Respuestas ARQUITECTURA

¿Qué es una pila? Describir el comportamiento con anidamiento de múltiples procedimientos/funciones utilizando pila.
Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el funcionamiento y uso de la pila

Tema: **Subrutinas**

Tecnicas de pasaje de parametros:

- **Registros:** Se usan los registros para almacenar los parametros. Limitado por los pocos registros disponibles
- **Memoria:** Se usa un area especifico de memoria para guardar parametros. Dificil de estandarizar.
- **Pila:** El metodo mas usado, cont.

La pila es una estructura de datos que posee un registro, Stack pointer o SP, cuya direccion es el “tope” de la pila, es decir, lo ultima que se agrego a esta. Se la utiliza a traves las instrucciones PUSH (guardar un dato en el tope de la pila) y POP (recibir dato en el tope). Es muy utilizada en el funcionamiento de las subrutinas, para pasar parametros y guardar direcciones de retorno, el procedimiento de la llamada a al subrutina y el pasaje de parametros consta de varios pasos:

1) **Guardar BP:**
   La primera instruccion es **PUSH BP** (se guarda este puntero en la pila) y la siguiente **MOV BP, SP** (se mueve el actual SP a BP). Esto tiene la funcion de definir el PUNTERO BASE para la subrutina, a partir de esa direccion nos podremos ir desplazando (en la pila) por los distintos datos necesarios para el proceso.

2) **Pushear Variables Locales y Registros (OPCIONAL):**
   Si es necesario se guardaran en la pila (es decir, por sobre el BP), Variables locales y Registros que puedan ser utilizados en la subrutina. Los registros solo se guardan si van a ser modificados en la subrutina, si no no es necesario.

3) **Acceso a Parametros:**
   Para acceder a los parametros, se utiliza el BP que fue guardado en el paso 1. Se suman 2 + direccion de retorno (esta apilada bajo el BP) + tamaño de los parametros que esten entre el que se quiere acceder y el primero. Por ejemplo, si se tienen 3 parametros y se quiere acceder al 1 (el que esta mas “abajo” en la pila) se debe sumar 2 + direccion de retorno + tamaño de 2 parametros (los params 2 y 3) para acceder a la direccion del 1.

4) **Salida de la subrutina:**
   Actualmente, desde el tope hacia “abajo” de la pila se encuentran, en orden:
   -Registros guardados
   -Variables locales
   -BP
   -Direccion de retorno
   -Parametros
   Debemos desapilar los registros (si hay) y luego movemos SP a BP, hacemos un POP (sacamos BP de la pila) y finalmente usamos la instruccion RET, que nos devovlera al programa del cual fue llamada la subrutina.

En caso de que tengamos anidamiento de subrutinas, el procedimiento se repite de igual manera, cada subrutina tendra su propio BP y su espacio de datos en la pila a partir de este, y se deben desapilar en orden inverso al que fueron sido llamadas (la subrutina mas “profunda” va a ser la primera en ser desapilada cuando se comience a retornar)

Tema: INTERRUPCIONES

Finalidad de las interrupciones. Para que se utiliza un controlador de interrupciones
Explique el mecanismo de interrupción. Describa las distintas fuentes de interrupción que conozca y el tratamiento a realizar cuando hay múltiples interrupciones.

Las interrupciones se utilizan para “avisar” a la cpu que hay algo de lo que debe hacerse cargo, puede ser un error en una instruccion, un fallo de hardware, algun temporizador terminando un ciclo que requiera ciertas acciones, o un dispositivo de E/S requiriendo atencion del procesador.

Existen de dos tipos, enmascarables y no enmascarables. Si una interrupcion es enmascarable, entonces la CPU puede ignorarla o al menos retrasarla y seguir con su ejecucion previa. Las no enmascarables requieren accion inmediata y no pueden ser ignoradas.

Si entre una instruccion y la siguiente, se generó una interrupcion (marcada con un flag), entonces se suspendera la ejecucion normal, guardando su contexto, y se llevará la ejecucion a la subrutina de la interrupcion llamada.

Si se generan interrupciones mientras se esta atendiendo una interrupcion, esta sera ignorada hasta que se termine la subrutina de la primera. Una vez terminada se examinara el nuevo pedido de interrupcion. Todo esto es asi a menos que la nueva interrupcion sea de mayor prioridad que la primera, en cuyo caso se interrumpe la primera y se ejecutara el pedido de mayor prioridad

GESTOR DE INTERRUPCIONES:

El gestor es necesario debido a que el procesador no tiene muchas entradas para pedidos de interrupciones, y al mismo tiempo existen muchos productores de interrupciones, por lo cual necesitamos de otro dispositivo que se encargue de gestionar estos pedidos. El PIC sirve para gestionar muchas fuentes de interrupciones y enviarle la señal a la CPU cuando se genera una interrupcion. Ademas tiene la capacidad de establecer prioridades por si se generan varias en simultaneo.

Tema: ENTRADA/SALIDA

Describir la estructura de un módulo de E/S. ¿Qué es DMA y cómo funciona?

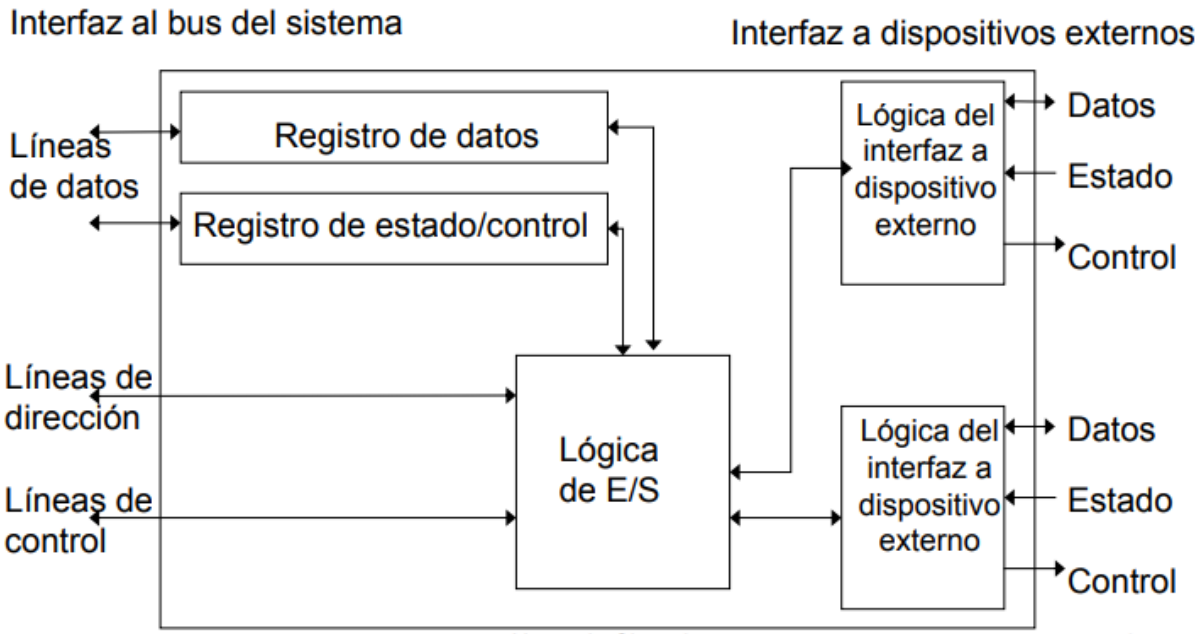
¿Cómo es la estructura de un módulo de E/S? Describa las posibles técnicas que puede utilizar una CPU para realizar operaciones de E/S.

La existencia de muchos dispositivos de entrada/salida, y tan distintos entre sí, genera la necesidad del módulo de E/S, que hace de intermediario entre los dispositivos y la CPU.

El módulo se debe ocupar de ciertas cosas:

- **Control:** gestiona los pedidos de interrupcion de los perifericos e informa a la CPU sobre sus estados. Deteccion de errores
- **Transferencia:** Pasaje de datos e instrucciones desde disp. a CPU y visceversa.
- **Almacenamiento:** Puede almacenar pequeñas cantidades de datos temporalmente antes de ser transferidos.

Estructura de modulo de E/S:



El modulo consiste en varias entradas para los dispositivos, cada una con su linea de Datos, Estado y Control. Por el lado de comunicacion con el sistema interno de la computadora, tiene dos registros para almacenamiento de datos, y coneccion con el bus de datos, direcciones y control.

El modulo tiene la capacidad de ocultar informacion sobre el periferico al procesador, para hacer las operaciones lo mas simples posibles.

Existen 3 tecnicas de gestion de E/S:

- **E/S Programada:** En esta tecnica, la CPU recibe desde el modulo que hay una operacion de E/S pendiente, una vez realizada la operacion por el modulo, la CPU comprueba periodicamente si la transferencia de datos desde/hacia el dispositivo haya sido completada. Hasta que no se termine la CPU quedaria ociosa, esperando para realizar otra transferencia. Por lo tanto, se pierde mucho tiempo en el que la CPU podria estar realizando otras operaciones

- **E/S con Interrupciones:** En este caso, una vez que la CPU ordena al modulo que realice cierta operacion, esta sigue con su ejecucion, y cada cierto periodo chequea si se efectuo algun pedido de interrupcion, en cuyo caso lo atendera y realizara la transferencia. Una vez terminada retoma su ciclo normal y cuando se la precise de nuevo se enviara otra interrupcion. Esto genera mucha mejor eficiencia del uso de la CPU que la tecnica anterior. Tambien se deben tener en cuenta estructuras para identificar cual dispositivo fue el que genero la interrupcion, pero de esto se ocupa el modulo

- **DMA (Acceso directo a memoria):** En esta tecnica, el modulo de E/S (DMAC) obtiene una mejora substancial, y es que puede realizar transferencias hacia y desde memoria/perifericos sin intervencion de la CPU. El unico momento en donde

la CPU esta presente es para inicializar la operacion, pasandole toda la informacion necesaria para la transferencia al modulo (tamaño, tipo de transferencia, direcciones). Una vez que el dispositivo esta listo para la transf. el DMAC toma el control del bus de datos y comienza la transferencia con la memoria principal. Una vez terminado esto, el DMAC libera el bus y genera una interrupcion a modo de aviso de finalizacion.
Esto puede generar que la CPU sufra problemas por no poder utilizar el bus de datos mientras lo utiliza el DMAC, se puede resolver de distintas maneras:

- **Cache:** La cpu no necesariamente tiene que acceder a memoria si lo que necesita esta presente en la cache
- **Rafaga:** El DMAC transfiere todo de una, pero esto bloquea mucho a la CPU
- **Cycle stealing:** El DMAC pide acceso al bus, pero cuando se le otorga solo transfiere una palabra, y luego libera nuevamente el bus. Esto genera transferencias mas lentas pero es un buen balance entre CPU y DMAC para el acceso al bus

Tema: SEGMENTACION DE CAUCE

¿Qué es la segmentación de cauce? Explique los atascos producidos por saltos

¿Qué ventajas nos brinda un cauce segmentado? Describa las diferentes formas que pueden mejorar el funcionamiento de un cauce cuando ejecuta instrucciones de transferencia de control.

La segmentacion de cauce es una forma de organizar el hardware de la CPU para dividir las instrucciones en segmentos, y asi ejecutar mas de una al mismo tiempo.

Consiste en 5 etapas, y el tiempo que tardara una etapa se amoldara a lo que tarde la mas lenta (asi todas tardan lo mismo). Estas etapas son:

1. **F - Fetch:** Se busca la instruccion a ejecutar y se incrementa el PC. Se utiliza la Memoria de Instrucciones
2. **D - Decode:** Se decodifica la instruccion y se obtienen los operandos. Se utiliza el Banco de Registros si existe un operando almacenado en ellos
3. **X - Execute:** Se ejecuta la operacion en la ALU
4. **M - Memory:** Se accede a memoria de datos (si se requiere).
5. **W- Writeback:** Se escribe el resultado en registros (si se requiere)

Por lo tanto, y asumiendo que no hay ningun atasco, se podrian tener 5 instrucciones ejecutandose simultaneamente, multiplicando por 5 el tiempo de un procesador secuencial, pero sin reducir el tiempo que tarda en procesarse una sola instruccion.

Hay ciertos inconvenientes que pueden suceder usando segmentacion de cauce, los cuales terminan generando **ATASCOS**. Se pueden dar de distintos tipos:

- **Riesgos estructurales:** Dos instrucciones deben acceder al mismo recurso en el mismo ciclo. Por ejemplo, una instruccion esta en su etapa M (acceso a memoria) y otro en su etapa F (tambien accede a memoria, buscando la instruccion). Sin una manera concreta de resolverlo, esto generaria un atasco retrasando toda la ejecucion un ciclo

  Solucion: Hay tres posibles soluciones frente a este problema:
  - **Duplicacion:** Teniendo recursos “repetidos” se puede resolver, cada instruccion usando uno distinto
  - **Separacion:** Se separa la memoria en memoria de datos y de direcciones, asi no habra conflictos en este tipo de problemas
  - **Turnos:** Las instrucciones se toman turnos en mitades de ciclo de reloj, una en el primero y otra en el segundo, y asi no se desperdicia un ciclo entero

- **Riesgos de datos:** Suceden cuando un operando no esta disponible para ser utilizado en una instruccion, debido a distintas causas:

  - **RAW:** Una instruccion genera un dato que debe ser leido por otra posterior

    Soluciones:

    - **Adelantamiento de operando:** Al finalizar la operacion que genera el dato se saltea el paso de almacenar el resultado (se almacena igual pero se prioriza pasarselo a la otra instruccion)

    - **Reordenamiento de codigo:** Se separan lo mas posibles las instrucciones que causan la dependencia de datos a partir del uso de un compilador inteligente, evitando el problema (pero generando posibles nuevas dependencias WAW y WAR)

  - **WAW:** Se escribe dos veces en una misma direccion. Solo es un problema si se reordenan las instrucciones

  - **WAR:** Se escribe un dato antes de que fuera leido por una instruccion anterior

- **Riesgos por saltos:** Cuando una instruccion puede generar un salto (es decir que la proxima instruccion no necesariamente es la siguiente secuencialmente) se genera un problema, ya que para saber cual sera la proxima instruccion se debe saber el resultado de la primera.
    - **Saltos Incondicionales:** Aca la problematica es menor, ya que sabemos que el salto se va a efectuar, entonces la unica preocupacion es en buscar rapidamente la direccion de salto
    - **Saltos Condicionales:** Estos saltos generan el problema grande, ya que se debe saber lo antes posible o incluso predecir la direccion de salto, para evitar la perdida de muchos ciclos.

      Soluciones:

        - **Adelanto del resultado:** En vez de conocer el resultado de la condicion en la etapa X (ejecucion), se resuelve en la etapa D (decodificacion). Esto resta dos ciclos a la penalizacion
        - **Prediccion de saltos (Hardware):** Se utilizan distintas tecnicas con la intencion de predecir si se saltara o no basado en los previos saltos, si se acierta no se pierde ningun ciclo, y si falla se atrasa un ciclo. Se pueden usar estructuras como un simple historial basado en los saltos anteriores que predictira en base a ello, o el Branch Target Buffer (BTB), una cache pequeña que guarda los datos de los saltos (instruccion que salta, direccion a donde salta y ultima accion tomada).
        - **Salto retardado (Software):** Se trata de reordenar las instrucciones de forma tal que, mientras se resuelva el salto, la siguiente instruccion en ser ejecutada sea una independiente del resultado del salto.

Tema: Cache

Justifique el uso de dos niveles de caché.

Caché: mencione algoritmos de reemplazo y políticas de escritura.

¿Por qué funciona una jerarquía de memoria? Describa las políticas de ubicación y de reemplazo de bloques en memoria caché

La jerarquia de memoria tiene el objetivo de brindar la mayor cantidad de espacio, la velocidad mas rapida y al menor costo posible. Esto lo logra combinando distintos niveles, cada uno independiente y usando tecnologia diferente.

Se deben cumplir dos propiedades:

- **Inclusion:** Los datos presentes en un nivel deben estar tambien en los niveles inferiores.
- **Coherencia:** Todas las copias de una misma informacion debe ser igual en todos los niveles

Por que funciona? Principios de localidad

- **Localidad Temporal:** Un dato que fue accedido recientemente tiene altas chances de volver a ser accedido, por lo tanto se necesitan menos transferencias cuando esto sucede

- **Localidad Espacial:** Los datos cercanos a el dato accedido tienen altas chances de ser referenciados tambien, por lo tanto se ahorran transferencias entre niveles si se “suben” los datos en bloque de manera “predictiva”

Por que dos niveles de caché?

Cuando se separa la memoria cache en dos, se tiene por un lado la caché L1 (nivel uno), que se encuentra en un sector interno facil de acceder para el procesador, y la L2, mas grande que L1 pero con acceso un poco mas lento. Si no se encuentra un dato en L1, se chequea si existe en L2, es una forma de evitar aun mas los costosos accesos a memoria principal

Tipos de correspondencia:

Como determinar en que linea de la memoria caché se va a escribir el bloque de memoria entrante? Se presentan 3 tecnicas:

- **Correspondencia directa:** Segun el numero que tenga el bloque en MP, se hace una funcion de hash a partir de ese valor y segun la cantidad de lineas de la caché. Por ejemplo, se tienen 10 lineas y se ingresa el bloque 8, se hace 8 mod 10, entonces el bloque se ubicara en la linea 8 de la caché.
  Ventajas: Es muy rapido encontrar un bloque, se aplica la funcion y se lo encuentra rapidamente
  Desventajas: Si hay dos bloques que corresponden a la misma linea, se tendran que estar reemplazando cada vez que sean cargados a caché

- **Correspondencia asociativa:** Los bloques se ubican en cualquier lugar de la caché. Esto resuelve el problema de la directa, pero crea uno nuevo, y es que la busqueda de cada bloque implica buscar en cada linea previa a la que se encuentra

este, por lo tanto es muy ineficiente.

- **Correspondencia asociativa por bloques:** una mezcla de ambas tecnicas. La caché se divide en bloques que contienen varias lineas. Un bloque de memoria es ubicado en un bloque segun su numero en MP, pero adentro del bloque puede ser ubicado en cualquier ubicacion.

Politicas de sustitucion:

Para correspondencia directa, el bloque a reemplazar siempre sera el que este en el lugar correspondiente al bloque entrante.

Para Indirecta se tienen varias alternativas:

- **LRU:** El bloque que se uso hace mas tiempo es reemplazado, requiere un control de tiempo
- **FIFO:** El bloque que entro hace mas tiempo es el que se reemplaza, requiere control de acceso
- **LFU:** El bloque que se utilizo menos veces es reemplazado, requiere control de uso
- **Aleatoria:** Se sustituye una linea al azar

Politicas de escritura:

Cuando se escribe sobre un dato en caché, en algun momento este cambio debe verse reflejado en todas las memorias “inferiores” (cache L2, Memoria Principal, etc), ya que otros dispositivos podrian querer acceder a este a traves de MP y leer un dato erroneo.

Hay distintas formas de mantener la **Coherencia** cuando se realizan escrituras, se dividen a partir de si el cambio es efectuado en Caché (Acierto) o en Memoria Principal (Fallo, no se encontro el bloque en caché):

- **Acierto:**
  - **Write-through:** Cuando se escribe un bloque en caché, se actualiza inmediatamente en MP. Esto garantiza que no habra incoherencia de datos, pero genera mucho trafico de datos y ralentiza las operaciones de escritura.
  - **Write-back:** Se escribe en caché y se marca al bloque como modificado (dirty). El cambio no se ve reflejado en MP hasta que ese bloque de memoria no sea reemplazado, lo cual puede generar incoherencia, pero genera menos trafico.

- **Fallo:**
    - **Write allocate:** Cuando se escribe en MC, se lleva el bloque tambien a caché, habitual con Write-Back

    - **No-Write allocate:** Cuando se escribe en MC, no se lleva el bloque a caché, habitual con Write-Through

Tema: RISC

Describa las características que diferencian a los procesadores RISC respecto de los CISC.
Describa cómo se debe implementar la estructura de pila en un procesador de tipo RISC cuyos registros son genéricos (basarse en MIPS) ¿Cómo se deberá trabajar el anidamiento de procesos / funciones? RTA: VENTANA

Que es RISC?

Las computadoras de arquitectura RISC (repertorio reducido de instrucciones/reduced instruction set) plantean varias diferencias (en general, mejoras) a comparacion de la arquitectura SISC. Estas son las principales:

- **Registros:** Poseen una gran cantidad de registros de uso general, lo cual genera mas memoria de alta velocidad. Tambien pueden no tener mas registros, sino mejor tecnologia de compilacion para optimizar el uso de los mismos (generando un efecto similar)

    - **Almacenamiento de variables:** la mejora en el campo de los registros hace que las variables locales sean mas rapidas de acceder, ya que se guardan en registros en vez de MP, duran mas tiempo ya que hay mas registros.

    - **Subrutinas:** Se utiliza la tecnica de "**ventanas de registros**" cuando se llaman varias subrutinas anidadas, en donde se separan en 3 Areas los contextos de cada subrutina. Los registros de parametros (entrada), datos locales y temporales (salida). Los registros de salida de un nivel seran los de entrada del nivel llamado a continuacion (el mas "profundo" en las llamadas), generando un menor uso de cantidad de registros.

    - **Registros simbolicos y compilador:** Cuando se podria generar una dependencia de datos por el uso de varias instrucciones sobre un mismo registro, el compilador puede crear registros "simbolicos" para asignarles a esas instrucciones y que no haya problemas. Es decir, el compilador usa otros registros (o en caso de que no haya disponibles, usara MP) para

representar al registro real, y luego lo guardara en el real cuando ya no se generen dependencias de datos

- **Instrucciones:** Tienen un repertorio de instrucciones pequeño y simplificado, solo teniendo las instrucciones imprescindibles para el funcionamiento, y las tareas mas complejas estaran compuestas de varias instrucciones.

- **Segmentacion:** Estan enfocadas en optimizar la segmentacion de instrucciones, por lo cual pueden ejecutar muchas instrucciones en poco tiempo a comparacion.

El problema que tenian las SISC tenia que ver con tener demasiadas instrucciones, debido a que se creaban nuevas para los lenguajes de alto nivel. Por lo cual se mezclaban con las instrucciones de mas bajo nivel y habia demasiados modos de direccionamiento.

Tema: **BUSES**

¿Que es un Bus? Describa los diferentes tipos, métodos de arbitraje y técnicas de sincronización. Mencione las principales diferencias entre PCI y SCSI.

Que es un Bus, tipos de buses, temporización y métodos de arbitraje

Bus: Es un “camino” de comunicacion entre dos dispositivos, que suele estar agrupado en varias lineas (se transmite 1 bit por linea). En una computadora se precisan buses de tres tipos principales:

- **Datos:** Transmite todos los datos (instrucciones incluidas), el ancho del bus definira el tamaño de la palabra de la arquitectura (8,16,32,64 bits)

- **Direcciones:** Identifica la ubicacion de un dato en memoria, el ancho de este bus define la capacidad maxima de memoria en el sistema.

- **Control:** Transmite señales de control como timers, lectura/escritura o interrupciones

Hay 2 grandes divisiones de buses:

- **Dedicados:** Lineas separadas para cada tipo, por ej: 16 lin direcciones, 16 lin datos, 1 lin control

- **Multiplexados:** Lineas compartidas de direcciones y datos, por ej: 16 lin direcciones y datos, 1 lin control, 1 lin control para definir direcciones o datos

Existen 2 formas de tomar control sobre un bus, osea ser el Maestro o Arbitro:

- **Arbitraje centralizado:** Un unico dispositivo tiene control sobre los tiempos del bus (normalmente la CPU o el DMAC)

- **Arbitraje distribuido:** Cada modulo tiene acceso al bus y se tienen tecnicas para regular quien toma control y cuando

Tecnicas de temporizacion:

- **Sincronica:** Los eventos en el bus estan coordinados a traves de la señales de reloj, al cual todos los dispositivos tienen acceso mediante el bus de control. Los eventos suelen durar un solo ciclo de reloj (por ej, entre dos señales de 1 o “subida”)

- **Asincronica:** No se utiliza el reloj, si no que los dispositivos se comunican entre si para informar que se esta realizando un evento.

El bus PCI y el SCSI son dos medios de comunicacion con los perifericos, el SCSI es utilizado principalmente para medios de almacenamiento como discos duros, mientras que el PCI se utiliza hoy en dia para conectar distintos tipos de placas (de video por ejemplo) a la motherboard.

Tema: **SUPERESCALARES**

PROCESADORES SUPERESCALAR -CARACTERISTICAS.
Describa las políticas de emisión de instrucciones en un cauce segmentado.

Enfoque **supersegmentado** del cauce: Se dividen las etapas que ocupan mas tiempo en sub-etapas, generando mas instrucciones ejecutandose en simulteaneo, pero sin mejorar el tiempo total de una instruccion.

Enfoque **superescalar:** Se ejecutan mas de una instruccion en simultaneo, a partir de la duplicacion de ciertos componentes de la CPU/ALU. Asi por ejemplo se pueden captar multiples instrucciones o ejecutar operaciones numericas simultaneas. Esto mejora el grado de paralelismo (mas instrucciones ejecutandose en simultaneo).

Estas mejoras son muy favorables, pero traen nuevos conflictos en la ejecucion. Para que dos instrucciones se puedan ejecutar en paralelo sin ningun problema, deben ser **Independientes**, es decir que la ejecucion de una no dependa ni interfiera de ninguna forma con la otra. Existen distintas formas de causar problemas entre dos instrucciones, y cada una tiene su propia resolucion:

- **Riesgos RAW:**
- **Riesgos WAW y WAR:**
- **Riesgos por salto:**

Grado de paralelismo: Depende de la cantidad de instrucciones que se puedan ejecutar en simultaneo, y para tener un grado alto, se necesitan prestaciones de parte de la computadora (recursos duplicados, captacion simultanea de instrucciones) y las instrucciones no deben generar dependencias entre si, es decir ser independientes.

Formas de emitir instrucciones:

- **Emision y resolucion ordenadas:** Las instrucciones deben empezar y terminar en orden secuencial
- **Emision ordenada y resolucion desordenada:** Las instrucciones se emiten en orden, pero cuando terminan no deben esperar a sus anteriores si fueron mas rapidas
- **Emision y resolucion desordenadas:** Las instrucciones van llegando y se incluyen en una estructura en donde esperaran a poder ser ejecutadas segun la disponibilidad de la unidad que necesiten usar, por lo tanto es las mas rapida pero la mas desordenada (causa dependencias si no se atiende)

Renombre de registros:

Este desorden va a generar dependencias de datos (WAW y WAR especificamente) que deben ser tratadas, y se usa la tecnica de renombre de registros, es decir el uso de registros simbolicos que vimos previamente, para solucionarlas, dejando solo las dependencias RAW.

Tema: **SMP y CLUSTERS**

¿Qué características posee un multiprocesador simétrico (SMP)?
Funcionamiento de un cluster
Comparacion entre ambos

Que cosas tienen en comun?

Ambas arquitecturas son **MIMD,** es decir Multiple Instrucción Multiple Data. Estas maquinas poseen multiples procesadores que ejecutan diferentes secuencias de instrucciones en simultaneo, y se diferencian entre si en como es su memoria, compartida (SMP) o distribuida (Clusters), a parte de otras caracteristicas.

SMP (Simmetric Multi-Proccesor)

Los SMP son computadoras autonomas que poseen dos o mas procesadores interconectados. Todos los procesadores poseen el mismo tiempo de acceso a memoria, tienen acceso a MC y E/S, y desempeñan las mismas funciones. Estas maquinas tienen un SO integrado.

Poseer multiples procesadores trae ventajas, como la velocidad de ejecucion si pueden ser utilizados en paralelo, y la buena disponibilidad, en caso de que uno de los procesadores no este disponible, se podran seguir usando el resto. Ademas, poseen escalabilidad ya que se pueden agregar mas procesadores, mejorando el rendimiento aun mas.

Tienen un problema importante: el uso de **Bus Compartido,** el cual limita el rendimiento a sus ciclos de reloj. Para reducir el numero de accesos al bus, cada procesador tiene su propia caché, pero esto en si puede traer problemas de coherencia de datos(Resuelto con hardware).

Clusters

Son muchas computadoras interconectadas a traves de una red que funcionan como un solo recurso. Cada computadora es un **nodo,** y presentan una escalabilidad absoluta, es decir que practicamente no tienen limite. Poseen una mejor disponibilidad que SMP, ya que si un nodo se desconecta, no tendra mucho impacto en el cluster entero. Su principal desventaja frente a SMP es su mayor complejidad, que conlleva mas espacio y energia, ademas de la posesion de muchas computadoras en el primer lugar.