Preguntas Final

Explique qué es y qué elementos constituyen el ISA. Describa los elementos más importantes. ¿De acuerdo con la cantidad de operandos como se clasifican las instrucciones? ¿Cuáles son las ventajas de las instrucciones con tres operandos y registros de propósitos generales (arquitectura Register-Register) comparada con la de dos operandos en registro y uno en memoria (Register-Memory)

El "Conjunto de instrucciones de la arquitectura" o "ISA" es un modelo abstracto que define toda la información necesaria para programar en lenguaje máquina. Es la interfaz entre el hardware y el software y nos da un conjunto de reglas para comunicarnos de manera eficiente con el hardware, en otras palabras, es una abstracción de cómo utilizar el procesador para que sea implementada por la microarquitectura, que es el conjunto de técnicas de diseño para implementar el ISA.

Elementos del conjunto de instrucciones de la arquitectura:

- **Los modos de operación soportados**, un modo de operación define que restricciones de recursos y que operaciones puede realizar/disponer un programa al ser ejecutado por el procesador. Esto con el objetivo de proteger la funcionalidad de la computadora frente a fallas y comportamiento malicioso. Un procesador tiene muchos modos de operación, que están dispuestos en una jerarquía, van desde los más privilegiados (sin restricciones, modo kernel), a los menos privilegiados(recursos e instrucciones restringidas, modo usuario). Para que un programa pueda ejecutarse, el ISA provee de mecanismos especiales para que un modo inferior, pueda acceder a los recursos del modo superior (instrucciones o interrupciones). Si bien las capacidades de las que disponen los modo usuario suelen ser un subconjunto de aquellas del modo supervisor, en algunos casos pueden ser muy diferentes, es decir, que un modo usuario emule capacidades que el sistema no tiene.
- **Organización de la memoria:** Las posiciones de la memoria están organizadas linealmente en orden consecutivo. Cada una esta enumerada con un numero único para cada espacio de memoria, es decir que cada palabra de memoria tiene una dirección única dentro del espacio de memoria correspondiente. La organización de la memoria es una clasificación de TODA la memoria disponible. La misma se clasifica según que uso tendrá o que información almacenara. La memoria se compone de **3 espacios:**
    - El espacio de programa: Es el espacio donde se almacena el programa que se va a ejecutar(instrucciones), además de la dirección donde se inicia la ejecución del programa(reset).
    - El espacio de datos: Es el espacio donde se almacenan los datos que utilizara el programa durante su ejecución y las E/S de los periféricos.
    - El espacio de configuración: Es el espacio donde se almacenan los datos de configuración del procesador como la tabla de páginas, registros de configuración, etc.
- **Tipos de datos:** El dato más pequeño direccionable por un procesador es el byte, todas las palabras de tamaño mayor se almacenan como secuencias de bytes, estas son llamadas palabras

multibytes, las mismas pueden ser almacenadas de dos formas distintas, Big-endian o little endian.

- **Gestión de eventos:** Define como nuestro procesador atenderá los eventos inesperados que cambian el flujo de instrucciones, causados de manera interna o externa al procesador, es decir de manera sincrina o asíncrona al reloj interno del procesador. Estos eventos pueden ser:
    - **Interrupciones,** estas son suspensiones temporales de la ejecución de un programa, se ejecutan al terminar una instrucción, son originados por otros programas distintos al que está siendo ejecutado los cuales requieren realizar una tarea de manera inmediata.
    - **Excepciones,** estas abortan la ejecución de una instrucción, son originadas debido a errores en el procesador o en los procesadores auxiliares, generalmente por violaciones a la seguridad del mismo.
- **Conjunto de instrucciones:** el conjunto de instrucciones es el formato que contienen las instrucciones que pueden ser ejecutadas por el procesador, todo conjunto de instrucciones debe ser completo, flexible, auto contenido, eficiente.
- **Cantidad de operandos en las instrucciones:** se refiere a la cantidad de direcciones utilizadas para operandos que hay en una instrucción, existen varios tipos:
    - ISA con operandos implícitos: El formato de instrucción de este modelo no utiliza campo de dirección por que los operandos están implícitos en la instrucción. La memoria de la computadora toma la forma de una pila, las instrucciones operan solo con los datos en el tope de las pilas. Tiene ciertas desventajas como su dificultad para paralelizar instrucciones, su necesidad de instrucciones para cargar datos, etc. Su mayor ventaja es que tiene bajos requerimientos
    - ISA con un operando explícito, el formato de este modelo utiliza solo un campo dirección para un operando, ya que el otro operando y el destino son almacenados en un acumulador de manera implícita. Las desventajas de este modelo son similares a las del anterior, es un modelo difícil de paralelizar y su cuello de botella es el acumulador, además es difícil dividir la ejecución de las instrucciones en segmentos, por otro lado, las ventajas de este modelo están relacionadas con sus requerimientos y con su sintaxis, ya que tiene pocos requerimientos de hardware y al ser de sintaxis sencilla tiene instrucciones cortas y flexibles, y por lo tanto programas fáciles de comprender.
    - ISA con registros de propósitos generales, el formato de instrucción de este modelo utiliza tres direcciones, dos para operandos y uno donde se almacena el resultado. Esto funciona así ya que en este modelo se utilizan registros, en los que se puede almacenar tanto operandos como resultados información necesaria para llevar a cabo las instrucciones. Hay distintas formas de implementar esta arquitectura, en los que cambia la ubicación de los operandos:
        - Pueden estar los tres operandos (fuentes y destinos) localizados en memoria, sus desventajas están relacionadas con el alto tráfico de memoria que conlleva, por otro lado sus ventajas están relacionadas con su sintaxis sencilla que permite compiladores más sencillos.
        - Puede haber dos operandos en los registros (uno fuente y destino) y otro en la memoria, sus ventajas están relacionadas con su formato de instrucción que es fácil de codificar.
        - pueden estar los tres operandos localizados en registros con instrucciones de load y store para mover datos entre memoria y registros. Sus ventajas están relacionadas

con su numero fijo de ciclos de reloj por instrucción, lo cual permite una facilidad para paralelizar y segmentar la ejecucion de instrucciones, sus desventajas son que la calidad del compilador influye en la ejecucion de instrucciones y que se debe implementar un mayor numero de instrucciones.

- **Modos de direccionamiento**, estos definen como las instrucciones identifican a los operandos. Especifican como calcular la dirección de memoria efectiva de un operando mediante el uso de la información contenida en los registros y constantes contenidas en la instrucción (los llamados inmediatos).

Un ISA puede clasificarse según la complejidad de sus operaciones:

- Conjunto de instrucciones reducido, también llamado RISC, este conjunto simplifica la microarquitectura implementando de manera eficiente las instrucciones más utilizadas para optimizar el desempeño. Sus objetivos son paralelizar las instrucciones y reducir los accesos a memoria.
- Conjunto de instrucciones complejas, también llamado CISC, es un modelo de arquitectura de computadoras que se destaca por ser muy amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros que hacen que el código sea compacto.

Explique cómo funciona una microarquitectura segmentada. Explique cuáles son sus ventajas y problemas. ¿Cuál es la arquitectura de computadora óptima para su implementación? Fundamente. ¿En qué se diferencia de una microarquitectura superescalar?

Para comenzar una microarquitectura es una descripción de cómo será implementado el comportamiento descripto por el conjunto de instrucciones(ISA) a través de un sistema digital, esta se representa como diagramas de bloque que describen las interconexiones de los diversos elementos de la microarquitectura, es importante tener un buen diseño de la microarquitectura ya que este ***afecta directamente*** al desempeño del sistema, influyendo en el costo, rendimiento, consumo energético, etc. La microarquitectura segmentada es una implementación del conjunto de instrucciones RISC, es un diseño en el cual la ejecución de una instrucción se divide en varias ***etapas independientes*** lo cual requiere registros para almacenar los valores generados por cada una, esto permite aprovechar la diferencia de tiempos de ejecución de las instrucciones, a diferencia de la arquitectura monociclo en la que cada instrucción dura un ciclo de reloj. Otra ventaja de esta implementación es que permite ejecutar mas de una instrucción por ciclo de reloj en el mismo pipeline(solo si estas instrucciones no son dependientes), teóricamente hablando, si el pipeline esta dividido en n etapas, el procesador podrá ejecutar hasta n instrucciones simultáneamente, aun así dividir el pipeline en muchas etapas no aumenta el rendimiento de nuestro procesador, ya que en la práctica aumentar el número de etapas también aumenta tanto el tiempo requerido para llenar y vaciar el pipeline, como la circuitería necesaria para implementarlo.

Riesgos *de la arquitectura segmentada*

Los riesgos son situaciones que **imposibilitan** el comienzo de la ejecución de la próxima instrucción en el próximo ciclo. Se clasifican en, riesgos estructurales(el recurso necesario para ejecutar la instrucción esta ocupado), riesgos de datos(se necesita esperar a que se complete una instrucción previa para disponer del dato necesario en la instrucción actual) y riesgos de control(estos suceden cuando el flujo de programa

cambia según el resultado de instrucciones previas, es decir, se realiza una acción de control que altera el flujo de programa).

La micro arquitectura superescalar es una arquitectura que permite aumentar la paralelizacion a nivel de máquina, aumentando los recursos (registros y unidades de ejecución) disponibles en el procesador, es decir se aumenta la cantidad de pipelines. Cada pipeline consiste de múltiples etapas que pueden ejecutar múltiples instrucciones al mismo tiempo, permitiendo el procesamiento simultaneo de múltiples grupos (streams) de instrucciones. A diferencia de esto la micro arquitectura segmentada solo aumenta el paralelismo a nivel de instrucción solapando la ejecución de instrucciones.

Por ultimo, si bien la arquitectura adecuada para implementar la micro arquitectura segmentada depende de varios factores como el coste, la energía, etc. La mas adecuada para aumentar el rendimiento es la arquitectura de Harvard, esto ya que permite paralelizar el acceso a los datos y a las instrucciones, porque tiene un buses separados para la memoria de datos y para la memoria de instrucciones.

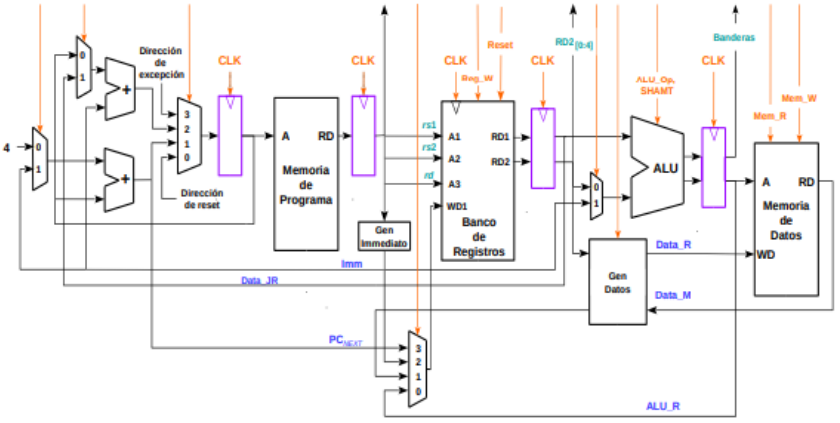
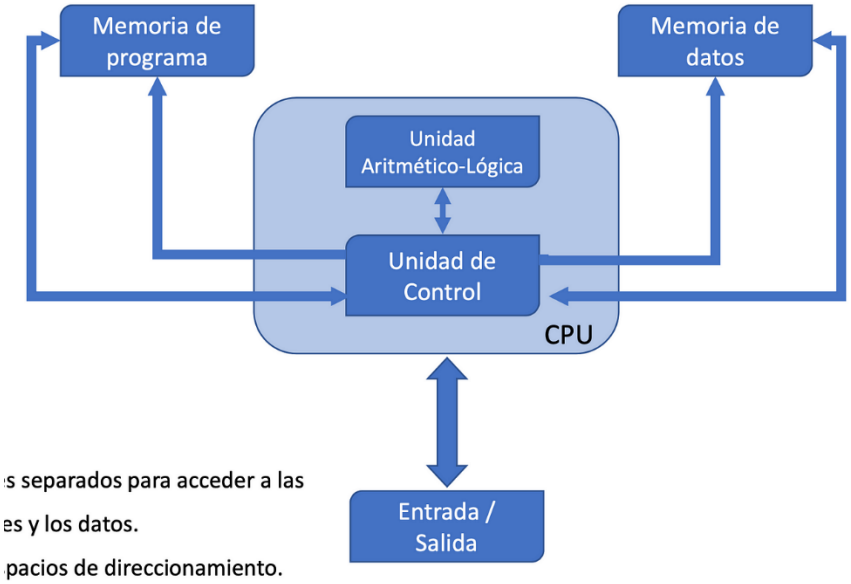


Fig 1: diagrama en bloque del datapath de una arquitectura segmentada



s separados para acceder a las
es y los datos.
pacios de direccionamiento.

Fig 2: diagrama en bloque de la arquitectura de hardvard.

¿Que son los riesgos de datos? Explique cómo se resuelven.

Los riesgos de datos son situaciones en las que una instrucción se refiere o depende de un resultado que no se ha calculado o recuperado. En estas situaciones el resultado de las instrucciones que muestran estas dependencias se ve afectado. Existen tres tipos de riesgos de datos:

- Read After Write(RAW), también llamada dependencia de datos, se refiere a una situación en la que una instrucción se refiere a un resultado que aún no se ha calculado.
- Write After Read(WAR), también llamada dependencia de nombre, se refiere a una situación en la que una instrucción escribe un dato en un registro, antes de que otra instrucción lea ese dato, es decir una instrucción depende de un valor que se reemplazo antes de ser recuperado o leído.
- Write After Write(WAW), también llamada Reutilización de registro se refiere a una situación en la que dos instrucciones sucesivas escriben el mismo registro, presentando una dependencia de nombre.

Los métodos utilizados para resolver los riesgos de datos son varios:

- Adelanto de datos, este método permite resolver el riesgo de leer un dato de un registro antes de escribir en el mismo (riesgo de nombre), requiere que el dato que necesitamos este disponible para ser utilizado en alguna parte del datapath y consiste en adelantar los datos directamente desde donde estén, pueden estar en la etapa de escritura en memoria (luego de ser ejecutados por la ALU) o en la etapa de escritura en registros (sucede cuando el dato es obtenido desde memoria), se adelantan hasta la etapa de ejecución(antes de ser utilizado por la ALU, donde se utilizará el dato adelantado o no según corresponda). La forma de implementar esto es con una unidad de adelanto, la cual estará en la etapa de ejecución y seguirá las condiciones dadas en la figura 3 para decidir si un operando se tomara de la unidad de adelanto o directo desde registro.
- Detenciones, estas resuelven los riesgos generados por una lectura después de una escritura cuando el dato no está disponible en el datapath (por ejemplo, si todavía no se leyó desde memoria). Consiste en detener la ejecución de instrucciones posteriores a la detención (en el código) utilizando una instrucción especial NOP, además existe otra forma de utilizar las detenciones, que involucra reemplazar todas las instrucciones del datapath por instrucciones NOP. Se utiliza una unidad de detección de peligros que funcione durante la etapa de decodificación, que sirva para poner una detención entre la instrucción de carga y las instrucciones que dependan de ella, esto se hace siguiendo las condiciones de la figura 4, se debe detener la etapa de búsqueda también, para que no avance el PC y no perder la instrucción detenida.
- Reordenamiento de Programa, en algunos casos se pueden resolver los riesgos de datos simplemente reordenando las instrucciones desde nuestro código para que haya más ciclos de reloj entre dos instrucciones que tengan dependencias.

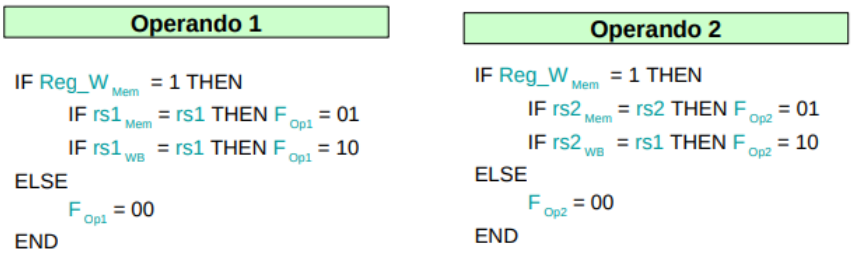


Fig. 3: condiciones de la Unidad de Adelanto para utilizar el dato adelantado o no

```
IF Mem_R = 1
    IF rd = rs1 OR rd = rs1 THEN En = 0
ELSE
    En = 1
END
```

Fig. 4: condiciones para insertar una detención.

¿Que son los riesgos de control? Explique cuáles son las posibles soluciones. Explique cómo funciona un predictor. Explique cuáles son sus ventajas y problemas. ¿Cuáles son los lugares donde se pueden implementar los predictores? ¿En qué se diferencia una predicción estática de una dinámica?

Los **riesgos de control** suceden cuando la ejecución de una instrucción se ve imposibilitada debido a que la misma depende de la decisión de una instrucción anterior generalmente relacionada a bifurcaciones, es decir existe una dependencia de una instrucción o varias a la decisión de un salto (flujo de control). Para resolver estas dependencias se utilizan distintos métodos, los cuales preservan los resultados del programa, pero pueden cambiar su flujo de instrucciones. Estos métodos son los siguientes:

- Esperar a la decisión del salto, este método consiste en esperar que la instrucción de salto llegue a la etapa de ejecución para así de esta manera poder conocer si el salto será tomado o no. Para esto se usan operaciones NOP. Esto tiene la ventaja de que preserva por completo el orden de programa, pero a la vez demanda más tiempo.
- Predicción de saltos, este método consiste en predecir como seguirá el flujo de instrucciones luego del salto, y empezar a ejecutarlo de manera solapada con el salto, si la predicción es errónea, se realiza un flush a las instrucciones ejecutadas luego del salto, por otro lado, si es correcta se continua con el flujo de instrucciones normalmente. Para poder realizar esto se debe pensar lo siguiente, existen dos tipos de saltos, los incondicionales y los condicionales, para los primeros debemos determinar solo la dirección final de salto, ya que el mismo se toma siempre, en cambio para los condicionales se debe determinar también si el salto será ejecutado o no, y luego la dirección de salto. La información necesaria para determinar la dirección final de salto puede encontrarse en la misma instrucción de salto(jal), o en algún registro y parte de la instrucción. El problema también es que para predecir un salto también debemos saber que tipo de instrucción es, lo cual no sucede hasta la etapa de decodificación, lo que a poder implementar la predicción de saltos en tres etapas diferentes.
    - Detectar los saltos en paralelo con la decodificación, se utiliza un decodificador de saltos dedicado para detectar las instrucciones de salto antes del final de la decodificación.

- Detección después de la búsqueda, se utiliza un buffer de instrucciones de salto, para detectar las instrucciones de salto antes de que sean decodificadas.
- Detección durante la búsqueda, a diferencia de las alternativas anteriores, detecta las instrucciones al tiempo que se leen de la memoria de instrucciones, esta alternativa se utiliza solo en procesadores mas avanzados.

Si se dispone de toda la información necesaria para realizar el salto se utiliza una unidad de salto que adelanta su ejecución hasta la etapa posterior a la búsqueda, por otro lado si esto no es asi, se realiza una especulación (predicción de salto) sobre el resultado del salto(esto sucede tanto para los salto condicionales como para los incondicionales) y su direccion final, si la decisicion o especulación es tomada, entonces se continua normalmente, en otro caso se descartan las instrucciones ejecutadas especulativamente. La forma de realizar esta especulación varia según que queramos averiguar.

- ***Destino del salto***, si queremos averiguar la dirección final del salto de un registro, deberíamos esperar al datafetch, lo cual ***aumenta el tiempo*** requerido para realizar el salto, para adelantar esta información se introduce un ***buffer de destino de saltos***(BTB), la cual es una memoria de doble puerto que almacena las direcciones de programa correspondientes a los saltos, en conjunto con el destino de saltos, es decir prácticamente es una cache de destinos de saltos.
- ***Decisión del salto condicional***, si bien no es posible saber la decisión de salto hasta que la instrucción de salto condicional haya pasado la etapa de ejecución, la decisión del salto se puede predecir en un gran porcentaje de los casos ya que los algoritmos y datos ***presentan regularidades***, que pueden ser detectadas por estos predictores. Estas técnicas de predicción se clasifican en, ***predicción estática*** esta es la técnica más sencilla, predice basándose solo en información de la instrucción de la siguiente manera, si la dirección de salto(destino) es menor que la instrucción de salto se ejecuta siempre, ya que este tipo de saltos se utiliza en bucles, si pasa lo contrario no se ejecuta siempre, ya que este tipo de saltos se utiliza generalmente en instrucciones condicionales(if then else), para implementar esta técnica se incluyen las condiciones de la fig. 5 en la unidad de salto, este tipo de predictores se utilizan en procesadores más sencillos; ***predicción dinámica***, un predictor dinámico trabaja en tiempo de ejecución intentando aprender el comportamiento de programa para predecir si un salto es tomado o no, para realizar esto se incluye una tabla de historia de saltos(BHT) la cual es una memoria de doble puerto que sirve para almacenar la información sobre los saltos más importantes del programa y generalmente se integra con la BTB, para predecir el resultado de un salto se utiliza una máquina de estados la cual sigue el diagrama de estados de la figura 6, actualizando el estado actual de la predicción de un salto, luego de la ejecución del mismo, el rendimiento varía según la cantidad de bits que utilicemos para llevar los resultados de cada salto, el diagrama en bloque de este predictor lo vemos en la fig 7; ***predicción hibrida***, este utiliza en conjunto ambas técnicas ***incluyendo un selector***, que para cada salto decide que técnica da mejores resultados.

```
IF Jump = 00 THEN J = 00                                          Reset
IF Opcode = BEQ AND rs1 = rs1 THEN J = 01; JN = 1
IF Opcode = JAL THEN J = 01; JN = 1                               Predictor estático
IF Opcode = AUIPC THEN J = 01; JN = 1
IF Opcode = BEQ⋁BNE⋁BLT⋁BGE THEN J = 01; JN = 1
IF Jump = 01 AND Branch = 1 THEN J = 10; JR = 1
IF Jump = 10 AND Branch = 0 THEN J = 10; JR = 0
IF Jump = 11 THEN J = 11                                          Excepción
ELSE
      J = 01; JN = 0;                                             Ejecución normal
END
```

Fig. 5, condiciones para un predictor de saltos estático.

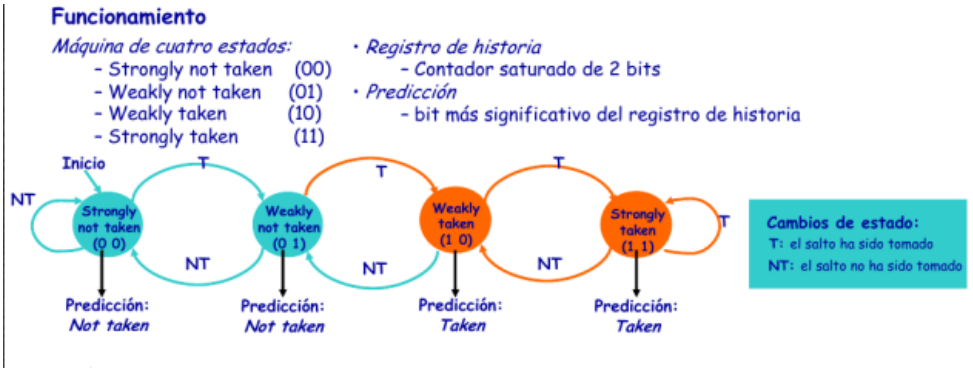


Fig 6. Diagrama de estados de predictor de dos bits.

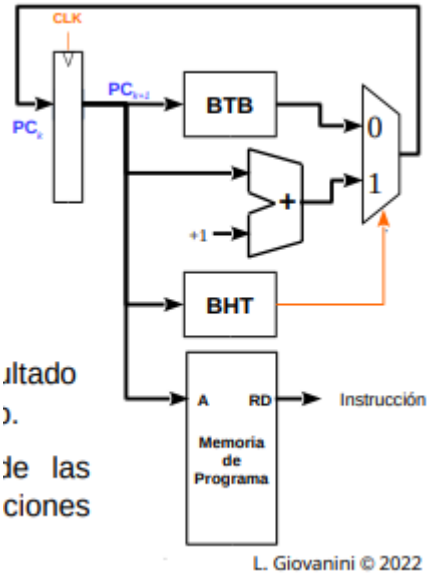


Fig 7. Diagrama en bloque de un predictor dinamico

¿Qué es la ejecución fuera de orden?¿Cómo se puede implementar?

Las políticas de ejecución son reglas que se siguen para poder aumentar el rendimiento de nuestro procesador. Esto se logra cambiando el orden en que las instrucciones son leídas, ejecutadas y los resultados actualizados. Existen diferentes políticas de ejecución que pueden ejecutarse:

- In-Order Issue – In-Order Completion - Las instrucciones son leidas, ejecutadas y los resultados actualizados en el orden en que son ejecutadas en el programa. Es muy ineficiente por las dependencias, las instrucciones deben ser detenidas (stall) y el sistema vaciado (flush)
- In-Order Issue – Out-of-Order Completion - las instrucciones son leidas en el orden en que son ejecutadas en el programa, pero son ejecutadas y los resultados actualizados de acuerdo con la disponibilidad de recursos y las dependencias existentes. Es muy eficiente ya que evita las dependencias de datos, las instrucciones solo son detenidas (stall).
- Out-of-Order Issue – Out-of-Order Completion - Las instrucciones son leidas, ejecutadas y los resultados actualizados de acuerdo con la disponibilidad de recursos y las dependencias existentes. Esta politica tiene como objetivo desacoplar la decodificacion de la ejecución, continuar leyendo y decodificando hasta que la ventana este llena y ejecutar la instruccion cuando la unidad funcional esta disponible.

La ejecución fuera de orden rellena los huecos de tiempo de las instrucciones que están listas para ejecutarse para después reordenar los resultados y aparentar que fueron procesadas de manera normal. Para implementar esta idea se introduce el concepto de planificación dinámica de la ejecución de las instrucciones. Para implementar la ejecución fuera de orden existen dos algoritmos de planificación dinámica:

- Algoritmo de scoreboar, utiliza tablas para mantener un registro de las instrucciones que se recuperan, emiten y ejecutan, para determinar los recursos que se utilizan y se necesitan, y para seguir que instrucción modifica que registro. El algoritmo usa esta información para planificar dinámicamente las instrucciones, determinando cuándo y dónde comienza y termina la ejecución de una instrucción. Para controlar la ejecución de las instrucciones el algoritmo utiliza tres trablas:
    - Tabla de estado de instrucción, esta indica, para cada instrucción que se está ejecutando, en cuál de las etapas se encuentra.
    - Tabla de estado de unidad funcional, indica el estado de cada unidad funcional y mantiene nueve campos
        - Busy, indica si la unidad se utiliza o no.
        - Op, indica la operación a realizar en la unidad.
        - Fi, indica el registro de destino de la operación.
        - Fj, Fk, indican los registros fuentes de la operación.
        - Qj, Qk, indica las unidades funcionales que producirán los registros Fj, Fk.
        - Rj, Rk, son indicadores que muestran cuándo Fj, Fk están listos para y aún no se han leído.
    - Estado de registro, indica para cada registro, qué unidad de función escribirá los resultados

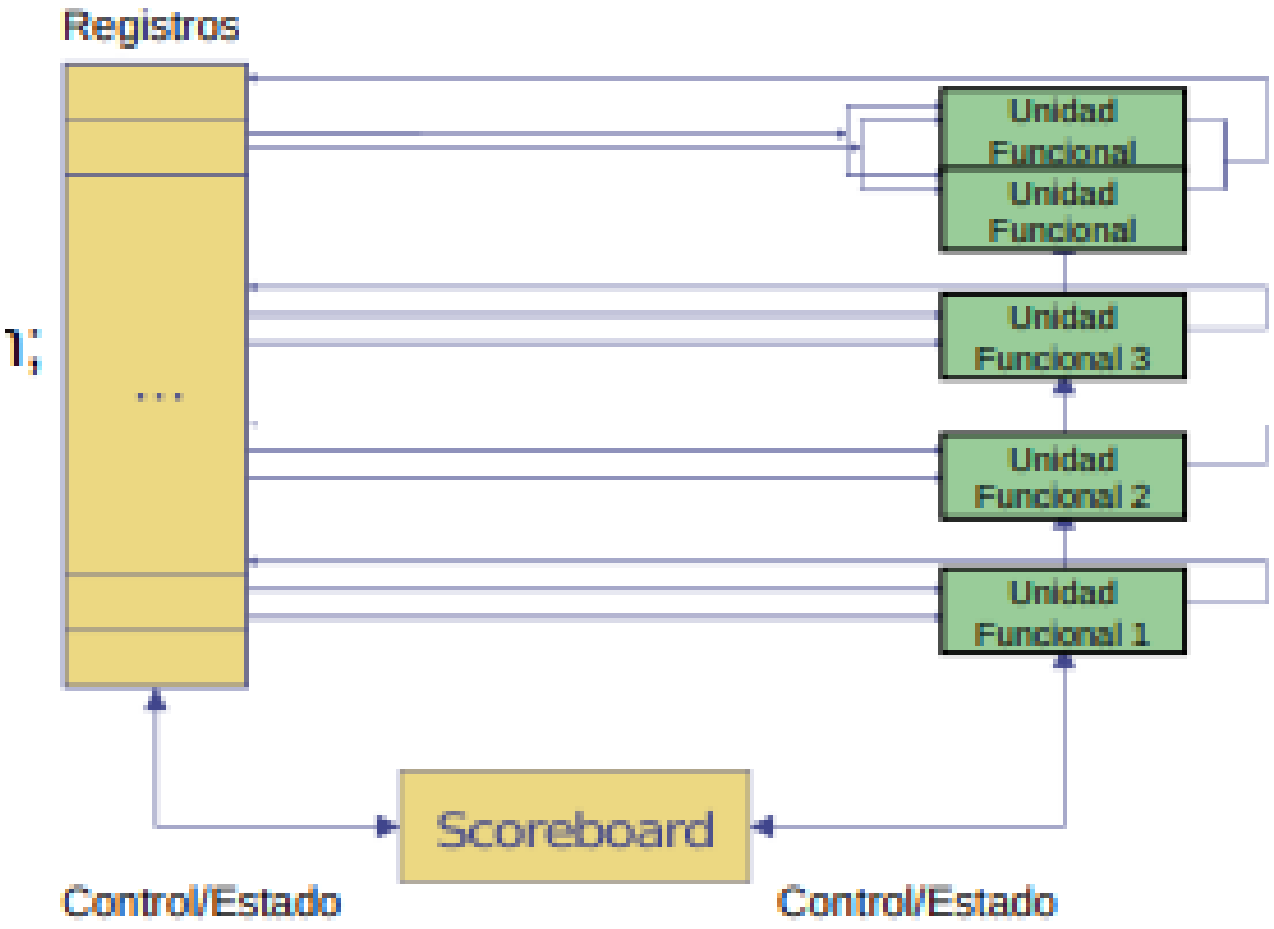


Fig. 8, diagrama en bloque de la implementación del algoritmo scoreboard

- Algoritmo de **Tomasulo**, este aprovecha los recursos de la cpu a travez de la técnica conocidad como renombrado de registros. Esta es una técnica utilizada para eliminar las dependencias falsas de datos que surgen de la reutilización de registros mediante instrucciones sucesivas que no tienen dependencias reales de datos entre ellos. Existen dos formas de implementar el renombrado de registros, una es a travez de una tabla que contiene un archivo de registro, que almacena cuales registros estan siendo utilizados y busca dependencias WAW; y la otra forma es a través de estaciones de reserva. La misma se basa en el uso de registros asociativos, cuando se emite una instrucción a una unidad de ejecución las entradas del archivo de registro correspondientes a la entrada de la cola de emisión se leen y envían a la unidad de ejecucion. El algoritmo de Tomasulo aprovecha el paralelismo a nivel de instruccióna

Explique como funciona una microarquitectura multihilo. Explique cuales son sus ventajas y problemas. ¿Es lo mismo que una microarquitectura multinucleo? ¿En que se diferencian? Fundamente. ¿ Cuales son los elementos fundamentales para el desempeño de una microarquitectura multinucleo? Desarrolle.

Luego de que llegamos a obtener el máximo nivel de paralelismo que se puede aprovechar de la microarquitectura superescalar esta sin embargo tiene un único hilo de ejecución (thread) lo que desaprovecha el paralelismo a nivel de instrucción ya que no se pueden aprovechar todos los recursos disponibles. Entonces la microarquitectura multihilo aprovecha los eventos de latencia prolongada para conmutar entre hilos que estén listos para su ejecución, es decir que el procesador físico ejecuta instrucciones de varios hilos actuando como varios procesadores lógicos. Para poder intercambiar hilos el procesador físico debe poder tener múltiples contextos en el hardware, tener la capacidad de cambiar de contexto y poder ocultar las largas latencias. **Como las instrucciones de cada hilo son independientes se pueden ejecutar multiples instrucciones al mismo tiempo, aprovechando el paralelismo a nivel de hilo.** El único problema de esta arquitectura es que los hilos comparten ciertos componentes como las unidades de ejecución. Las Modificaciones necesarias son:

- Múltiples contadores de programas, registros y un mecanismo mediante el cual una unidad de búsqueda selecciona uno hilo en cada ciclo (política de fetch/issue);
- Una pila de retorno separada para cada hilo para predecir las direcciones de retorno de cada subrutina;
- Mecanismos de fetch/issue de instrucciones, vaciado de la cola de instrucciones y mecanismos de captura para cada hilo;
- Identificador de hilo por cada entrada del buffer de destino de salto (BTB) para evitar la predicción de saltos fantasmas.

Para cambiar la asignación de recursos a cada hilo en cada momento existen dos opciones:

- Multi hilo temporal: Asigna los recursos a cada hilo durante un cierto tiempo, dependiendo de los recursos y dependencias, se ejecuta un solo programa a la vez. Hay dos formas de implementarlo:
    - Multihilo grueso: esta implementación se basa en ejecutar un hilo hasta que es bloqueado por un evento de latencia prolongada y conmutar a un hilo que esté listo para ejecutarse, como puede ser ir a buscar un dato a memoria. Trabajan bien pero su principal deficit es que muchas veces conmutan por eventos de latencias cortas.
    - Multihilo entrelazado, esta implementación se basa en Intercalar la ejecución de diferentes hilos de manera rotatoria, omitiendo los hilos bloqueados. Su ventaja es que elimina las dependencias de datos, ya que los hilos son independientes, pero realentiza

la ejecución de hilos individuales listos para ejecutarse. Se conoce como barrel procesing

- Multihilo simultaneo, esta opción busca ejecutar instrucciones de diferentes hilos en cualquier etapa del datapath simultáneamente. Sus ventajas se dan ya que supera las limitaciones del bajo ILP de un solo hilo, y oculta los riesgos de control. Sus desventajas estan relacionadas con su alta complejidad para llevarlo a cabo y su necesidad de aumentar los recursos(se necesita que el procesador pueda ejecutar instrucciones de varios hilos simultáneamente y que haya una mayor cantidad de registros).

Fig. 8, representación de cómo funcionan las diferentes implementaciones de la microarquitectura multihilo.

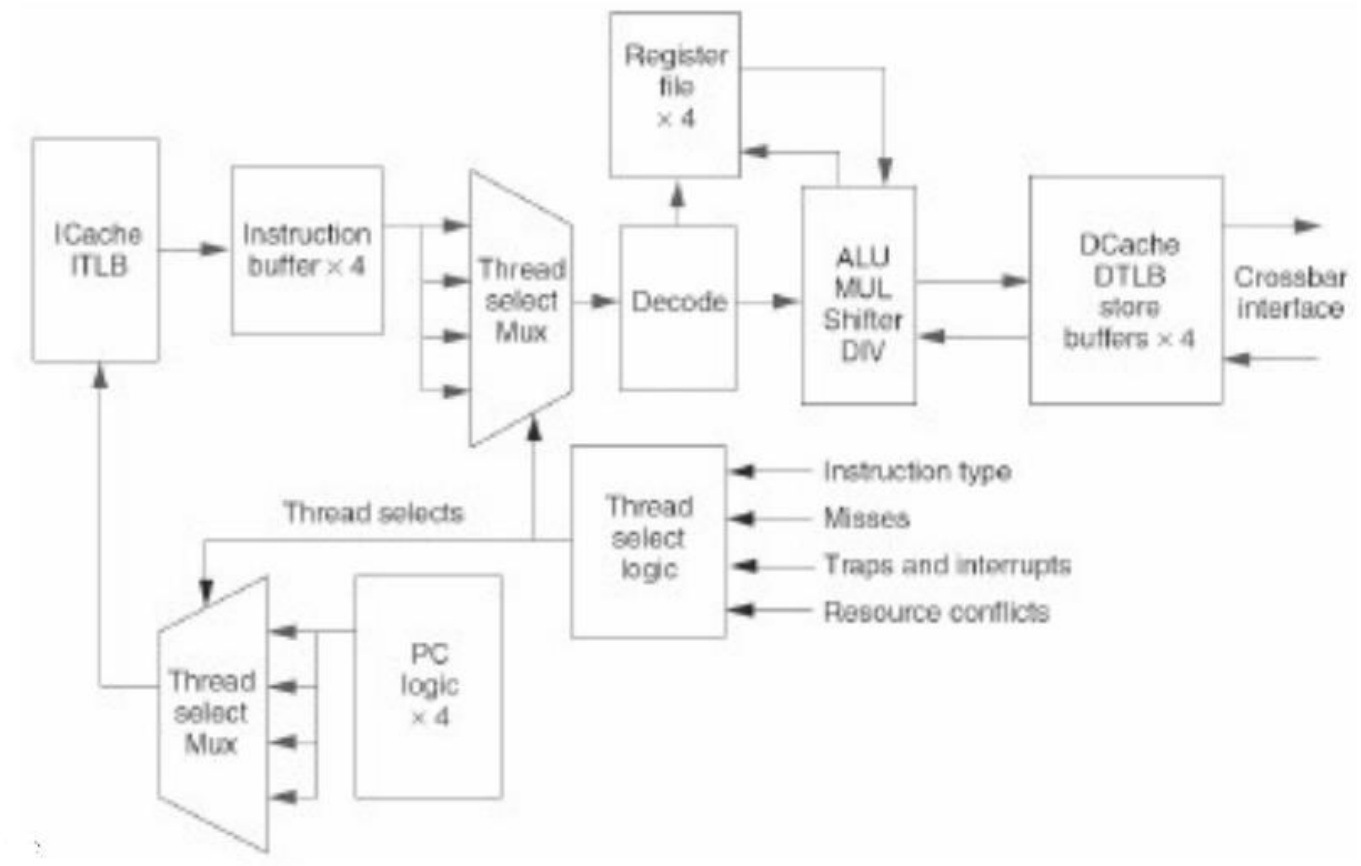


Fig. 8, Implementación de la microarquitectura multihilo.

No es lo mismo que una microarquitectura multinucleo que una multihilo

La arquitectura multinucleo implementa múltiples núcleos en una misma CPU, en cambio la arquitectura multihilo utiliza el mismo núcleo. Un núcleo puede estar acoplado en mayor o menor medida con los demás, pero la ejecución de sus instrucciones son totalmente independientes.

Por otro lado, en una arquitectura multihilo los hilos no son del todo independientes ya que se pueden generar conflictos por la utilización de recursos ya que es una microarquitectura de multiples hilos que comparten un solo nucleo.

Elementos fundamentales de una arquitectura multinucleo

Estos elementos son:

- **La arquitectura de la memoria,** consiste en cómo están implementados los niveles

  L2 y L3 de memoria cache ya que estos mismos pueden ser:

  - Compartida: Una única memoria de nivel L2 que se comparte entre todos los núcleos y tiene contenidos replicados. Esto permite una baja latencia de comunicación entre nucleos pero también aumenta el acomplamiento.
  - Distribuida: Cada núcleo tendrá su propia memoria y el contenido no está replicado. Esto disminuye la latencia de comunicación pero disminuye el acoplamiento.

Las memorias de primer nivel siempre son dedicadas y las de ultimo nivel siempre son compartidas.

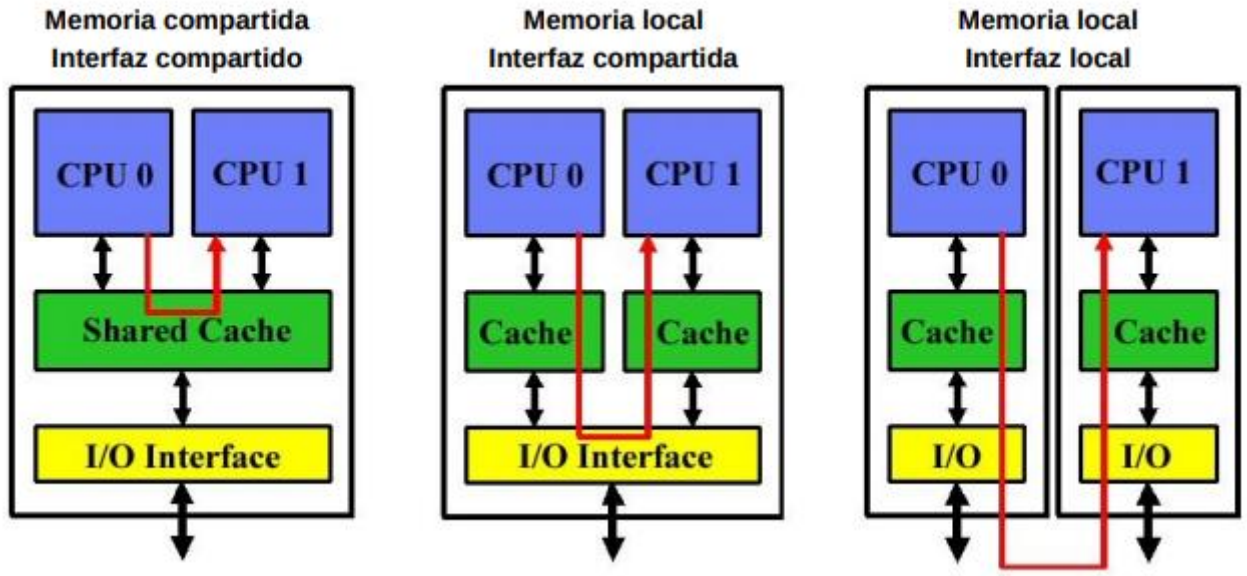


Fig. 9, diferentes formas de compartir memoria.

- **Sistemas de comunicación:** se refiere a como están interconectados los núcleos y en qué modo se utilizan. Indican cuanto tiempo deberá esperar cada núcleo en ir a buscar los datos. Este es el mismo en sistemas simétricos y cambian entre los núcleos para sistemas asimétricos.

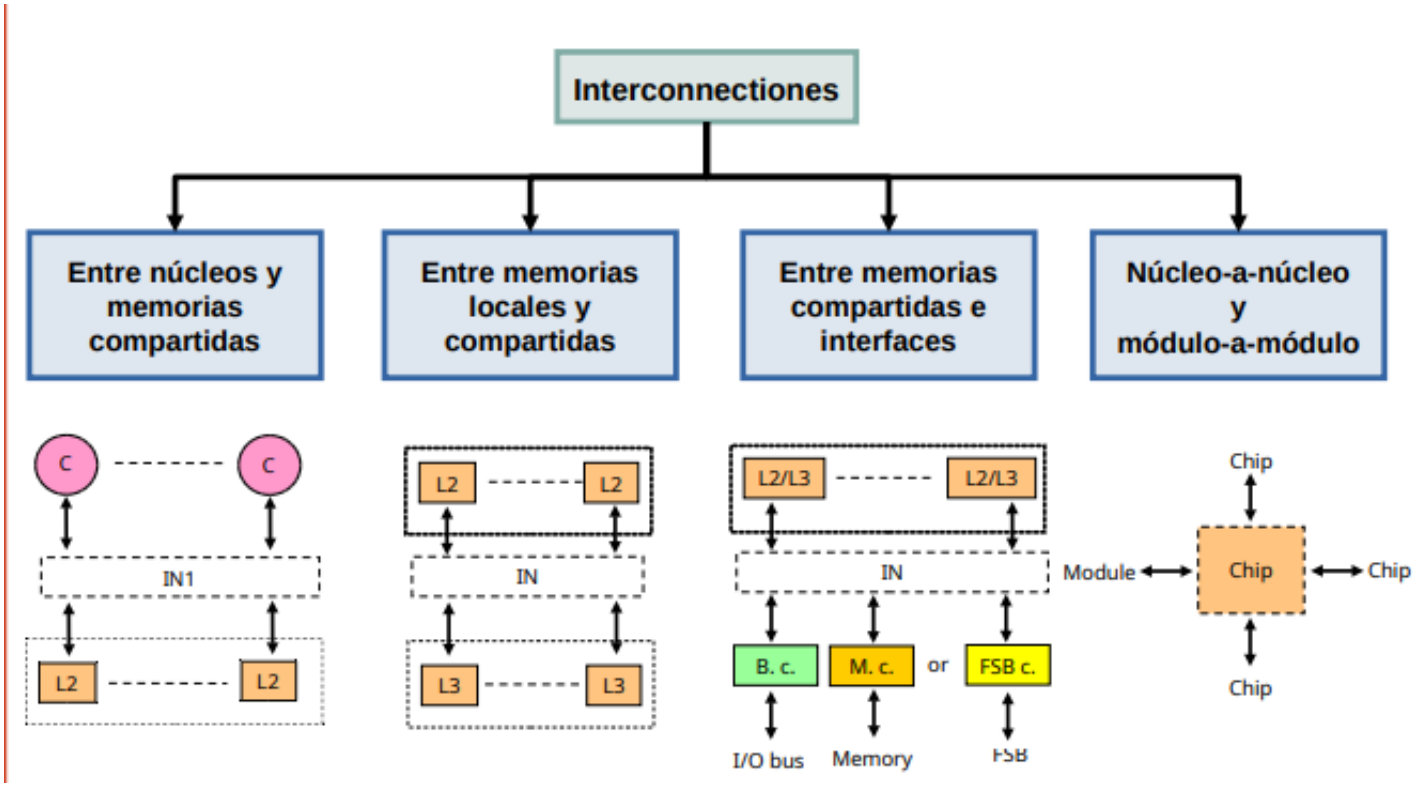


Fig. 10, diagramas de interconexiones.

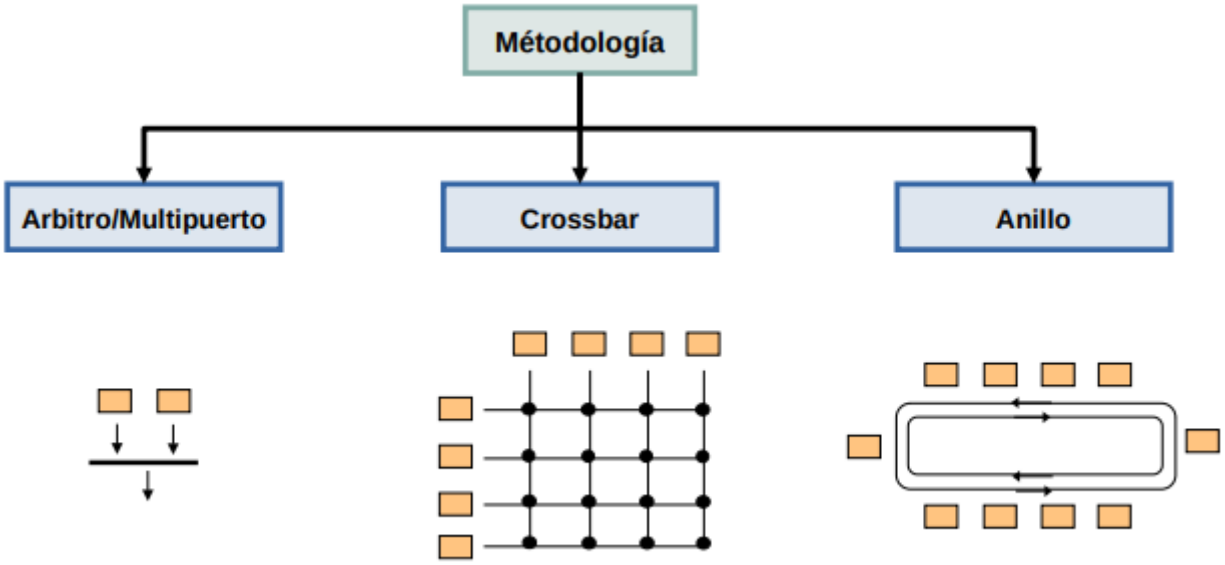


Fig. 10, diagramas de metodologías de conexión entre núcleos.

- **Elementos de procesamiento:** Según los elementos de procesamiento utilizado la microarquitectura puede ser
    - Homogénea, donde todos los elementos de procesamiento serán iguales. Todos los núcleos pueden realizar cualquier tarea
    - Heterogenia: Los elementos de procesamiento serán distintos y existirá un núcleo maestro que les dará las ordenes a los núcleos esclavos.

Los procesadores multinúcleo representan una nueva tendencia en la arquitectura de computadoras que:

- Disminuye el consumo de energía y la generación de calor; λ Minimiza la longitudes de las conexiones y las latencias de interconexión;
- Permite un verdadero paralelismo a nivel de subprocesos con escalabilidad;

Para utilizar todo su potencial, las aplicaciones deberán pasar de un modelo único a uno de subprocesos múltiples, lo que implica que

- Las técnicas de programación paralela y concurrente ganarán importancia.
- Programar los sistemas multinúcleo de manera que permita que las aplicaciones se beneficien del continuo crecimiento en el rendimiento de los procesadores;

Explique cómo funciona la memoria cache de una computadora y para que se la utiliza ¿Cuantos niveles de cache hay en una computadora? ¿Están organizados de la misma manera? Explique los mecanismos de mapeo de datos de una cache ¿Cuáles son sus ventajas y desventajas? Explique cuáles son las fuentes de error en una cache

La memoria cache es una memoria ubicada en el nivel más alto en la jerarquía de la memoria, tiene alta velocidad y pero alto coste económico, y existe para reducir el costo promedio, en tiempo y/o energía, para acceder a la información de la memoria principal. Para cumplir con este objetivo la cache utiliza los conceptos de localidad temporal y espacial, los cuales se refieren a que si un dato es accedido es altamente probable que este mismo sea utilizado nuevamente o que los datos en las direcciones próximas también sean accedidos. Las memorias caches lo que hacen es actuar como un buffer entre la memoria RAM y la CPU, contiene los datos e instrucciones que son solicitados con frecuencia para que estén inmediatamente disponibles si la CPU los solicita. Existen 4 niveles de cache:

- El primer nivel(L1) se ubica cerca del núcleo y se divide en un bloque para los y otro para las instrucciones. Es el mas pequeño, pero rápido tipo de cache, tiene una velocidad cercana a la de los registros.
- El segundo nivel(L2) se ubica también dentro del procesador, pero no se divide en instrucciones y datos, actua como un backup para el primer nivel de cache. En un procesador multinucleo cada nucle tiene una cache de este tipo que no comparte con los demás. Este tipo de memoria es mas lenta que la L1 pero también menos costosa.
- El tercer nivel(L3) se ubica dentro del procesador, no se divide, pero en los procesadores multinucleo funciona como un repositorio o backup compartido entre todos los nucleos. Es la mas grande, la mas lenta y le menos costosa.
- El cuarto nivel(L4) se ubica fuera del procesador, en la memoria principal o en ciertos periféricos como los discos externos.

Existen dos formas de implementar la cache, una es la cache virtual y la otra es la cache fisica, la diferencia entre ellas es que tipo de instrucciones utiliza, es decir, su posición respecto a la MMU.

Existen diferentes tipos de cache, victima (datos de cache eliminados por falta de espacio), trace(instrucciones que se decodificaron), micro operation cache, BTC(se refiere al BTB que almacena las direcciones de salto asociadas a instrucciones).

La cache se direcciona utilizando direcciones parciales de memoria (debido que direcciona solo una parte de cada bloque de memoria), los bits superiores de la dirección indican la correspondencia entre cache y bloques de memoria(un bloque de memoria contiene varios datos a los que se puede acceder, por ello se utiliza el desplazamiento), y el desplazamiento define el dato especifico que se esta buscando.

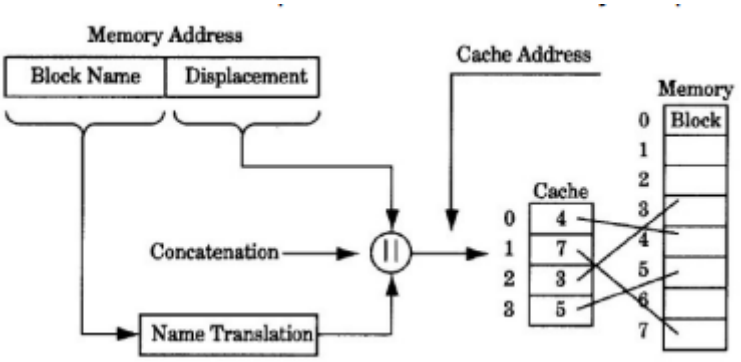


Fig. 10, Diagrama en bloque de la obtención de una dirección de un dato de la cache.

La operación que realiza el procesador para la lectura y la escritura de un dato en memoria siguen los siguientes diagramas

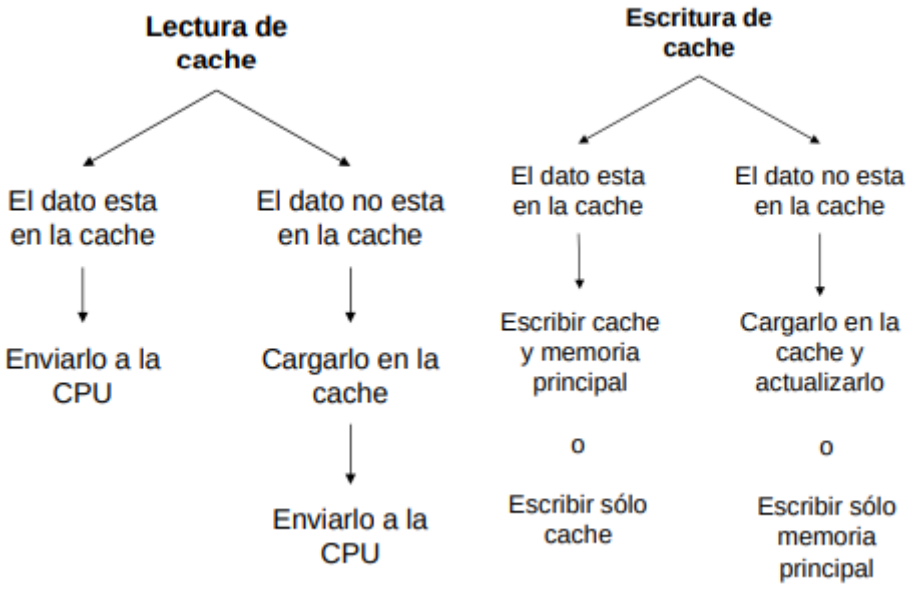


Fig. 11, Diagrama de árbol para la operación de lectura/escritura en memoria con una cache.

Además, el acceso a la cache puede hacerse en paralelo con el acceso a la memoria principal para no perder tiempo.

Por ultimo la cache utiliza funciones de mapeo para determinar como se asignan los bloques de la memoria principal a las líneas de cache, para esto existen dos enfoques para el mapeo

- Mapeo directo, este mapeo obliga a que solo haya un bloque de datos por cada conjunto (recordemos que en memoria pueden haber varios bloques de datos por cada conjunto, ya que un conjunto está formado por varios bloques) solo necesita tener tres datos en memoria, los datos de la cache, la etiqueta para direccionar la cache y el estado del bloque de cache. El problema de este enfoque se da cuando hay más bloques de memoria que líneas de cache, además cuando más de un dato del mismo bloque debe ser almacenado, si la memoria asignada al bloque no tiene un tamaño adecuado, existirá un conflicto por la misma línea de cache. Su ventaja es que no tiene aliasing(varias direcciones virtuales se asignan a la misma dirección física) y permite una predicción

simple y rápida sobre si un dato se encuentra o no en cache, su desventaja es que tiene problemas con la localidad espacial.

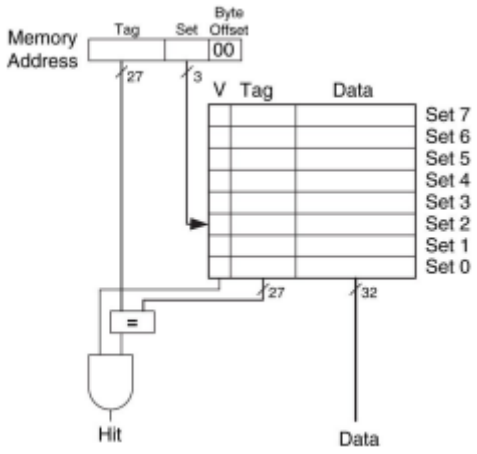


Fig. 12, diagrama en bloque de como funciona el mapeo directo en una cache.

- Mapeo asociativo, el enfoque de memoria cache por mapeo asociativo supone que hay varios bloques por conjunto, por lo que contiene una entrada por cada bloque de datos en cada conjunto, permitiendo tener localidad espacial (recordemos que los bloques de un mismo conjunto están ubicados en direcciones de memoria cercanas). Mientras más aumentamos la cantidad de bloques que puede contener un mismo conjunto en la memoria cache menos fallos tendrá. Un memoria cache asociativa prueba todas las vias simultáneamente, es decir busca el dato en todos los bloques de un conjunto al mismo tiempo.

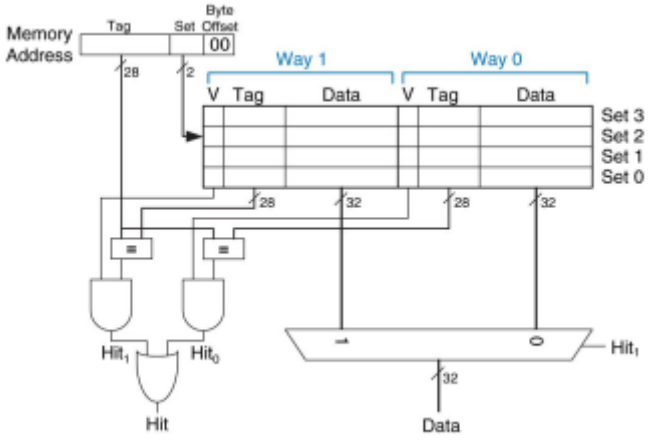


Fig. 13, diagrama en bloque de cómo funciona el mapeo asociativo con dos vías en una cache.

En la cache se pueden dar ciertos errores, los cuales son intentos fallidos de leer o escribir una parte de los datos en el cache, lo que resulta en un acceso a memoria principal con una latencia mas alta. Se pueden clasificar según sus causas:

- Error de arranque en frio, ocurren en el primer acceso a un bloque, este debe ser llevado a la cache. A este fallo se lo llama error de primera referencia.
- Error de capacidad, ocurren porque los bloques se descartan de la memoria caché debido a que no puede contener todos los bloques necesarios para la ejecución del programa, es decir que el

conjunto con el que se trabaja tiene mas bloques, que las vias (ways) disponibles en la cache (el conjunto de trabajo del programa es mucho mayor que la capacidad de la memoria cache).

- Error de conflicto, ocurre porque varios bloques se asignan al mismo conjunto de bloques. También se lo conoce como fallo de colisión o fallo de interferencia.

Cambiar los parámetros de la memoria caché puede afectar a uno o más tipos de fallas. las fallas que aparezcan en un cierto nivel de cache se pueden compenzar con el siguiente nivel de cache, ya que cada nivel de cache tiene una probabilidad de fallar diferente según la cantidad de memoria que posea(mirar fig 13).

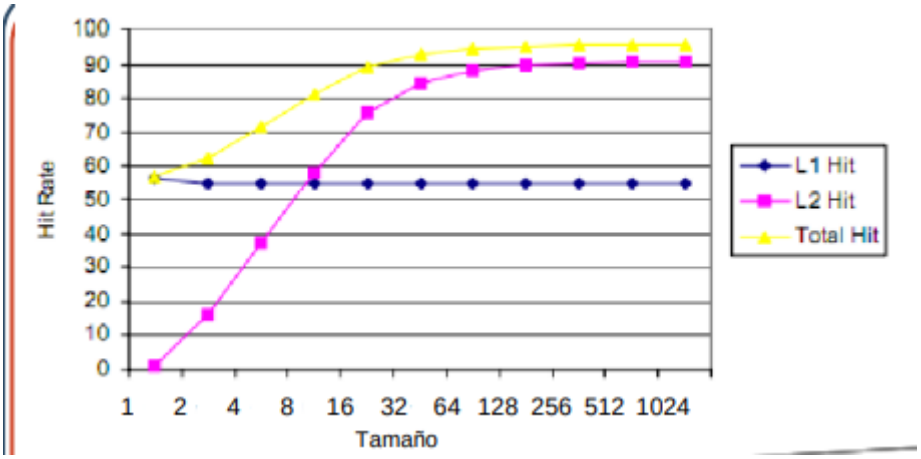


Fig. 13, grafico de cantidad de fallas combinando los dos primeros niveles de cache.

Por ultimo cuando la cache asignada a un conjunto (línea de cache) está llena, se debe tomar una política de remplazo, existen varias estrategias, de las cuales las más usadas son:

- LRU, descarga la que menos se han usado recientemente primero utilizando bits de edad para cada bloque de una línea.
- TLRU, introduce el tiempo de uso de un bloque eliminando los menos utilizados.

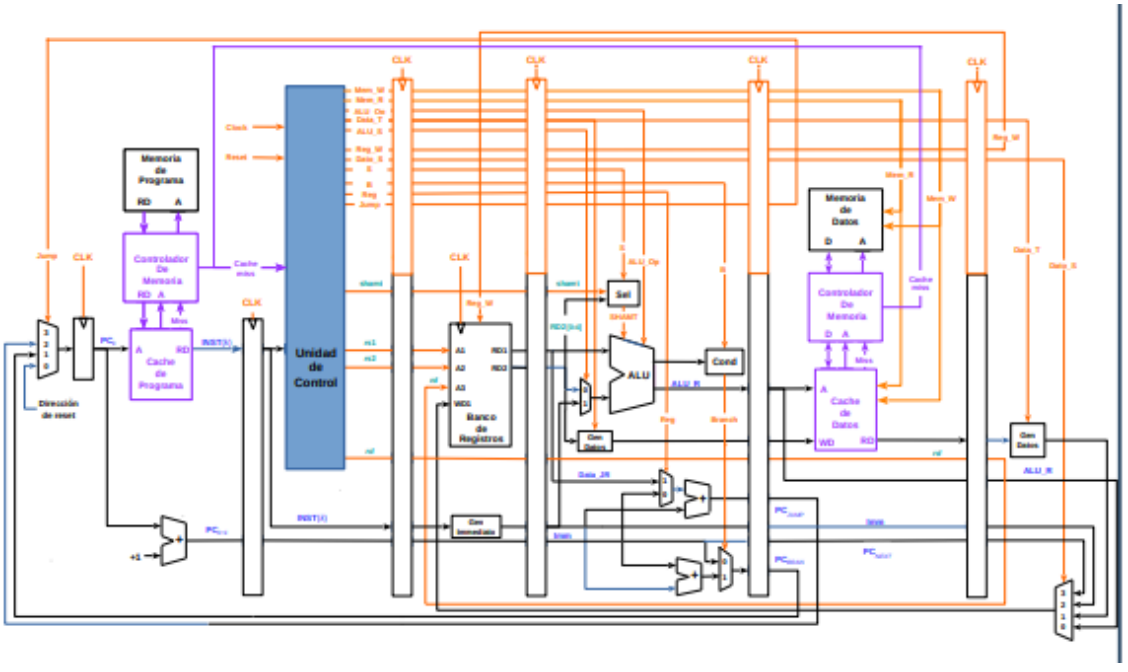


Fig. 14, Diagrama en bloque de arquitectura segmentada con cache.

Explique ¿cómo funciona la gestión de periféricos? Explique cómo funciona la gestión de periféricos por interrupciones. ¿cuales son sus ventajas y problemas? ¿Cómo se gestionan las interrupciones anidadas? ¿Es lo mismo una interrupción que una excepción? Explique en que se diferencian. ¿Para qué se utilizan las Excepciones?

para comenzar, un periférico se comunica con los componentes a travez de un conjunto de líneas de que transportan señales, conocidas como buses del sistema(existen buses para datos, señales de control y direcciones), estos son sistemas de comunicación y fueron mejorando con el tiempo, existieron tres generaciones que fueron aumentando la complejidad, flexibilidad y rendimiento(hoy en dia se utilizan buses de tercera generación, los cuales tienen mucha flexibilidad ya que permiten conectar buses internos como de maquinas entre si, pero aumenta mucho la complejidad, ya que se deben atender solicitudes con requerimientos muy diferentes), estos buses deben ser gestionados de alguna forma ya que llevan muchos datos importantes. Debido a esto surge el concepto de arbitraje de bus, el cual refiere a la gestión de buses del sistema durante su operación, existen dos enfoques para esta gestión.

- Centralizado, existe un arbitro único para todo el bus del sistema, este logra mayor eficiencia, pero es poco robusto, si falla el arbitro falla todo el sistema.
- Distribuido, cada dispositivo puede gestionar el bus, este logra mayor robustes, pero menor rendimiento ya que la gestión del bus queda asignado entre varios árbitros.

Luego la gestión de periféricos se refiere a como se gestionan justamente las transferencias de datos entre los elementos de la arquitectura, los periféricos encargados de las E/S del sistema y viceversa, existen 3 mecanismos, gestión programada, gestión por acceso directo y gestión por interrupciones.

- Gestión programada, en este mecanismo, el procesador es el que tiene control directo sobre la operación de transferencia con los periféricos, por lo tanto, las tareas necesarias para atender los periféricos se organizan en el programa, es decir el procesador se encarga de recibir los datos de los periféricos, cargarlos en memoria, etc. Este mecanismo tiene mucha flexibilidad en la gestión, ya que se tiene control total sobre el proceso, mejorando su respuesta ante eventos sincrónicos, por otro lado, posee altos tiempos de latencia, necesita mayor programación y no puede atender eventos asincrónicos. Suele utilizarse para sistemas dedicados, en los que se debe cumplir tareas específicas.
  Los pasos ejecutados durante una gestión programada son:
  1. El procesador direcciona un dispositivo y solicita una operación;
  2. El dispositivo direccionado realiza la operación;
  3. El dispositivo activa los bits de estado y espera a una nueva operación;
  4. El procesador comprueba periódicamente el estado de esos bits, hasta que detecta que la operación fue completada;
  5. En caso contrario procesador espera y vuelve a comprobarlo más tarde.

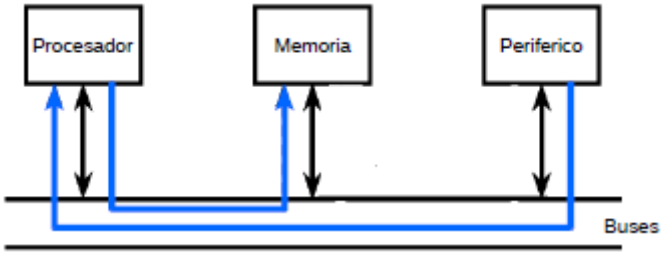


Fig 7. Representación de la gestión programada.

Gestión por Acceso directo, es mecanismo que busca liberar al procesador durante la transferencia de los datos, esto lo hace permitiendo a los dispositivos comunicarse con la memoria a través de una unidad dedicada a controlar los accesos directos a memoria(DMAC), sin la que se sobrecargue el procesador. Aunque no se necesite a la CPU para la transacción de datos, se necesita el bus del sistema por lo que existen diferentes estrategias para regular su uso, permitiendo así que no quede totalmente acaparado por el controlador. El periférico avisa al controlador de DMA, la CPU programa al controlador, el cual toma control de los buses y transfiere los datos, mientras el DMAC está operando, la CPU debe esperar a que finalice dicha tarea sin ejecutar ninguna instrucción. Para mejorar el rendimiento de este mecanismo se incluye una memoria cache que permite a la CPU seguir trabajando mientras el DMAC mantiene ocupado el bus. La DMA puede llevar a problemas de ***coherencia de cache*** si tenemos una CPU equipada con memoria caché y una memoria externa que se pueda ser accedidas directamente por los dispositivos que utilizan DMA. La secuencia de operación es la siguiente:

1. El procesador inicializa el DMAC programando AR y WC;
2. El periferico realiza una petición de DMA al DMAC;
3. El DMAC le responde con una señal de aceptación;
4. El DMAC activa la línea de petición de DMA al procesador;
5. Al final del ciclo del bus en curso, el procesador pone las líneas del bus del sistema en alta impedancia y activa la sesión de DMA;
6. El DMAC asume el control de los buses de direcciones y control;
7. El periferico transmite una nueva palabra de datos al registro temporal del DMAC;
8. El DMAC ejecuta un ciclo de escritura en memoria para transferir el contenido del registro temporal a la posición M[AR];
9. El DMAC decrementa WC e incrementa AR;
10. 10.El DMAC libera el bus y desactiva la línea de petición de DMA;
11. El DMAC compara WC con 0

11.1 Si WC > 0, se repite desde el paso 7;

11.2 Si WC = 0, el DMAC se detiene y envía una interrupción al procesador

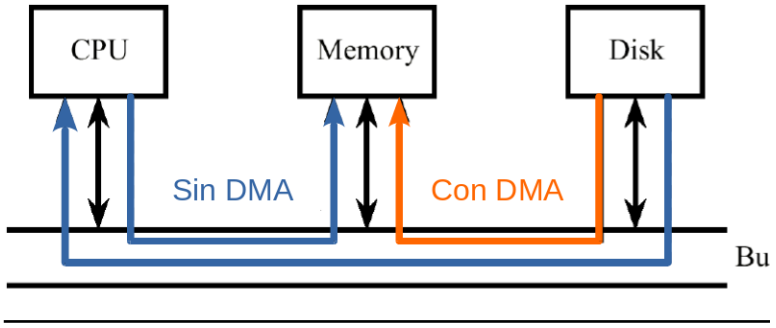


Fig 8. Representación de la gestión por acceso directo.

- gestión por interrupciones, este mecanismo utiliza señales provenientes de hardware o software que indican al procesador que un evento de alta prioridad requiere la ***interrupción del código*** que se está ejecutando, es decir frenan la ejecución de un programa para ejecutar otras rutinas de mayor importancia. Las interrupciones se pueden clasificar según su ***mecanismo de activación***, cuando se producen por errores durante la ejecución de una instrucción o fallas en el hardware, se les llama ***excepciones*** (son síncronas con el procesador) y pueden ser de origen preciso(se conoce su causa) o impreciso (cuando no se conoce), Para implementarlas, se agregan etapas extras de detección de excepciones en el ciclo de instrucción; cuando las interrupciones son producidas por eventos externos al procesador, generalmente relacionadas con periféricos que requieren atención, se les llama ***interrupciones por hardware*** (IRQ);cuando son producidas por el programa en ejecución se les llama ***interrupción por software***(TRAP), para generarlas existen ciertas instrucciones en el código máquina. Para implementar una interrupción, ya sea generada por software p hardware, se agrega una etapa de detección de interrupciones al final del ciclo de

instrucción, si se detecta una el procesador suspende la ejecución de la próxima instrucción, guarda la dirección de la próxima instrucción a ejecutar, almacena la información necesaria para continuar su ejecución y carga el contador de programa con la rutina de atención de la interrupción. Las interrupciones por hardware pueden ser activadas de diferentes formas, por ***nivel*** (es una interrupción que se activa cambiando el nivel de la línea de interrupción correspondiente a un periférico hasta que sea atendida), por ***flanco*** (es una interrupción que se activa cambiando y manteniendo el nivel de la línea de interrupción hasta que sea atendida)y también puede activarse por ***mensajes***(cambio en el estado de presencia de un estado, generalmente utilizada para comunicación entre dispositivos). Existen diferentes formas para identificar la fuente de una interrupción:

- ***consultas***, este método funciona de manera similar a la gestión de periféricos programada, el procesador identifica la fuente de manera sistemática comprobando todos los dispositivos. Para implementarla, la computadora incorpora registros que guardan la información de las peticiones de interrupción (Interrupt Flag). Lleva poco hardware y es muy flexible, pero es muy lenta debido a que debe comprobar todos los dispositivos.
- ***Vectorizadas***, este mecanismo asocia cada interrupción con un vector utilizado para sincronizar el procesador con el periférico, estos vectores se almacenan en un espacio reservado de la memoria, llamado tabla de vectores. Este método tiene la ventaja de poder procesar multiples interrupciones con poco hardware, pero a su vez tiene mucho acceso a memoria por lo que resulta lenta y utiliza mucho los buses del sistema.
- ***Hardware dedicado,*** se utiliza un circuito dedicado a gestionar las interrupciones cuyas señales de petición se controlan de forma independiente para cada periférico. Esto lo hace rápido, eficiente, veloz y flexible, pero también mas costoso de fabricar.

Una vez que le hemos dado una prioridad a las instrucciones, existen diferentes métodos para gestionar la ejecucion de interrupcion:

- ***Interrupciones anidadas,*** en un sistema que permite las interrupciones anidadas, las interrupciones se pueden ejecutar en cualquier momento y lugar, si se esta ejecutando una y llega una petición para ejecutar otra interrupcion de mayor prioridad(ISR), esta se atenderá inmediatamente, luego la segunda de mayor prioridad y asi.
- ***Inhibición de interrupciones,*** esto consiste en desactivar una interrupcion en momentos en los que el procesador esta ejecutando rutinas criticas que no puede interrumpir, aun asi existen cierta interrupciones(interrupciones no enmascarables) que no pueden ser deshabilitadas, por lo que son atendidas obligatoriamente.

Por ultimo las rutinas de servicio están almacenadas en alguna parte de la memoria de instrucciones, para determinar la direccion de estas existen dos métodos:

- ***Direcciones fijas***, se hallan cableadas en el procesador, por lo que no pueden cambiar.
- ***Direcciones variables,*** no se hallan cableadas, sino que se utilizan punteros para poder direecionar la rutina(direccionamiento directo) o un vector con la informacion de la misma y su direccion(direccionamiento indirecto).

Las etapas de la ejecución de una interrupción son:

1. Completar la ejecución de la instrucción en curso (si se puede);
2. Determinar la fuente de la interrupción;
3. Analizar si se atiende, o no, la interrupción;
4. Si se atiende, salvar la información del programa en ejecución;
5. Determinar la dirección de la rutina de atención y ejecutarla;

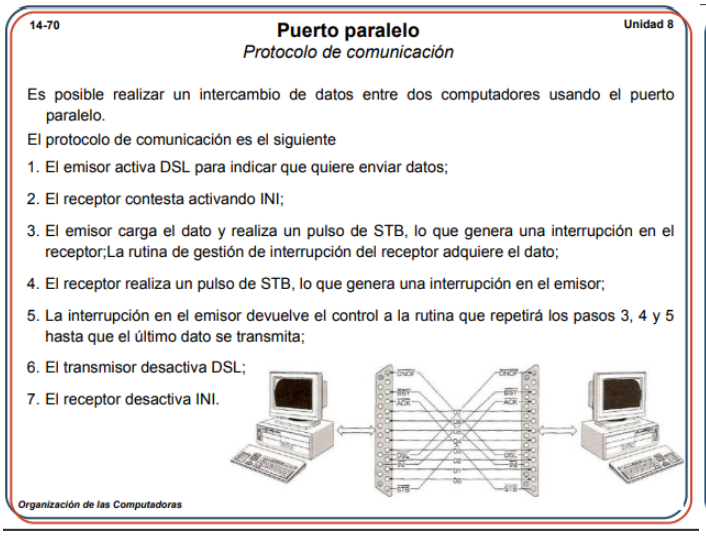
6. Una vez finalizada la ejecución de la rutina de interrupción reasumir la ejecución del programa

Estos mecanismos tienen la ventaja de que el procesador ejecuta otro código mientras se espera que ocurra un evento, pero al ser variable la cantidad de eventos, se dificulta el cálculo de tiempos de ejecución, y es complejo la gestión de interrupciones simultaneas. La ***gestion de interrupciones simultaneas*** requiere que prioricemos la ejecucion de unas sobre otras, ya que hay algunas que son mas urgentes y requieren ser tratadas al inicio. Por ello existen diferentes mecanismos para solucionar este problema

- ***Hardware especifico***, este seria un dispositivo que combine varias fuentes de interrupcion en una o mas líneas, al tiempo que permite asignar los niveles de prioridad.
- ***Consulta***, esta técnica consiste en gestion por software, es decir existe una rutina que debe encargarse de determinar el valor de cada interrupcion en base al dispositivo que la genera.
- ***Conexión en cadena,*** este método de gestión conecta en serie cada periférico, formando una especie de cadena conocida como Daisy chain, la prioridad de cada dispositivo estará dada por su posición en la cadena.

Explique ¿qué es y cómo funciona un puerto serie? Explique cuáles son sus ventajas y desventajas respecto al puerto paralelo. Explique la principal diferencia entre un puerto serie síncrono y uno asíncrono. ¿Cuáles son los mecanismos de un puerto asíncrono para lograr la sincronización entre emisor y receptor? ¿Qué hace la UART?

El ***puerto paralelo*** es una interfaz de comunicaciones entre un computador y un periférico que permite enviar y/o recibir datos en ***paquetes de bits***, generalmente en formato de 8 bits o un byte(no quiere decir que solo se transmitan 8 bits en un puerto ya que también hay señales de control y estado), un puerto paralelo está compuesto de tres registros (datos, estado y control) de 8 bits cada uno, los cuales permiten acceder a información sobre el periférico, gestionar su funcionamiento e intercambiar datos con el mismo, la capacidad de este puerto de enviar bits en paralelo lo hace rápido pero más costoso, además tiene predisposición a sufrir errores en distancias largas.



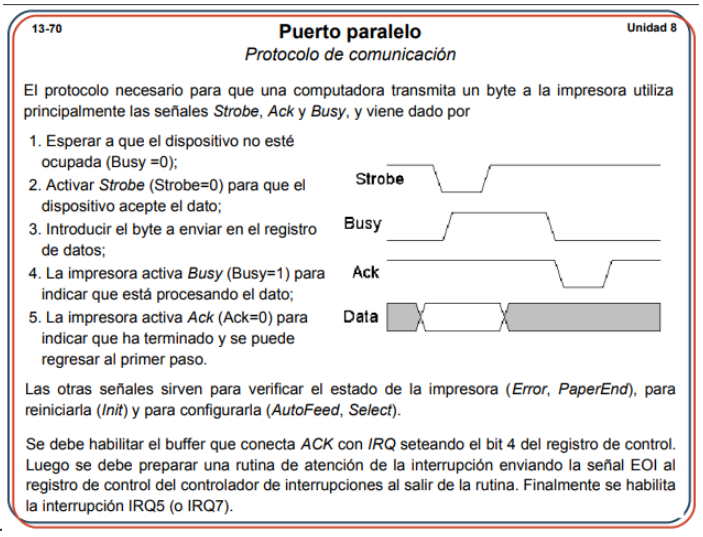


Fig. 10, protocolos necesarios para la comunicación paralela ***CPU-Impresora*** y ***CPU-CPU***

Por otro lado, el puerto serie es una interfaz de comunicaciones que transmite datos digitales ***un bit a la vez***, a diferencia del puerto paralelo que transmite varios bits simultáneamente. El término puerto serie generalmente identifica al hardware compatible con el estándar ***RS-232***. Los datos deben ser preprocesador antes de la transmisión, lo que lo hace más versátil para una variedad de dispositivos y aplicaciones. Existen dos formas de transmitir datos en un puerto serie

- ***La trasmisión serial síncrona*** es una manera de transmisión serial en la que el receptor y el emisor están sincronizados con el mismo reloj, en esta se intercambian una o varias señales de control junto con los paquetes de información. Para sincronizar el emisor y receptor, Puede existir una señal de reloj que controle la sincronización o esta se puede realizar a través de un protocolo con estructuras de información predeterminadas. La transmisión es controlada por el emisor y se transmite entre dos grupos de datos denominados delimitadores (datos y señales de control). La ventaja de esta transmisión es su alta velocidad de transmisión y su regularidad con los datos (se envían datos constantemente), por otro lado, los equipos requeridos para llevarla a cabo son más complejos y costosos.
- ***La transferencia serial asíncrona***, Una transferencia asíncrona ocurre cuando no hay relación temporal entre los relojes del transmisor y el receptor. La sincronización se realiza en cada dato transmitido y a través de estructuras de datos. El ritmo de presentación de la información en el receptor no tiene que coincidir con el del transmisor. Es necesario que los datos contengan información de temporización. Para implementar un sistema asíncrono es necesario acordar ciertos parámetros de señalización como el tipo de operación a realizar, el tamaño de dato y el orden, la velocidad de transmisión (en baudios), el código de detección corrección de errores utilizado y la estructura de información dedicada a la sincronización. El proceso de la transmisión serial toma datos organizados en bytes y los transmite como una secuencia de bits. En el destino los bits de datos transmitidos son reorganizados en bytes. Cada unidad contiene un registro de desplazamiento para conversión serie-paralelo. La transmisión se realiza a nivel de bloques de datos(SDU), existe un bit de comienzo que indica el inicio de un SDU, uno o dos bits de final que indican que el SDU ha terminado, y se pueden añadir bits para control de errores. La gran ventaja de esta transmisión es su bajo costo para implementarla, además de su flexibilidad frente a transmisiones con flujo de datos irregular, por lo que puede conectarse con una mayor variedad de dispositivos.

Para implementar la transmisión en serie se incluye una unidad llamada ***USART***, que es un dispositivo de ***hardware*** de computadora para la comunicación en serie en el que el formato de datos y velocidades de transmisión son configurables. Una USART es un ***emisor/receptor*** programable cuyas funciones son:

- ***Conversión paralelo-serie***, serializar datos internos para transmitirlos a través de una línea serie.
- ***Conversión serie-paralelo***, recibir transmisiones serie y restituir los datos a su forma original.
- ***Generación del reloj*** para la transmisión y recepción de los datos.
- ***Generación y validación*** de los protocolos de transmisión.

La ***USART*** tiene 4 bloques:

- Bloque de transmisión, este se compone de un registro de desplazamiento encargada de serializar el dato para su transmisión, además de un circuito combinacional que se encarga de tomar el dato a transmitir, agregarle los bits de comienzo, fin y paridad, y por ultimo transmitirlo al receptor con la

velocidad de emisión adecuada. Para realizar la sincronización del dato recibido se debe comprobar cada bit en la mitad del intervalo del tiempo que dura para evitar la lectura de falsas transiciones producto del ruido en la línea.

- Bloque de recepción, este se encarga de recibir el dato a una cierta velocidad, se compone de un registro de desplazamiento para paralelizar el dato recibido, además se encarga de detectar y extraer los bits de inicio y fin. Para realizar la sincronización del dato recibido se debe comprobar cada bit en la mitad del intervalo del tiempo que dura para evitar la lectura de falsas transiciones producto del ruido en la línea.

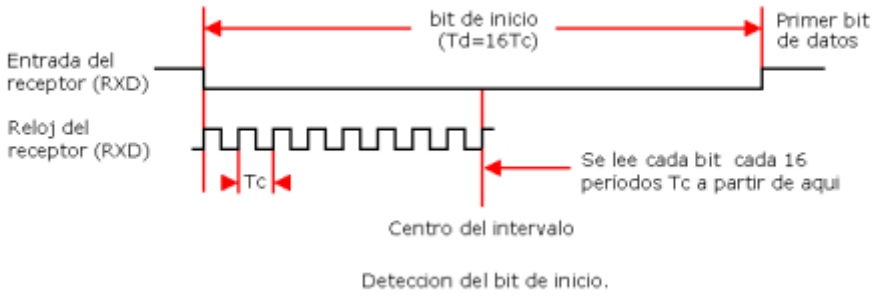


Fig 9. Diagrama explicativo sobre cómo se leen los bits recibidos en la UART.

- Bloque de temporizador, este utiliza un frecuencia que proviene de un oscilador externo y la divide por el contenido del registro divisor, es decir se encarga de generar las velocidades de la USART.
- Bloque control de modem, este utiliza lógica de control del módem, la cual se encarga de gestionar las señales de los protocolos de comunicación RS.