal → Optimizar los algoritmos y estructuras de datos utilizados en el software para aprovechar la localidad espacial y la localidad temporal.

   Entre otros...

6. Describa y compare las correspondencias entra la memoria principal y la caché. ¿Cómo se determina, en cada caso, la hilera/fila de ubicación de un bloque de palabras provenientes de la memoria principal? Ejemplifique.

   Ya que hay menos líneas de caché que bloques de memoria principal, se necesita un algoritmo que haga corresponderlos a líneas de caché. Además se requiere algún medio para determinar qué bloque de memoria principal ocupa actualmente una línea dada de caché.

   a- Correspondencia directa: Consiste en hacer corresponder cada bloque de memoria principal a solo una línea posible de caché. Se implementa utilizando la dirección de memoria principal, que puede verse dividida en tres campos: w bits

menos significativos identifican cada palabra dentro de un bloque de memoria principal; los s bits restantes especifican uno de los 2^s bloques de la memoria principal; se interpretan estos s bits como una etiqueta de s-r bits (más significativa) y un campo de línea de r bits. Este último campo identifica una de las $m = 2^r$ líneas de la caché.

b- Correspondencia asociativa: Cada bloque de memoria principal puede cargarse en cualquier línea de la caché. La lógica de control de la caché interpreta una dirección de memoria simplemente como una etiqueta y un campo de palabra. El campo de etiqueta identifica unívocamente un bloque de memoria principal.

c- Correspondencia asociativa por conjuntos: La caché se divide en v conjuntos, cada uno de k líneas. La lógica de control de la caché interpreta una dirección de memoria como tres campos: etiqueta, conjunto y palabra. Los d bits de conjunto especifican uno de entre $v=2^d$ conjuntos. Los s bits de los campos de etiqueta y de conjunto se especifican uno de los 2^s bloques de memoria principal.

7. Justifique el uso de dos niveles de caché.

El uso de dos niveles de caché (L1 y L2) es una característica clave en la jerarquía de memoria. Algunos motivos para el uso de dos niveles de caché son:

- Reducción del tiempo de acceso promedio: Permite el acceso rápido a datos de uso común sin la necesidad de acceder a la memoria principal (RAM). El L1 es el más rápido pero el más pequeño y cercano al núcleo del procesador, mientras que el L2 es más grande pero un poco más lento.
- Aprovechamiento de la localidad de referencia: Permite explotar la localidad espacial y temporal, ya que los datos accedidos en el L1 pueden propagarse al L2 si son utilizados repetidamente.
- Optimización de costos y consumo de energía: Permite equilibrar el rendimiento y los recursos disponibles de manera más eficiente. El L1 puede ser diseñado con tecnologías más

avanzadas y costosas, mientras que el L2 puede ser accesible con tecnologías más económicas.

- Reducción de la latencia de acceso a la memoria principal: Cuando un dato no está en el L1 pero sí en el L2, aún se puede acceder a él más rápidamente que si tuviera que recuperarse directamente desde la memoria principal. Esto ayuda a reducir la latencia general de acceso a memoria, mejorando el rendimiento del sistema.
- Mayor escalabilidad: Permite una mayor escalabilidad y rendimiento en sistemas de gama alta sin aumentar significativamente los costos y la complejidad.

RISC y CISC

1. Describa tres características que usted considere las más importantes de las arquitecturas RISC.

   Algunas características de las arquitecturas RISC son:

   - Una instrucción por ciclo: Se ejecuta una instrucción máquina cada ciclo máquina. Deben ejecutarse más rápido ya que no hay que acceder a la memoria de control de microprograma durante la ejecución de la instrucción.
   - Operaciones registro a registro: Simplifica el repertorio de instrucciones y por lo tanto la unidad de control. Además, fomenta la optimización del uso de registros.
   - Modos de direccionamiento sencillos: Esta característica simplifica el repertorio de instrucciones y la unidad de control.
   - Formato de instrucción sencillo: Se suele usar 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 también son fijas. Los formatos sencillos simplifican la unidad de control.

2. Describa las características que diferencian los procesadores RISC respectos a los CISC.
   Las arquitecturas RISC se caracterizan por un conjunto de instrucciones simple, instrucciones de longitud fija y una ejecución más simple, lo que las hace adecuadas para una ejecución más eficiente y predecible.
   Las arquitecturas CISC, por otro lado, tienen un conjunto de instrucciones más complejo y versátil, lo que las hace adecuadas para tareas más variadas pero a costa de una mayor complejidad de hardware y menos previsibilidad en el tiempo de ejecución.

   No existe una clara barrera diferenciadora, muchos diseños incluyen características de ambos criterios. No existe un par de máquinas RISC y CISC directamente comparables, no hay un conjunto de programas de prueba definitivo.

Superescalares y Supersegmentado

1. ¿Qué son los procesadores superescalares?
   Un procesador superescalar es aquél que usa múltiples cauces de instrucciones independientes. Cada cauce consta de múltiples etapas, de modo que puede tratar varias instrucciones a la vez, por lo que saca provecho del paralelismo en las instrucciones. Capta varias instrucciones a la vez, y a continuación intenta encontrar instrucciones cercanas que sean independientes entre sí y puedan ejecutarse en paralelo.

2. ¿De qué depende el paralelismo de una máquina superescalar?
   Las limitaciones del paralelismo son:

- Dependencia de datos verdadera: Cuando una instrucción necesita leer datos que son el resultado de una instrucción previa.
- Dependencia relativa del procesamiento: 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: Es una pugna de dos o más instrucciones por el mismo recurso al mismo tiempo.
- Dependencia de salida: Se produce cuando dos instrucciones tratan de escribir en un mismo registro.
- Antidependencia: Cuando una instrucción intenta escribir datos antes de que otra instrucción previa haya terminado de leer esos mismos datos.

3. ¿Cuál es el objetivo de usar 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 resolver conflictos de dependencia de almacenamiento de datos y permitir una ejecución más eficiente y paralela de instrucciones.

4. ¿Qué características tienen los procesadores superescalares?
   Las características principales de los procesadores superescalares son:
   - Estrategias de captación simultánea de múltiples instrucciones (se pueden llevar a cabo más de una instrucción simultáneamente).
   - Lógica para determinar dependencias verdaderas entre valores de registros y mecanismos para comunicar esos valores.
   - Mecanismos para iniciar o emitir múltiples instrucciones en paralelo.
   - Recursos para la ejecución en paralelo de múltiples instrucciones (Conlleva la duplicación de algunas o todas las partes de la CPU/ALU).

- Mecanismos para entregar el estado del procesador en un orden correcto.

5. ¿Qué características 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, se dobla la velocidad de reloj interna, lo que permite la realización de dos tareas en un ciclo de reloj externo. Por otro lado, un procesador superescalar puede ejecutar instrucciones en diferentes cauces de manera independiente y concurrente.

6. Compare las políticas de emisión de instrucciones.
   Políticas de emisión de instrucciones:
   - 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 resultado en ese mismo orden.
   - Emisión en orden y finalización desordenada → Las instrucciones se pueden completar en cualquier orden una vez que han sido ejecutadas, siempre y cuando no altere el resultado final. La emisión se para cuando hay cualquier tipo de conflicto o dependencia de datos.
   - Emisión desordenada y finalización desordenada → Las instrucciones se emiten a la etapa de ejecución en el orden en que están listas para ser ejecutadas, y se pueden completar en cualquier orden, siempre y cuando no altere el resultado final.

7. Elija una alternativa de emisión/finalización y justifique.

BUSES

1. ¿Qué es un bus? Describa los diferentes tipos de modos de arbitraje y sincronización.

   Un bus es un camino de comunicación entre dos o más dispositivos. Se trata de un medio de transmisión compartido, se conectan varios dispositivos y cualquier señal transmitida por uno de ellos está disponible para que los otros puedan acceder a ella.

   Solo un dispositivo puede transmitir con éxito en un momento dado, por lo que se necesitan algún método de arbitraje para hacerlo:

   - Centralizado: Un único dispositivo hardware (controlador del bus o árbitro) es responsable de asignar tiempos en el bus.
   - Distribuido: No existe un control central, cada módulo dispone de lógica para controlar el acceso y los módulos actúan conjuntamente para compartir el bus.

   • El objetivo de ambos es designar un dispositivo (el procesador o un módulo de E/S) como maestro del bus, el cual puede iniciar una transferencia de datos con otro dispositivo que actúa como esclavo.

   La temporización o sincronización es la forma en la que se coordinan los eventos del bus.

   • Sincrónica: La presencia de un evento en el bus está determinada por un reloj. El bus incluye una línea de reloj a través de la que se transmite una secuencia en la que se alternan intervalos regulares de igual duración a uno y a cero. Un único intervalo a uno seguido de otro a cero se conoce como ciclo de reloj o ciclo de bus y define un intervalo de tiempo unidad (time slot).
   • Asincrónica: La presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo.

- La temporización sincrónica es más fácil de implementar y comprobar, pero es menos flexible ya que todos los dispositivos deben usar la misma frecuencia de reloj.

2. Mencione las principales diferencias entre un bus PCI y SCSI.

   Bus PCI (Peripheral Component Interconnect - Interconexión de Componente Periférico) → Es un bus de ancho de banda elevado, independiente del procesador, que se puede utilizar como bus de periféricos o bus para una arquitectura de entreplanta.
   Bus SCSI (Small Computer System Interface - Pequeña interfaz del sistema de cómputo) → Es una interfaz estándar para la transferencia de datos entre distintos dispositivos del bus de la computadora. Se utiliza para comunicar dispositivos rápidos, como discos CD-ROM, dispositivos de audio y dispositivos de almacenamiento externo de datos. Requiere un controlador de interfaz.

3. ¿Qué elementos característicos definen un bus?

   Los elementos claves de un bus son:

   1- Tipos de buses: Las líneas del bus pueden ser dedicadas (está permanentemente asignada a una función o a un subconjunto físico de componentes del computador) o multiplexadas (utiliza las mismas líneas para usos diferentes).
   2- Método de arbitraje: Puede ser centralizado (un único dispositivo hardware es responsable de asignar tiempos en el bus) o distribuido (cada módulo dispone de lógica para controlar el acceso y los módulos actúan conjuntamente para compartir el bus).
   3- Temporización: Es la forma de coordinar los eventos en el bus, puede ser sincrónica (la presencia de un evento en el bus está determinada por un reloj) o asincrónica (la presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo).
   4- Anchura del bus: Está determinada por la cantidad de líneas que tiene el bus. En un bus de datos, la anchura determina cuántos bits se pueden transferir al mismo tiempo. En un bus de direcciones, la anchura del bus determina la máxima capacidad de memoria posible en el sistema.

5- Tipo de transferencia de datos: Todos los buses permiten tanto transferencias de escritura (dato de maestro a esclavo) como de lectura (dato de esclavo a maestro). En ciertos buses también son posibles algunas operaciones combinadas, como la lectura-modificación-escritura (lectura seguida inmediatamente de una escritura en la misma dirección), o la lectura-después-de-escritura (una escritura seguida inmediatamente de una lectura en la misma dirección). Algunos buses permiten también transferencias de bloques de datos.

4. Tipos de buses, temporización y métodos de arbitraje

Las líneas de un bus pueden ser:

- Dedicadas: Está permanentemente asignada a una función o a un subconjuntos físico de componentes del computador.
- Multiplexadas: Utiliza las mismas línea para usos diferentes, por ejemplo se transmiten la información de dirección y datos a través del mismo conjunto de líneas utilizando una línea de control de Dirección Válida.
  La ventaja es el uso de menos líneas (ahorra espacio y costes).
  La desventaja es que necesita una circuitería más compleja en cada módulo.

Solo un dispositivo puede transmitir con éxito en un momento dado, por lo que se necesitan algún método de arbitraje para hacerlo:

- Centralizado: Un único dispositivo hardware (controlador del bus o árbitro) es responsable de asignar tiempos en el bus.
- Distribuido: No existe un control central, cada módulo dispone de lógica para controlar el acceso y los módulos actúan conjuntamente para compartir el bus.

• El objetivo de ambos es designar un dispositivo (el procesador o un módulo de E/S) como maestro del bus, el cual puede iniciar una transferencia de datos con otro dispositivo que actúa como esclavo.

La temporización es la forma en la que se coordinan los eventos del bus.

- Sincrónica: La presencia de un evento en el bus está determinada por un reloj. El bus incluye una línea de reloj a través de la que se transmite una secuencia en la que se alternan intervalos regulares de igual duración a uno y a cero. Un único intervalo a uno seguido de otro a cero se conoce como ciclo de reloj o ciclo de bus y define un intervalo de tiempo unidad (time slot).
- Asincrónica: La presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo.

- La temporización sincrónica es más fácil de implementar y comprobar, pero es menos flexible ya que todos los dispositivos deben usar la misma frecuencia de reloj.

TAXONOMÍA DE FLYNN

1. ¿Qué son los MIMD de la taxonomía de Flynn?
   La Taxonomía de Flynn es una clasificación utilizada para categorizar las arquitecturas de computadoras según la cantidad de instrucciones y datos que pueden procesar simultáneamente.

Una de las categorías es MIMD (Multiple Instruction, Multiple Data), donde múltiples instrucciones pueden ser ejecutadas de manera simultánea en diferentes unidades de procesamiento o núcleos, y procesan múltiples conjuntos de datos simultáneamente.

2. Describa las 4 variantes de arquitectura de la Taxonomía de Flynn.
   La Taxonomía de Flynn clasifica las arquitecturas de computadores en cuatro categorías principales en función de cómo manejan las instrucciones y los datos.
   - SISD (Single Instruction, Single Data): Una sola unidad de procesamiento ejecuta una única secuencia de instrucciones que opera en un único conjunto de datos en cada ciclo de reloj.
   - SIMD (Single Instruction, Multiple Data): Una sola instrucción se aplica simultáneamente a múltiples conjuntos de datos. Esto significa que múltiples unidades de procesamiento ejecutan la misma instrucción en diferentes conjuntos de datos al mismo tiempo.
   - MISD (Multiple Instruction, Single Data): Múltiples secuencias de instrucciones se ejecutan en paralelo en un solo conjunto de datos.
   - MIMD: (Multiple Instruction, Multiple Data): Cuentan con múltiples unidades de procesamiento, cada una ejecutando su propia secuencia de instrucciones en su propio conjunto de datos de manera independiente. Esto permite la ejecución de programas diferentes o partes diferentes de un programa en paralelo.

CLUSTERS

1. Describa las características que diferencian los SMP respecto a los Cluster.

   Los SMP se caracterizan por su arquitectura de memoria compartida, donde múltiples procesadores comparten el mismo espacio de memoria y recursos, brindando equidad en el acceso a los recursos sin jerarquía. La comunicación entre procesadores es directa a través de la memoria compartida, lo que facilita la eficiencia en el intercambio de datos.
   Por otro lado, los Clusters se componen de nodos independientes interconectados mediante una red, cada uno con su propia memoria y recursos. La comunicación entre nodos se realiza a través de la red, lo que puede introducir latencia. Los Clusters ofrecen escalabilidad horizontal al permitir la adición de nodos para aumentar la capacidad de procesamiento, adecuándose a aplicaciones distribuidas.

   La principal ventaja de un SMP es que resulta más fácil de gestionar y configurar que un Cluster, además necesita menos espacio físico y consume menos energía, y son plataformas estables y bien establecidas.
   Los Clusters son superiores en términos de escalabilidad absoluta e incremental, y además también son superiores en términos de disponibilidad, puesto que todos los componentes del sistema pueden hacerse altamente redundantes.

2. Características de los Cluster.

   Se puede definir un Cluster como 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. Cada computador del Cluster se denomina nodo. Sus principales características son:

- Escalabilidad absoluta: Es posible configurar Clusters grandes que incluso superan las prestaciones de los computadores independientes más potentes.
- Escalabilidad incremental: Un Cluster se configura de forma que sea posible añadir nuevos sistemas a él en ampliaciones sucesivas.
- Alta disponibilidad: Puesto que cada nodo es un computador autónomo, el fallo de uno de los nodos no significa la pérdida del servicio.
- 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.

3. Compare los sistemas MP y Cluster.

   Los sistemas MP utilizan memoria compartida al igual que los SMP, pero pueden asignar roles específicos a los procesadores, introduciendo variabilidad en la equidad de acceso a los recursos. La comunicación entre procesadores puede ser más compleja debido a responsabilidades específicas de cada procesador.
   Por otro lado, los Clusters se componen de nodos independientes interconectados mediante una red, cada uno con su propia memoria y recursos. La comunicación entre nodos se realiza a través de la red, lo que puede introducir latencia. Los Clusters ofrecen escalabilidad horizontal al permitir la adición de nodos para aumentar la capacidad de procesamiento, adecuándose a aplicaciones distribuidas.

4. Funcionamiento de un Cluster.

   El funcionamiento de un Cluster implica la coordinación y utilización de múltiples nodos o computadoras interconectadas para trabajar juntas como una única entidad de procesamiento. Para gestionar y coordinar las operaciones de los nodos se utiliza software de gestión de Clusters, que supervisa el estado de los nodos, administra las tareas y permite la comunicación entre ellos.
   Las tareas o cargas de trabajo se distribuyen entre los nodos del Cluster. Esto puede hacerse de diferentes maneras, dependiendo

de la aplicación y del software utilizado. Implementa estrategias de balance de cargas que distribuyen las tareas de manera uniforme entre los nodos disponibles.
Los nodos del Cluster se comunican entre sí a través de la red de interconexión. Esto puede incluir la transferencia de datos, mensajes de control o la coordinación de tareas.
Incorporan mecanismos de tolerancia de fallos para garantizar la disponibilidad continua de los servicios. Si un nodo falla, las tareas pueden reasignarse a otros nodos en funcionamiento para evitar la interrupción del servicio.

MAYOR DESARROLLO

1- En un cause segmentado, con secuencia de instrucciones independientes ¿Qué consecuencias trae el paso de una instrucción de salto? Analice los casos de salto incondicional y de salto condicional. Mencione que posibles soluciones se pueden aplicar para evitar o disminuir las consecuencias.

2- Describa cómo se debe implementar la estructura de pila de un procesador tipo RISC cuyos registros son genéricos (basarse en el procesador MIPS) ¿Cómo se deberá trabajar anidamiento de procedimientos/funciones?

3- Describa las funciones que se utilizan en la política de ubicación de bloques en memoria Cache. Analice las políticas de escritura de datos desde el punto de vista de la coherencia de los mismo en la jerarquía.

4- ¿Qué características definen un procesador como superescalar? Describa las políticas de emisión de instrucciones en un cause segmentado.

5- ¿Cuáles son las arquitecturas que pueden encontrarse en la configuración MIMD de la taxonomía de Flynn?