Preguntas Final de Arqui

INTERRUPCIÓN (software y hardware)

- **¿Cuál es el mecanismo de interrupción?**

RTA: El mecanismo de interrupción permite a un dispositivo o evento externo pausar la ejecución normal de un programa para atender una tarea urgente. El procesador interrumpe el flujo del programa actual, guarda su estado (contexto), ejecuta una rutina especial conocida como rutina de servicio de interrupción (ISR) y luego restaura el contexto para continuar con el programa.

- **¿Cómo trabajan las interrupciones?**
- **¿Cuáles son los pasos básicos en el manejo de una interrupción?**

RTA: Las interrupciones permiten a los dispositivos o eventos externos comunicarse con el procesador de manera eficiente. Estas señales, enviadas por hardware o software, se revisan al final de cada ciclo de instrucción. Cuando ocurre una interrupción:

- Ocurrencia de la interrupción: El hardware o software genera una señal de interrupción.
- Reconocimiento/Notificación**:** El procesador detecta la interrupción y verifica su prioridad (si hay interrupciones múltiples).
- Guardar contexto: El estado actual del programa (como registros y contador de programa) se guarda en la pila.
- Ejecutar ISR**:** El procesador transfiere el control a la rutina de servicio de interrupción (ISR).
- Restaurar contexto: Una vez atendida la interrupción, el procesador restaura el estado guardado.
- Reanudar programa**:** El programa principal continúa desde donde fue interrumpido.

- **Explique características de las interrupciones.**

RTA:

- Sincrónicas: Generadas por el procesador debido a eventos como errores de hardware o fallas en el programa.
- Asincronía: Las interrupciones pueden ocurrir en cualquier momento, independientemente del flujo actual del programa.
- Clasificación:
  - **Interrupciones de hardware**: Provienen de dispositivos externos, como teclados o temporizadores.
  - **Interrupciones de software**: Generadas por instrucciones específicas (e.g., interrupciones int en ensamblador).
- Prioridad: Algunas interrupciones tienen mayor prioridad que otras y deben ser atendidas primero.
- Habilitación/Deshabilitación: El procesador puede habilitar o deshabilitar las interrupciones según el contexto.
- Vectores de interrupción: Cada interrupción está asociada a una dirección de memoria específica donde reside su ISR.
- Tiempo de respuesta rápido: El objetivo es atender eventos críticos lo más rápido posible.

- **Tratamiento de interrupciones múltiples.**

RTA:

- Prioridad de Interrupciones: Cada interrupción tiene un nivel de prioridad asignado.
  Si ocurre una interrupción de mayor prioridad mientras se atiende otra, el procesador puede pausar la ISR actual para manejar la de mayor prioridad (interrupción anidada).
- Registro de Enmascaramiento: El procesador puede enmascarar (ignorar) ciertas interrupciones mientras atiende otras. Ejemplo: El bit de habilitación de interrupciones global permite deshabilitar todas las interrupciones temporalmente.
- Cola de Interrupciones: Las interrupciones pendientes se colocan en una cola y se manejan en orden de prioridad.

- **¿Qué tipos de interrupciones existen?**

RTA: Existen principalmente dos tipos de interrupciones:

- Interrupciones por hardware:
  - Son generadas por dispositivos externos como teclados, temporizadores o discos.
  - Permiten al procesador atender eventos en tiempo real (e.g., un teclado enviando una tecla presionada).
- Interrupciones por software:
  - Son generadas por instrucciones específicas o eventos internos del procesador.
  - Ejemplo: fallos en el programa, como errores de división por cero o instrucciones especiales como INT en ensamblador.

- **¿Por qué es importante el mecanismo de interrupciones?**

RTA: El mecanismo de interrupciones es esencial porque permite:

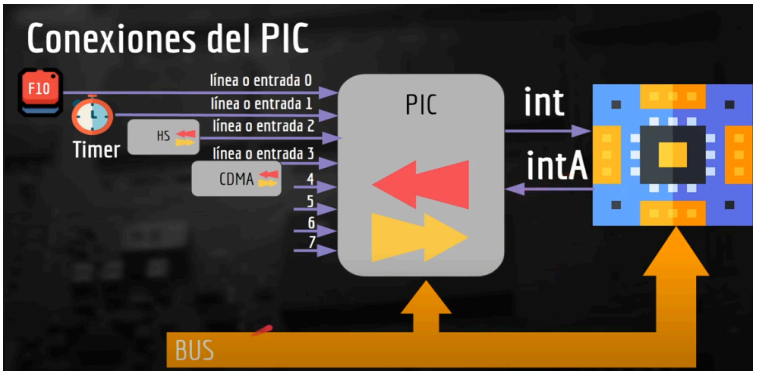
- Atender eventos en tiempo real sin necesidad de que el procesador esté constantemente monitoreando dispositivos (polling).
- Optimizar el uso del procesador al liberar recursos cuando no hay eventos urgentes.
- Gestionar errores o fallas, como problemas de hardware o software, de manera inmediata.

- **¿Cual es la función de un controlador de interrupciones?**

RTA: Un PIC es un dispositivo utilizado para gestionar y priorizar múltiples fuentes de interrupciones de hardware, combinándolas sobre una o más líneas hacia el CPU.
El PIC puede manejar hasta 8 solicitudes de interrupción independientes de manera simultánea, numeradas de INT0 a INT7. Cuando se producen varias solicitudes al mismo tiempo, el PIC determina cuál enviar al procesador según un esquema de prioridad predefinido: INT0 tiene la prioridad más alta, mientras que INT7 tiene la más baja.
Además, el PIC traduce estas solicitudes de interrupción en IDs de interrupción que el CPU utiliza para localizar el vector de interrupción correspondiente en la tabla de vectores. Esto permite que el sistema atienda cada interrupción y ejecute la subrutina adecuada.
El procesador accede al PIC mediante instrucciones de entrada/salida, como `IN` y `OUT`, utilizando estas direcciones de puertos para configurar o interactuar con el PIC.

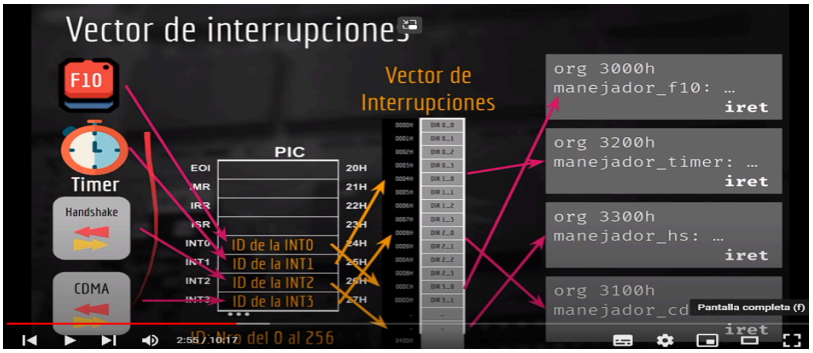
Estructura interna:

-20H - EOI: 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.
-21H - IMR: 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.
-22H - IRR: Registro de petición de interrupción, indica con bit en 1 las interrupciones demandadas hasta el momento.
-23H - ISR: Registro de interrupción en servicio, indica con bit en 1 cuál es la interrupción que está siendo atendida.
-INT0 … INT7: 8 registros, donde se cargan los id/identificadores de int.



- **¿A qué método de atención lo conocemos como "interrupciones vectorizadas" ?**

RTA: El vector de interrupciones es un sector de la memoria (normalmente ubicado en los primeros 1024 bytes, de 0000h a 0400h) donde se almacenan las direcciones de las subrutinas que manejan las interrupciones. Cuando se produce una interrupción específica (por ejemplo, el TIMER), el PIC utiliza el registro **INT** correspondiente a esa interrupción (INT0 … INT7) para determinar su **ID de interrupción**. (ID entre el 0 y el 255)
Este ID se multiplica por 4 (debido a que cada entrada ocupa 4 bytes) para obtener la dirección exacta en la tabla de vectores donde se encuentra la dirección de la subrutina que manejará dicha interrupción.
Nota: Algunos IDs de interrupción (como 0, 6 y 7) no están disponibles para su uso porque están reservados para interrupciones por software o excepciones del sistema.



- **¿Cuándo, por qué, y cómo utiliza una de las denominadas interrupciones por software?**

RTA: Las interrupciones por software se utilizan principalmente para realizar llamadas a funciones del sistema operativo o acceder a servicios del sistema. Son necesarias porque algunos procesadores tienen instrucciones específicas que permiten generar una interrupción de software, lo que cambia el flujo de ejecución al sistema operativo para que realice operaciones que no pueden ser gestionadas directamente por el programa. Esto se utiliza, por ejemplo, para manejar acceso a archivos, gestión de memoria, y control de dispositivos.

CAUSE

- **¿Qué es segmentación de cauce?¿Qué ventajas proporciona su implementación?**

RTA: 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 etapas que permitan una ejecución simultánea.
Las instrucciones pasan por varios estados en el proceso de ejecución, por lo tanto, varias instrucciones pueden ser procesadas simultáneamente (cada una en estados distintos)
VENTAJAS:
Mayor aprovechamiento del procesador: Múltiples instrucciones se procesan simultáneamente, evitando tiempos muertos en la CPU.
Modularidad: Dividir el procesamiento en etapas facilita el diseño y depuración del hardware.
Incremento en el rendimiento global: Aumenta la tasa de instrucciones ejecutadas por unidad de tiempo (Throughput), aunque no acelera cada instrucción individual.
Eficiencia energética: Permite aprovechar mejor los recursos físicos.

Tareas a realizar por ciclo
- búsqueda. IF - decodificación. DE - ejecución. EX - acceso a memoria. ME - almacenamiento. W

- **¿Cuánto mejora el rendimiento?**

RTA: La segmentación (pipeline) permite la ejecución simultánea de varias instrucciones en distintas etapas del cauce, lo que incrementa significativamente la productividad general del procesador. Aunque no reduce el tiempo necesario para ejecutar una instrucción individual, sí disminuye el tiempo promedio por instrucción al aprovechar mejor los recursos del hardware. Ejemplo:
-Supongamos que un procesador sin segmentación (cauce) tarda 5 ciclos de reloj por instrucción. Para ejecutar 4 instrucciones, necesitaría:
Tiempo total (sin cauce)=5 ciclos × 4 instrucciones =20 ciclos / 4 = 5
-Con segmentación, cada etapa del cauce se encarga de una parte de la instrucción, y las instrucciones se solapan. Después de un tiempo inicial (de llenado), se completa una instrucción por ciclo. Por lo tanto:
Tiempo total (con cauce)=5 ciclos (llenado) + 3 ciclos adicionales (3 instrucciones)=8 ciclos / 4 = 2
Resultado: El tiempo promedio por instrucción mejora de 5 ciclos (sin cauce) a 2 ciclos (con cauce).

- **Describa los tipos de dependencias que afectan el funcionamiento de los cauces.**

RTA:

1. Dependencias de datos: Una instrucción depende del resultado de una instrucción previa. *RAW:* La instrucción actual necesita leer un dato que aún no ha sido escrito por una instrucción previa. *WAR:* Una instrucción intenta escribir en una ubicación que otra aún no ha terminado de leer. *WAW:* Dos instrucciones intentan escribir en la misma ubicación, y el orden en que lo hagan importa.
   Solución para riesgos RAW: Se debe determinar cómo y cuándo aparecen esos riesgos. Será necesaria una unidad de detección de riesgos y/o compilador más complejo.
   - *A nivel de Hardware:* adelantamiento de operandos (forwarding) consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando sin esperar a la escritura.
   - *A nivel Software:* instrucciones NOP o reordenación de código.
2. Dependencias de control: Una instrucción depende del resultado de un salto (branch) condicional para saber si debe ejecutarse. (Las soluciones son BTB y DL, explicado en otra pregunta mas abajo)
3. Dependencias estructurales: Dos o más instrucciones compiten por el mismo recurso de hardware al mismo tiempo. *Soluciones*. Duplicación de recursos hardware. Separación en memorias de instrucciones y datos. Turnar el acceso al banco de registros

Estas dependencias pueden causar **atascos (stalls)** en el cauce, forzando al procesador a detener temporalmente la ejecución de ciertas instrucciones hasta que las dependencias sean resueltas.

- **En un cauce segmentado con secuencia de instrucciones independientes ¿que consecuencias trae el paso de una instrucción de salto?**

RTA: El problema es que en un bucle de N iteraciones (el caso de un for), se va a producir N-1 atascos de tipo BTS, esto es ya que en la ejecución de las instrucciones en un programa, recién en la etapa ID sabe si tiene que saltar o no, y justo cuando esta está siendo ejecutada, la etapa IF de la siguiente instrucción se ejecuta (la que se encuentra en la siguiente línea), si se produce un salto, la instrucción que se encontraba en la etapa IF se descarta, y se lee la instrucción correspondiente al salto, produciendo un atasco BTS.

- **Analice los casos de salto condicional y salto incondicional menciona que posibles soluciones se pueden aplicar para evitar o disminuir las consecuencias**

RTA: BTB es técnica de Hardware (y las demás mencionadas) y DS es técnica de Software (única):

- Branch Prediction: Utiliza una tabla/buffer para predecir el salto. Si salto la vez anterior, predice que salta. Si NO salto la vez anterior predice que NO salta. Almacenamiento de la tabla: dirección de la instrucción y dirección de salto. Si nunca ejecutó el salto, carga la siguiente instrucción, si no, carga la instrucción en la tabla. En esta técnica en general se producen 4 atascos, dos atascos BTS al principio, y dos atascos BMS al final, 1 se debe por cargar la instrucción incorrecta, la otra para actualizar la tabla/buffer de BTB. Su eficacia es notable cuando se ejecutan más de 5 iteraciones, ya que mejora el flujo del cauce y reduce el impacto de los saltos.
- Delay Slot: Cambia la manera en la que se ejecutan los saltos. Retarda el salto 1 ciclo: ejecuta siempre la instrucción siguiente. Solución simple: agregar un NOP, equivalente a tener un atasco, baja el CPI, pero aumentan los CICLOS, siempre es posible. *Solución eficiente:* reordenamiento de código. Ventaja: 0 atascos garantizados siempre. *Desventaja*: los programas tienen que modificarse para seguir funcionando.
- Flujos múltiples: Duplicar las partes iniciales del cauce y deja que éste capte las dos instrucciones utilizando los dos caminos. Cuando se determina cuál es correcta, descarta el camino incorrecto.
- Precaptar el destino del salto: Se precapta la instrucción del salto además de la siguiente a la bifurcación. Se guarda esta instrucción hasta que se ejecute la instrucción de bifurcación. Si se produce el salto, el destino ya habrá sido pre captado y lo ejecuta finalmente.
- Buffer de bucles: es una memoria pequeña de gran velocidad, gestionada por la etapa de captación de instrucción del cauce, que contiene, secuencialmente, las n instrucciones captadas más recientemente. Si se va a producir un salto, el hardware comprueba si el destino del salto está en el buffer.

- **Describa las políticas de emisión de instrucciones en un cauce segmentado**

RTA: respuesta en la parte de “PROCESADORES SUPERESCALAR”

PASAJE DE PARÁMETROS (buscar si hay mas pregunta sobre esto)

- **Pasaje de parámetros por pila: enumerando las etapas**

RTA:

1. Guardado de registros: Se salvan los valores de los registros que se van a utilizar (incluyendo el valor de los registros de propósito especial como BP/BX) para preservarlos durante la ejecución de la subrutina.
2. Colocación de parámetros en la pila: Se accede a los parámetros mediante los registros adecuados (generalmente a través de BP o BX), y se apila en la pila en el orden en que serán utilizados.
   Paso por valor: Si el parámetro es un valor (no una referencia), se apila el valor mismo a la pila.

Paso por referencia: Si el parámetro es pasado por referencia, se apila la dirección de memoria (puntero) a la pila.

3. Ejecución de la subrutina: Una vez que los parámetros se han colocado en la pila, la subrutina se ejecuta con esos valores.
4. Recuperación de registros: Al ejecutar la instrucción "RET", se retoman los valores previamente guardados en el paso 1, en orden inverso al que fueron guardados. Esto restablece el estado anterior a la llamada a la subrutina.

- **Explique los métodos de pasaje de argumentos a procedimientos o funciones.**

RTA:

Pasaje por valor:

- Se envía una copia del valor del argumento al procedimiento o función.
- Cualquier modificación realizada al valor dentro de la función no afecta al valor original en el programa principal.
- Es útil cuando no se desea alterar el valor original.

Pasaje por referencia:

- Se envía la dirección de memoria del argumento al procedimiento o función.
- Cualquier modificación al argumento dentro de la función afecta al valor original.
- Es útil cuando se desea modificar el valor original o trabajar con estructuras de datos grandes sin copiarlas.

Pasaje por nombre (o enlazado por texto):

- El argumento se evalúa como una expresión textual en el momento en que se utiliza dentro de la función.
- Es menos común y generalmente se encuentra en lenguajes que soportan macros o procedimientos diferidos.

Pasaje mixto:

- Algunos lenguajes permiten combinar pasaje por valor y por referencia según las necesidades de cada argumento. (ej: Pascal y C++)

- **Describa el comportamiento con anidamiento de múltiples procedimientos/funciones.**

RTA:

- Cada vez que se llama a una función, se reserva un espacio en la pila (segmento de pila) que almacena los argumentos, variables locales y la dirección de retorno para reanudar la ejecución. En funciones recursivas o llamadas anidadas, se crea un segmento de pila independiente para cada invocación, garantizando que cada llamada mantenga su propio estado. Estos segmentos se liberan en orden inverso cuando las funciones terminan.
- Una función puede acceder solo a: Sus propias variables locales y las variables de una función que la contiene directamente.
- Una función NO puede acceder a: Variables locales de otras funciones hermanas y variables de funciones definidas "por fuera" de su alcance directo.

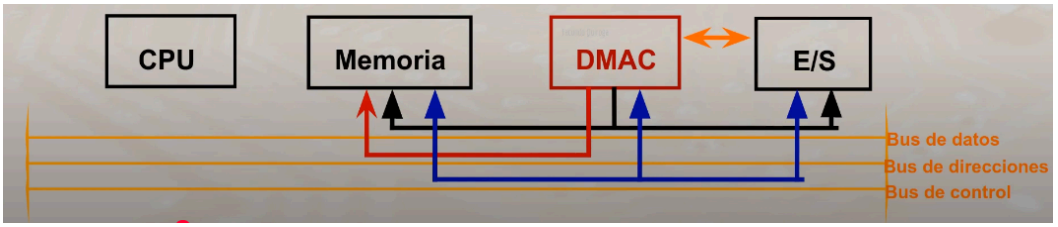
- En procedimientos o funciones anidados, las variables locales más cercanas en el nivel de anidamiento tienen prioridad. Si dos funciones tienen variables con el mismo nombre, la función interna usa su propia variable.

Consulta de estado o DMA

- **Que es el DMA**

RTA: DMA (idea o concepto) - DMAC (componente físico o hardware que implementa el método DMA)
- Es un método de transferencia de datos en el que un controlador especializado, llamado DMAC, realiza la transferencia directamente entre un dispositivo y la memoria principal, sin la intervención constante de la CPU. Cuando un dispositivo solicita una transferencia, la CPU cede el control del bus al DMAC (los 3 buses), proporcionándole los parámetros necesarios (cantidad de datos, direcciones, etc.). El DMAC se encarga de gestionar la transferencia utilizando el bus de datos, mientras la CPU sigue ejecutando instrucciones desde la caché. Durante la transferencia, el DMAC controla el flujo de datos en el bus. Al finalizar, envía una señal a la CPU para devolverle el control del bus.
Existen dos tipos principales de transferencia:

1. **Por bloques**: Se transfieren múltiples datos consecutivamente, hasta que se termine la transferencia o hasta que un evento externo lo interrumpe. Es más eficiente porque reduce la sobrecarga de coordinación con la CPU pero durante la transferencia, el bus está ocupado completamente por el DMA, bloqueando la CPU.
2. **Por palabras**: Se transfieren datos uno a la vez, alternando el control del bus entre la CPU y el DMAC. Es más flexible y permite una mayor interacción con la CPU pero requiere más operaciones de coordinación

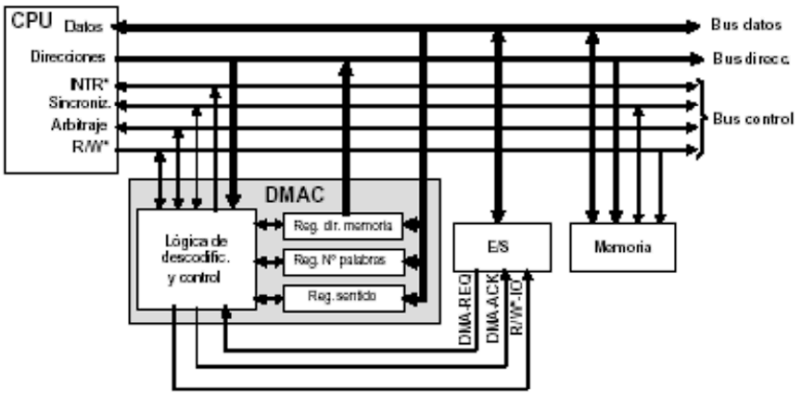


- **Componentes internos de DMA.**

RTA:

- Lógica de decodificación y control (es el cerebro de DMA, interpreta órdenes enviadas por la CPU y coordina operaciones internas, como gestionar el flujo de datos entre la memoria y los dispositivos)
- Registro de memoria (almacena la dirección de memoria base donde comenzará la transferencia de datos)
- Registro de N° de Palabras (almacena la cantidad de palabras(datos) que deben transferirse, se decrementa automáticamente después de cada palabra transferida y cuando llega a 0, indica que la transferencia ha terminado)f
- Registro de sentidos (indica la dirección de la transferencia. Dispositivo a Memoria: Transferencia de lectura. Memoria a Dispositivo: Transferencia de escritura)

Estructura de un DMAC



- **Describa las características funcionales del Acceso Directo a Memoria (DMA).**

RTA:

- Transferencia sin intervención constante de la CPU: permite que los datos se transfieran directamente entre la memoria principal y los dispositivos periféricos, sin que la CPU tenga que manejar cada palabra de datos.
- Cede el control de los buses: durante la transferencia, la CPU cede el control de los buses al DMAC para que realice la operación.
- Parámetros configurables: la CPU configura el DMA proporcionando los parámetros necesarios, como: dirección de inicio de la memoria, cantidad de datos a transferir,dirección del periférico y dirección del flujo (escritura/lectura).
- Transferencia en segundo plano: mientras el DMAC realiza transferencia, la CPU puede continuar ejecutando otras instrucciones desde la caché. mejorando la eficiencia global del sistema.
- Señal de interrupción al finalizar: una vez que la transferencia está completa, el DMAC envía una señal de interrupción a la CPU para notificar que ha terminado, devolviendo el control de los buses.
- Soporte para modos de transferencia: el DMA puede operar en diferentes modos: Modo por palabras (transfiere datos de uno en uno), Modo por bloques (transfiere múltiples datos en una sola operación)

- **Etapas de transferencia DMA**

RTA:

- INICIALIZACIÓN DE LA TRANSFERENCIA: La CPU debe enviar a la interfaz del periférico los parámetros específicos de la operación del periférico y, por separado, configurar al DMAC con los parámetros necesarios para realizar la transferencia directa de datos. Después de la inicialización la CPU retorna a sus tareas y ya no se preocupa más de la evolución de la transferencia.

Inicialización del interfaz (Bus master: CPU-Bus slave: Interfaz)
- Nº de bytes a transferir
- Tipo de transferencia (lectura/escritura)
- Otra información de control (pista, sector, etc.)

Inicialización controlador DMA(Bus master: CPU-Bus slave: DMAC)
- Nº de bytes o palabras a transferir
- Tipo de transferencia (lectura/escritura)
- Dirección de memoria inicial para la transferencia
- Nº de canal (para DMAs con varios canales)

- REALIZACIÓN DE LA TRANSFERENCIA: Cuando el periférico está listo para realizar la transferencia se lo indica al DMAC, este pide control del bus y se realiza la transferencia entre el

periférico y la memoria. Después de cada transferencia se actualizan los registros DMAC (número de bytes a transferir y dirección de memoria).

- FINALIZACIÓN DE LA TRANSFERENCIA: El DMAC libera el bus y devuelve el control a la CPU. El DMAC suele activar una señal de interrupción para indicar a la CPU la finalización de las operaciones de E/S solicitadas.

- **La coherencia de datos de un sistema jerárquico se ve afectada por el uso de DMA?**

RTA: Si, ya que el DMA accede directamente a la memoria principal para realizar transferencias, sin pasar por la CPU ni actualiza automáticamente el contenido de la canche.

-Entonces cuando se realiza una escritura directa en la memoria principal por el DMA, las copias de esos datos en la caché de la CPU pueden quedar desactualizadas, ya que la caché no es informada de los cambios realizados directamente en la memoria.

-Cuando se realiza una lectura directa en la memoria principal por el DMA, si la CPU ha modificado un dato en su caché pero aun no lo ha escrito en la memoria principal, el DMA leerá datos desactualizados, causando inconsistencias.

- **Tipos de transferencias**

RTA:

- Por rafagas (burst): el DMAC solicita el control del bus a la CPU. Cuando la CPU concede el bus el DMAC no lo libera hasta haber finalizado la transferencia del bloque de datos completo.
  Ventaja: transferencia rápida.
  Desventaja: puede degradar el rendimiento del sistema.
- Por robo de ciclo (cycle-stealing): el DMAC solicita el control del bus a la CPU. Cuando la CPU concede el bus al DMAC se realiza la transferencia de una palabra y luego el DMAC libera el bus. El DMAC solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo.
  Ventaja: No se degrada el rendimiento del sistema.
  Desventaja: la transferencia tarda más tiempo.

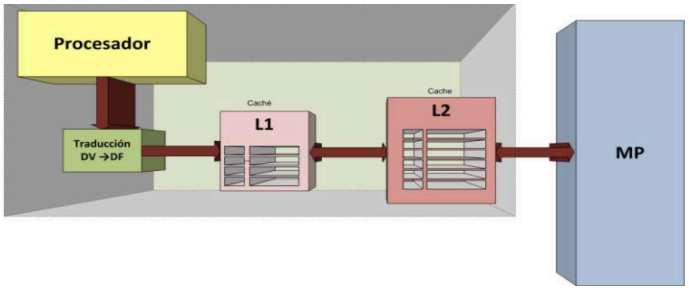
  Si bien el trabajo de la CPU es lento, no será tanto como si ella realizará la transferencia. Por lo tanto, para transferencia de E/S de múltiples palabras, es la técnica más eficiente.

MEMORIA CACHÉ

- **Compare las correspondencias entre la Memoria Principal (MP) y la Caché.**
- **Funciones de correspondencia entre memoria y memoria caché.**
- **Describa las funciones que se utilizan en la política de ubicación de bloques en memoria caché**
- **Describa los elementos a tener en cuenta en el diseño de una memoria caché.**

RTA:

1. Organización: tamaño, costo y niveles



2. Política de ubicación: Tipo de función de correspondencia
    - ***Directa***: un bloque solo puede estar almacenado en un lugar de la caché. La posición en la caché se determina mediante una función matemática que utiliza el número de bloque de la memoria principal y el tamaño de la caché: Nº línea caché = Nº bloque ref. MOD Nº líneas caché. Implementación sencilla. Bajo costo de Hardware. Puede ocurrir **conflicto de caché** si varios bloques de la memoria principal mapean a la misma línea de la caché.
    - *Totalmente asociativa*: un bloque puede almacenarse en cualquier lugar de la caché. El sistema busca el bloque completo en la caché utilizando una **etiqueta** (tag) asociada al bloque. Búsqueda costosa (en tiempo principalmente).
    - *Asociativa por conjuntos:* un bloque puede almacenarse en un conjunto restringido de lugares en la caché. Un conjunto es un grupo de líneas de la caché: Nº conjunto = Nº bloque ref. mod Nº conjuntos caché. Implementación más compleja que la directa pero es más flexible.
3. Política de reemplazo: Algoritmo de sustitución
    - *Directa*: no hay elección, sólo hay una posible línea para cada bloque.
    - Asociativa: **LRU** (se sustituye el bloque que se ha mantenido en la caché por más tiempo sin haber sido referenciado) - **FIFO** (se sustituye aquel bloque del conjunto que ha estado más tiempo en la caché) - **LFU** (se sustituye aquel bloque del conjunto que ha experimentado menos referencias) - **Aleatoria** (se sustituye al azar.

      Estos algoritmos deben implementarse en hardware (controlador de caché) para conseguir velocidad)
4. Política de escritura: cuando escribimos datos en la caché, debemos asegurarnos de que esos datos estén correctamente **actualizados en todos los niveles** de caché y memoria principal para evitar tener copias **inconsistentes** del mismo dato. Las políticas de escritura definen **cómo** y **cuándo** realizar estas actualizaciones para mantener la coherencia de los datos entre los diferentes niveles.
    - ***En acierto:*** (cuando tengo que escribir algo en la caché, elemento a escribir/cambiar se encuentra en la memoria caché):

      • ESCRITURA INMEDIATA (Write-through): se actualizan simultáneamente la posición de la caché y de la memoria principal. Se genera mucho tráfico y retrasa la escritura. Se mantiene la coherencia en todo momento. Suele combinarse con la técnica “no carga en escritura” (no-write allocate).

      • POST-ESCRITURA (Write-back): la información sólo se actualiza en la caché y esa información se marca con un bit de ACTUALIZAR o “sucio”. Cuando un bloque sea desalojado de la caché, se comprueba este bit y si está activo, se escribe la información de dicho bloque en la memoria principal. (la memoria principal puede contener información errónea en algún momento). Suele combinarse con la técnica carga en escritura (write allocate).
    - ***En fallo***: (cuando tengo que escribir algo en la caché, elemento a escribir/cambiar se encuentra en la mem fallo (elemento a escribir NO se encuentra en la memoria caché):

      • NO CARGA EN ESCRITURA (no-write allocate): el bloque no se lleva a la memoria caché. Se escribe directamente en la memoria principal. Habitual con escritura inmediata (Write-through).

      • CARGA EN ESCRITURA (write allocate): la información se lleva de la memoria principal a la caché. Se sobrescribe en la caché. Habitual con write-back.

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

RTA:

- **Ventajas:**
  - Mejora el tiempo de acceso global al aprovechar distintos niveles con diferentes velocidades.
  - Reduce la latencia de acceso a datos frecuentes.
  - Los niveles superiores (L2, L3) pueden ser más grandes y almacenar más datos.
- **Desventajas:**
  - Aumenta la complejidad del diseño del sistema.
  - Incrementa el costo del hardware.
  - Si la coherencia de los datos no se maneja bien, puede generar inconsistencias entre los niveles.

- **Si se pretende mejorar el tiempo de acceso medio a la memoria caché ¿Sobre qué parámetros será necesario trabajar y que propone como medidas para hacerlo?**

RTA:

Parámetros clave a trabajar:

1. **Latencia de la caché:** Reducir el tiempo de acceso a los datos almacenados.
2. **Tasa de aciertos (hit):** Maximizar la proporción de accesos que encuentran datos en caché.
3. **Tasa de fallos (miss):** Minimizar los accesos que no encuentran datos en caché y requieren buscar en memoria principal.
4. **Tiempo de reposición (miss penalty):** Reducir el tiempo necesario para cargar datos desde la memoria principal a la caché.

Medidas para mejorarlo:

- Utilizar políticas de correspondencia más eficientes, como **asociativa por conjuntos**, para mejorar la tasa de aciertos.
- Incrementar el tamaño de la caché, aunque esto debe balancearse con el costo y la latencia.
- Implementar **algoritmos de reemplazo óptimos**, como LRU (Least Recently Used), para minimizar los fallos.
- Usar niveles adicionales de caché (L2, L3) para reducir el tiempo promedio de acceso.
- Optimizar las políticas de escritura, como "write-back" en lugar de "write-through".

Risc/Cisc

- **Describa tres características qué usted considere las más importantes de las arquitecturas RISC.**

RTA:

- Repertorio de instrucciones limitado y sencillo: conjunto de instrucciones más pequeños y simples, permitiendo que se decodifican y ejecutan en menos ciclos del reloj (1).
- Gran número de registros de uso general: permite almacenar datos de manera más eficiente y reducir la necesidad de acceder a la memoria principal constantemente.

- Optimización de la segmentación de instrucciones: la segmentación es fundamental porque las instrucciones son simples y se pueden descomponer fácilmente en etapas que se ejecutan paralelamente. Esto permite mejorar el rendimiento, ya que el procesador puede aprovechar mejor las unidades de ejecución.

- **Ventajas y desventajas de RISC**

RTA:

- **Eficiencia y velocidad**: Como las instrucciones son simples y se ejecutan en un solo ciclo, los procesadores RISC suelen ser más rápidos en términos de ejecución de instrucciones.
- **Facilidad de segmentación**: Las instrucciones simples permiten un diseño de segmentación más eficiente y, por lo tanto, mejor rendimiento en procesadores con múltiples etapas.
- **Menos necesidad de decodificación compleja**: La simplicidad de las instrucciones facilita la decodificación, lo que mejora el rendimiento global.
- **Mayor cantidad de instrucciones**: Como las instrucciones son más simples, puede ser necesario un mayor número de instrucciones para realizar una tarea compleja en comparación con arquitecturas CISC (Complex Instruction Set Computer).
- **Dependencia del compilador**: Para optimizar el uso de registros y minimizar el acceso a la memoria, RISC depende en gran medida de un compilador eficiente.

- **Describa qué características considera las más importantes de las arquitecturas CISC.**

RTA

El diseño CISC busca optimizar la reducción de las instrucciones necesarias para ejecutar un programa. Para ello, incluye instrucciones más complejas que pueden realizar varias operaciones en un solo ciclo de reloj.

1. Repertorio amplio de instrucciones:
2. Operaciones complejas entre operandos: pueden realizar operaciones complejas directamente entre operandos almacenados en memoria o en registros internos, sin necesidad de separar las operaciones en varias instrucciones más simples.
3. Acceso directo a memoria: reduce la necesidad de cargar los valores en los registros.
4. Complicada decodificaciones de instrucciones: ya que son más complejas, incrementa la carga de trabajo en el procesador y puede reducir la velocidad de ejecución.

- **Describa las características que diferencian los procesadores RISC respectos a los CISC**

RTA:

Resumen de las principales diferencias:

Característica	RISC	CISC
Conjunto de Instrucciones	Reducido y sencillo	Grande y complejo
Ejecuta Instrucciones	Una instrucción por ciclo	Puede requerir varios ciclos por instrucción
Uso de Registros	Gran cantidad de registros	Menos registros de propósito general
Acceso a Memoria	Solo `LOAD` y `STORE` acceden a la memoria	Instrucciones complejas que acceden a memoria
Compiladores	Gran dependencia del compilador	Menos dependencia del compilador
Tamaño y Complejidad	Más sencillo y pequeño	Más complejo y grande
Rendimiento	Mejor rendimiento en tareas simples	Mejor rendimiento en tareas complejas

PROCESAMIENTO SUPERESCALAR

- **¿Qué son los procesadores superescalares?**

RTA:

Un procesador superescalar es aquel que usa múltiples cauces de instrucciones independientes, cada cause consta de múltiples etapas, de modo que puede tratar varias instrucciones a la vez.

El enfoque superescalar conlleva a la duplicación de algunas o todas las partes de la CPU/ALU.
Capta varias instrucciones a la vez y a continuación intenta encontrar instrucciones cercanas que sean independientes entre sí y puedan ejecutarse en paralelo. Puede eliminar algunas dependencias innecesarias mediante el uso de registros adicionales y el renombramiento de las referencias a registros en el código original.

- **¿Qué características definen un procesador superescalar? Describa las causas que pueden retardar el funcionamiento de los mismos**

RTA: **Características que definen un procesador superescalar:**

Sin paralelismo de instrucciones, el hardware no tendría qué procesar en paralelo, y sin paralelismo de la máquina, el hardware no podría sacar partido del paralelismo en el código.

Paralelismo de instrucciones:

- Se refiere a las **oportunidades en el código del programa** para ejecutar varias instrucciones simultáneamente, siempre y cuando:
  - Las instrucciones sean **independientes** entre sí.
  - No haya dependencias de datos, de control o conflictos de recursos.
- Esto es una propiedad del software, pero el hardware debe ser capaz de identificar y explotar este paralelismo.

Paralelismo de la máquina:

- Se refiere a las **capacidades del hardware del procesador** para aprovechar el paralelismo de instrucciones que ofrece el programa.
- Incluye elementos como:
    - Múltiples unidades funcionales.
    - Renombre de registros.
    - Ejecución fuera de orden.
    - Predicción de saltos eficiente.
    - Ventanas de emisión amplia

Paralelismo de instrucciones, Limitaciones:

- **-Dependencia de datos verdadera (RAW):** Una instrucción necesita un dato producido por otra instrucción previa.
- **-Dependencia relativa al procedimiento**: Las instrucciones que siguen a una bifurcación (condicional IF x ejemplo) tienen una dependencia relativa al procedimiento en esa bifurcación y no pueden ejecutarse hasta que ésta lo haga.
- **-Conflictos en los recursos:** Es una competencia entre dos o más instrucciones por el mismo recurso al mismo tiempo. Presenta el mismo comportamiento que una dependencia de datos, sin embargo, los conflictos en los recursos pueden superarse duplicando estos.
- **-Dependencia de salida (WAW) y Antidependencia (WAR):** afectan al paralelismo porque pueden generar conflictos de recursos, incluso si no están directamente relacionadas con los valores de los registros

- **Compare las políticas de Emisión de instrucción.**

RTA:

1. **Emisión y Finalización en orden**: Las instrucciones se emiten y completan en el mismo orden en que aparecen en el programa, sin reordenamientos.
2. **Emisión en orden y Finalización desordenada:** Las instrucciones se emiten en el orden del programa, pero pueden finalizar en un orden diferente según la disponibilidad de recursos y datos. Mejora la velocidad de las instrucciones que necesitan muchos ciclos.
3. **Emisión y Finalización desordenada**: Las instrucciones se emiten y finalizan en un orden diferente al del programa, aprovechando al máximo el paralelismo al reorganizar su ejecución según dependencias y recursos.

- **¿De qué depende el paralelismo de una máquina superescalar?**

RTA: El paralelismo de una máquina superescalar depende de varios factores que determinan su capacidad para ejecutar múltiples instrucciones simultáneamente:

1. **Número de instrucciones captadas por ciclo:**
    - A mayor cantidad de instrucciones captadas, mayor potencial para ejecutar en paralelo.
2. **Número de unidades funcionales:**
    - Se refiere a la cantidad de unidades funcionales disponibles (como ALUs, FPUs, unidades de carga/almacenamiento, etc.).
    - Más unidades funcionales permiten que varias instrucciones se ejecuten al mismo tiempo.
3. **Mecanismos para localizar instrucciones independientes:**
4. **Identificar paralelismo y organizar las etapas F, D y E en paralelo para diferentes instrucciones**
5. **Renombre de registros:**
    - Soluciona dependencias falsas (dependencia de salida o antidependencia) asignando registros físicos adicionales para permitir que varias instrucciones operen sin conflicto.

6. **Ventana de instrucciones (emisión desordenada):**
    - La ventana de instrucciones es el conjunto de instrucciones que el procesador analiza simultáneamente.

- **Reordenamiento de registros**

RTA: **El renombre de registros** se utiliza para resolver riesgos de dependencia de datos entre la **emisión** y la **finalización** de instrucciones. Esto incluye:

1. **Dependencia de salida (WAW):** Cuando dos instrucciones intentan escribir en el mismo registro.
2. **Antidependencia (WAR):** Cuando una instrucción se escribe en un registro que otra necesita leer.

El renombramiento asigna registros físicos adicionales para evitar estos conflictos, permitiendo que las instrucciones se ejecuten en paralelo sin interferencias, y maximizando el paralelismo en los procesadores superescalares.

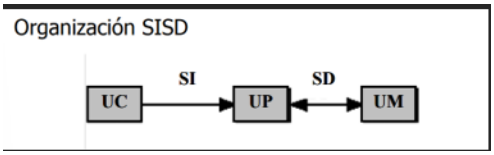
El único riesgo que quedaría es el RAW (dependencias verdaderas)

PROCESAMIENTO PARALELO

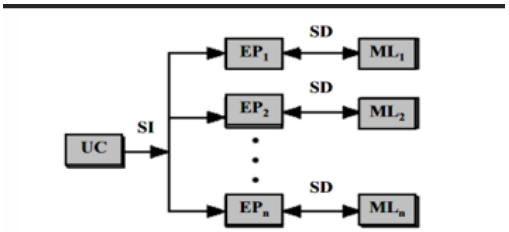
- **Describa las 4 variantes de arquitecturas de la taxonomía de Flynn**

RTA:

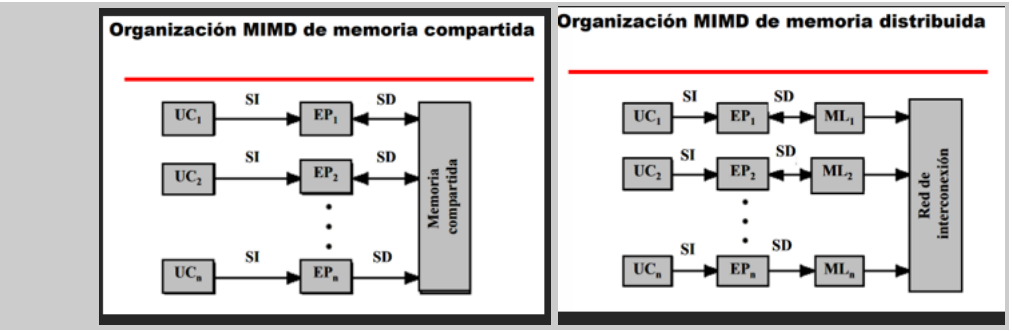
1. **SISD**: una secuencia de instrucciones y una secuencia de datos. Un único procesador interpreta una única secuencia de instrucciones (SI) para operar con los datos almacenados en una única memoria(UM). Computadoras monoprocesador caen en esta categoría.



2. **SIMD**: una secuencia de instrucciones y múltiples secuencias de datos. Una única instrucción máquina controla paso a paso la ejecución simultánea de un cierto número de elementos de proceso (EP). Cada elemento de proceso tiene una memoria dedicada (ML). Procesadores vectoriales y matriciales.



3. **MISD:** múltiples secuencias de instrucciones y una secuencia de datos. Se transmite una secuencia de datos a un conjunto de procesadores. Cada procesador ejecuta una secuencia de instrucciones diferente. Esta estructura nunca ha sido implementada.
4. **MIMD**: múltiples secuencias de instrucciones y múltiples secuencias de datos. Significa que un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes, Formas de comunicarse: Memoria compartida (SMP,sistemas NUMA) - y Memoria Distribuida (clusters)



- **MIMD , mas explicado**

RTA:

La taxonomía de Flynn clasifica a los sistemas de varios procesadores según sus capacidades de procesamiento paralelo. Una de las categorías es MIMD.
En la organización MIMD, los procesadores son de uso general: cada uno es capaz de procesar todas las instrucciones necesarias para realizar las transformaciones apropiadas de los datos.

Los computadores MIMD se pueden dividir según la forma que tienen los procesadores para comunicarse:

- Memoria compartida (fuertemente acoplada): Los procesadores comparten una memoria común, Todos los procesadores tienen acceso a todas las partes de la memoria y los procesadores se comunican unos con otros a través de esa memoria.
    - Multiprocesador simétrico (SMP): **Varios procesadores comparten una única memoria mediante un bus compartido u otro tipo de mecanismo de interconexión.** El tiempo de acceso es igual a todas las regiones de memoria, así como el tiempo de acceso a memoria es el mismo para todos los diferentes procesadores.
    - Acceso no uniforme a memoria (NUMA): El tiempo de acceso a zonas de memoria diferentes puede diferir.
- Memoria distribuida (débilmente acoplada):
    - Clusters: Computadoras completas interconectadas mediante algún tipo de red que trabajan conjuntamente como un único recurso de cómputo, creando la ilusión de que se trata de una única máquina.

- **Describa las características que diferencian los SMP respecto a los clusters**
- **¿Qué características describen un cluster de computadoras?**

RTA:

Similitudes:

- Ambos son soluciones diseñadas para aplicaciones de alta demanda de recursos, como bases de datos, simulaciones científicas y servicios empresariales críticos.
- Están disponibles comercialmente y se utilizan ampliamente en entornos donde la capacidad de procesamiento y la disponibilidad son esenciales.
- Pueden soportar cargas de trabajo intensivas y distribuir tareas para mejorar el rendimiento.

SMP (Multiprocesamiento Simétrico):

- **Definición:** Es una arquitectura donde múltiples procesadores comparten un único sistema operativo, memoria y recursos I/O, operando como una sola unidad.
- **Facilidad de administración:** Es más fácil de configurar y administrar porque todos los procesadores están estrechamente integrados y comparten los mismos recursos.
- **Compatibilidad:** Es más cercano a los sistemas de un solo procesador, lo que facilita la migración de aplicaciones diseñadas para sistemas tradicionales.

- **Rendimiento:** Es ideal para aplicaciones que requieren acceso rápido y compartido a la memoria central.
- **Escalabilidad:** Limitada, ya que agregar más procesadores eventualmente genera cuellos de botella en la memoria compartida y en el bus.
- **Disponibilidad:** Una falla en un componente central (como la memoria o el sistema operativo) puede afectar a todo el sistema.

Clúster:

- **Definición:** Es un conjunto de computadoras independientes conectadas en red que trabajan juntas como si fueran un único sistema.
- **Escalabilidad:** Ofrece una **escalabilidad incremental** superior, ya que se pueden agregar nodos adicionales al clúster sin problemas significativos de cuellos de botella.
- **Disponibilidad:** Tiene **mayor tolerancia a fallos**, ya que si un nodo falla, otros nodos pueden asumir su carga de trabajo.
- **Configuración:** Requiere mayor complejidad en la configuración y administración, ya que los nodos necesitan software especializado para coordinar tareas y manejar fallos.
- **Mejor relación precio/prestaciones:** Es posible configurar un cluster con mucha potencia a menos costo que un computador independiente.

BUS

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

RTA: Un bus es un camino de comunicación entre dos o más dispositivos, un medio de transmisión compartido en donde cualquier señal transmitida por un dispositivo está disponible para que otros dispositivos conectados al bus puedan acceder a ella. Solo un dispositivo puede transmitir con éxito en un momento dado (para que sus señales no se solapen y se distorsionen).

El bus conecta a los componentes principales de la computadora memoria, procesador, E/S) se denomina BUS DEL SISTEMA.

- **Método de arbitraje**

Puesto que, en un instante dado, sólo una unidad puede transmitir a través del bus, se requiere algún método de arbitraje. Los diversos métodos se pueden clasificar como **centralizados o distribuidos**.
En un esquema centralizado, un único dispositivo hardware, denominado controlador del bus o árbitro, es responsable de asignar tiempos en el bus.
En un esquema distribuido, no existe un controlador central. En su lugar, cada módulo dispone de lógica para controlar el acceso, y los módulos actúan conjuntamente para compartir el bus.
En ambos métodos de arbitraje, el propósito de designar un dispositivo, el procesador o un módulo de E/S como maestro del bus. El maestro podría entonces iniciar una transferencia de datos (lectura o escritura) con otro dispositivo que actúa como esclavo en este intercambio concreto.

- **Temporización**

La temporización es la forma en la que se coordinan los eventos en el bus, pueden ser:
Temporización síncrona: la presencia de un evento en el bus está determinada por un reloj.
Temporización asíncrona: la presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo.

La temporización **síncrona** es más fácil de implementar y comprobar pero es menos flexible que la **asíncrona**, ya que todos los dispositivos deben utilizar la misma frecuencia de reloj.

- **¿Qué elementos característicos definen un bus?**

RTA: Un bus está constituido por varias líneas capaces de transmitir señales binarias, a las cuales se les asigna una función particular y se pueden clasificar en tres grupos:

- **Bus (líneas) de datos**: Se utiliza un “camino” para transmitir datos entre los módulos del sistema.
  - Transmiten datos, a este nivel no existe diferencia entre “datos” e “instrucciones”.
  - El ancho del bus es un factor clave a la hora de determinar las prestaciones (8, 16, 32, 64 bits).
- **Bus (lineas) de direcciones**: Identifica la fuente o destino de un “dato”
  - El ancho del bus de direcciones determina la máxima capacidad de memoria posible del sistema
- **Bus (líneas) de control**: Se utiliza para controlar el acceso y el uso de las líneas de datos y de direcciones.
  - Transmite información de señales de control y temporización

- **Mencione las principales diferencias entre un bus PCI y SCSI.**

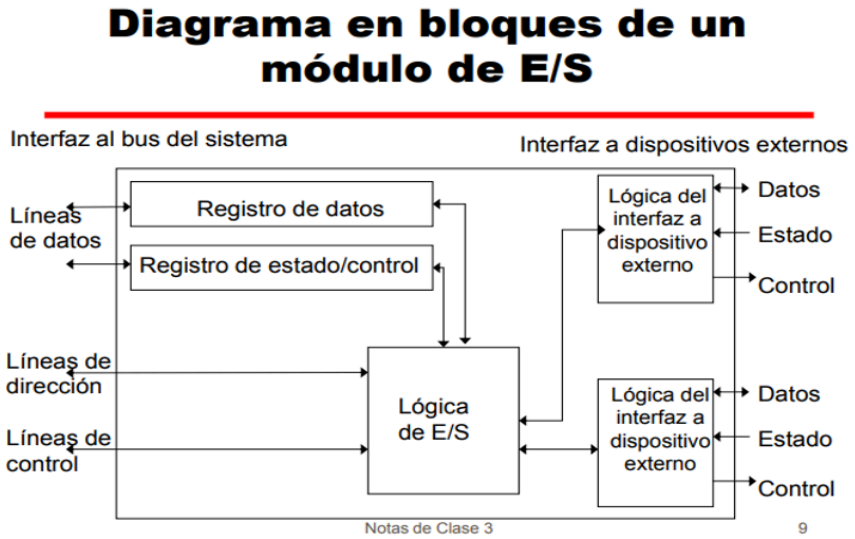
RTA:

Bus PCI (Peripheral Component Interconnect - Interconexión de Componente Periférico)
Es un bus estándar de computadoras para conectar dispositivos periféricos directamente en la placa base. de ancho de banda elevado, independiente del procesador.
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.

E / S

- **¿Cómo es la estructura de un módulo de E/S?**

RTA: Realiza la interfaz entre el procesador, la memoria (bus) y los 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 / 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 ej, 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 lineas 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.

- **Describa las posibles técnicas que puede utilizar una CPU para realizar operaciones E/S**

RTA:

E/S programada con espera de respuesta (Polling): En este enfoque, la CPU verifica repetidamente el estado de un dispositivo de E/S (entrada/salida) para saber si está listo para realizar una operación. Mientras tanto, la CPU puede estar inactiva, esperando la respuesta del dispositivo.

E/S con interrupciones: En este caso, la CPU puede continuar con otras tareas hasta que el dispositivo de E/S necesite atención. El dispositivo interrumpe la CPU cuando está listo para realizar una operación, lo que permite que la CPU responda solo cuando sea necesario, sin tener que estar verificando continuamente el estado del dispositivo.

E/S con acceso directo a la memoria (DMA): En esta técnica, el controlador DMA transfiere datos entre la memoria y el dispositivo de E/S directamente, sin la intervención constante de la CPU. Esto mejora la eficiencia al liberar a la CPU de la gestión de la transferencia de datos.

- **Describa los posibles modos de ubicación de los módulos de E/S**

RTA:

Los módulos de Entrada/Salida (E/S) son componentes del sistema encargados de gestionar la comunicación entre el procesador, la memoria y los dispositivos periféricos. Para acceder a estos módulos, existen dos posibles modos de ubicación:

- **E/S Mapeada en memoria:** Se necesita solo una línea de lectura y solo una línea de escritura en el bus. Existe un único espacio de direcciones para las posiciones de memoria y los dispositivos de E/S.
- **E/S Aislada:** Puesto que el espacio de direcciones de E/S está aislado del de memoria, este se conoce como E/S aislada. Los puertos de E/S sólo son accesibles mediante una orden específica de E/S, que activa las órdenes de E/S del bus. La ventaja es que se puede utilizar un amplio repertorio de instrucciones permitiendo una programación más eficiente.

- **¿Qué elementos componen una máquina con arquitectura Von Neumann? Describir la función de cada uno**

RTA: Una máquina con **arquitectura Von Neumann** tiene los siguientes elementos:

1. **CPU**: Se encarga de ejecutar las instrucciones. Tiene:
    - **ALU**: Realiza operaciones aritméticas y lógicas.
    - **Unidad de Control**: Interpreta las instrucciones y coordina la CPU.

    - **Registros**: Almacenan datos e instrucciones temporalmente.
2. **Memoria**: Guarda las instrucciones y datos que la CPU necesita. Incluye:
    - **RAM**: Memoria principal, donde se almacenan instrucciones y datos.
    - **Memoria Caché**: Memoria más rápida para acceder a datos frecuentes.
3. **Dispositivos de Entrada/Salida (E/S)**: Permiten la interacción con el exterior (teclado, pantalla, etc.).
4. **Buses**:
    - **Bus de Datos**: Transporta los datos entre componentes.
    - **Bus de Direcciones**: Lleva las direcciones de memoria.
    - **Bus de Control**: Coordina las operaciones entre componentes.

Todo esto trabaja de manera conjunta para ejecutar las instrucciones y procesar datos en la computadora.