**RESUMEN ARQUITECTURA DE COMPUTADORAS**

**ENTRADA / SALIDA**

Los dispositivos de E/S son siempre más lentos que la CPU y la RAM, por lo que se diseñaron distintos sistemas para que esta lentitud afecte lo mínimo posible al sistema.  
  
Por esta razón se crearon los módulos de E/S, dicho modulo se comunica con los periféricos y la CPU mediante el puerto de E/S, por dicho puerto mandan la información de **control** (entrada, entrada/salida o salida), **estado** (listo / ocupado) y **datos**. Así mismo el módulo de E/S posee un **control lógico** (para el direccionamiento de datos), un **transductor** (para conversión de datos) y un **buffer** (para la retención y adaptación de 1, 8, 16, Etc. bits).

Por lo tanto estos módulos requieren realizar operaciones de direccionamiento, transferencia de datos y gestión de transferencia.

**Metodos de conexión E/S**

**E/S Asignada en Memoria** (memory-wraped)  
 Las datos de E/S se encuentran en la memoria, es decir la entrada de datos, mediante el bus de direcciones, esta mapeada directamente a la memoria. No se distingue la parte de E/S de la parte de memoria, por lo que se pueden usar todo tipo de instrucciones. La desventaja es que ocupa memoria.

**E/S Aislada** Las posiciones de E/S están separadas de la memoria del sistema por lo que se necesitan líneas especiales para memoria y para E/S e instrucciones para acceder a la memoria E/S (in, out, mov). Esto da la ventaja de dejar totalmente libre la memoria del sistema.

**Técnicas de gestión de transferencia E/S**

**E/S programada con espera de respuesta**

En este caso la CPU tiene el control completo de E/S. En esta técnica la CPU se encarga de hacer continuamente consultas de estado para ver cuando el dispositivo está listo para recibir datos, es decir que la CPU quedaría en un estado ocioso hasta que el periférico ese disponible.

**E/S con interrupciones**

El CPU únicamente envía un pedido, posteriormente el modulo E/S envía un pedido de interrupción a la CPU cuando esté listo el pedido. Mientras tanto la CPU puede seguir ejecutando otras instrucciones.

**E/S con acceso directo a memoria (DMA)**

Este tipo de gestión tiene la ventaja de que pude controlar la transferencia de datos a memoria directamente gracias al DMAC (DMA controller), sin el uso de la CPU. El DMAC tiene que ser capaz de solicitar el uso del bus, especificar la dirección de memoria y configurar el control (tipo de operación y señales de sincronización).

**Evolución de dispositivos E/S:**

- CPU controla periféricos.

- Se crea un dispositivo de E/S.

- Se le añade las interrupciones.

- Se crea el DMA.

- Se le añade al DMA un procesador (CDMA).

- Se le agrega una memoria local.

Se termina transformando en una mini computadora.

**Canales de E/S:**

Son una extensión del DMA, con la completa capacidad y autonomía de ejecutar instrucciones E/S.

Es decir:

- CPU ordena inicio de ejecución de programa en memoria.

- Programa indica dispositivos, áreas de memoria, prioridades y acciones.

- El Selector o Multiplexor se encargar de todo lo relacionado E/S

**Tipo de Canales:**

**Selector:** Controla varios dispositivos uno por vez, dicho canal va a transferir la totalidad de un dispositivo el cual va a ser manejado en su totalidad por el controlador o módulo de E/S.

**Multiplexor:** Puede manejar E/S con varios dispositivos simultáneamente. Posee 2 formas multiplexor de bytes (acepta y transmites caracteres) y multiplexor de bloques (intercala bloques de datos de distintos dispositivos).

**INTERRUPCIONES**

Se consideran interrupciones a las generadas por los dispositivos de E/S, estas tienen distinto nivel de prioridad dependiendo de c/u.

**Excepciones / Traps:**  
 Son interrupciones pero de alta prioridad y generadas por el CPU, dichas interrupciones se generan cuando hay un error importante como falla de programa, de hardware o acceso a datos de memoria.

**Ciclo de instrucción:**

**-Captación de Instrucción.**   
 Al finalizar una instrucción se comprueba los pedidos de interrupción, en caso de que haya algún pedido se guarda en memoria los datos del programa que se venía ejecutando (Dir. Prox. Instrucción y estado de CPU) y se carga la Dir. de comienzo de la interrupción. Se inhiben otras interrupciones.

-Ejecución de Instrucción  
 Se ejecuta la interrupción y en su finalización se reactivan las interrupciones.

**-Gestión de Interrupción:** En caso de que haya más de un pedido de interrupción, estos se van ejecutando en orden de prioridad y cuando se hayan ejecutado en su totalidad se vuelve a la ejecución del programa principal. Para esto se carga los datos previamente guardados (Dir. de la próxima instrucción y estado del CPU).

**Información extra:**

Así mismo las interrupciones se pueden ir teniendo diferentes clasificaciones.

-Por como procesan las interrupciones que puede ser Procesamiento de interrupciones secuenciales o con procesamiento de interrupciones anidadas (Secuencial).

-O por cómo se reconoce el periférico que solicito la interrupción:

Por Interrupciones multi-nivel: Es muy caro pero consiste en que cada dispositivo que puede generar una interrupción tiene una entrada física al CPU, cada entrada tiene un nivel de prioridad con el cual puede o no interrumpir a otra interrupción.

Línea de interrupción única: Se posee una única entrada común a los dispositivos, y se procede a preguntar a todos los dispositivos si solicitaron una interrupción (Polling/Encuesta).

Interrupciones Vectorizadas: Usan una línea de pedido común todos los dispositivos, cuando se realiza el pedido de interrupción se guarda un identificador (vector) del dispositivo que lo solicito, este vector será auto colocado por el periférico.

**DMA (directo a memoria)**

**Etapas de transferencia:**

**Inicialización de la transferencia** – Se establecen los parámetros de transferencia

Inicialización de la interfaz – La CPU establece a interfaz

N° byte, tipo transferencia (R/W) y datos control (pista, sector, etc.).

Inicialización de DMA – La CPU establece a DMA

N° bytes, tipo transferencia (R/W), Dir. Memoria y canal (dependiendo).

**En este momento la CPU ya se despreocupa de la transferencia.**

**Realización de la transferencia** – Cuando el periférico le informa al DMAC que está listo.

El DMAC pide control del bus e inicia transferencia

Después de cada palabra se actualizan n° bytes o palabras y dirección.

**Finalización de transferencia**

DMAC libera bus y devuelve control a CPU.

Suele informar a CPU mediante interrupción.

**Tipos de transferencia:**

**Por RAFAGAS** (Burst):  
 La DMA usa el bus para transferir todos los datos de una rápidamente sin tener en cuenta si la CPU necesita hacer uso del mismo para acceder a memoria. Por lo tanto se sufre un degradamiento del rendimiento del sistema

**Por ROBO DE CICLO** (Cycle-stealing):

La DMA hace uso del bus en los periodos de instrucciones en las que el CPU no va a necesitar entrar a memoria, ya sea porque posee los datos en cache o por que dicha instrucción no requiere de la misma, la DMA transfiere una palabra y vuelve a preguntar si puede seguir hasta terminar con todos los datos. Este método no degrada la velocidad el sistema, pero la velocidad de transferencia de datos E/S es más lenta. Por lo tanto este método es más eficiente para transferencias de múltiple palabras.

**SEGMENTACION DE INSTRUCCIÓNES**

**SEGMENTACION DEL CAUCE (PIPELINING)  
PROCESADOR ESCALAR SEGMENTADO**

El pipelining es una técnica de hardware invisible al nivel programador la cual logra aumentar la productividad (through-put) más no la velocidad de ejecución de una instrucción, consiste en la separación de las instrucciones en 5 fases para la posible ejecución en simultaneo de múltiples instrucciones. Para lograr esto se hace que cada instrucción realice una fase distinta al mismo tiempo, por lo que se logra estar ejecutando hasta 5 instrucciones simultáneamente.

**Fases del pipelining:**

**BUSQUEDA (F, fetch, MI)**

Se accede a memoria.

Se incremente PC.

**DECODIFICACION (D, decode, BR)**

Se decodifica la instrucción.

Se accede a los registros por el/los operandos (si es necesario).

Se calcula el valor del operando inmediato (si hace falta).

**EJECUCIÓN (E, execute, ALU)**

Se ejecuta la operación en ALU

**ACCESO A MEMORIA (M, memory Access, MD)**

Se accede a memoria (si es necesario).

**ALMACENAMIENTO (W, writeback, BR)**

Se guarda el resultado en un registro (si es requerido).

**Formato de instrucción:**

Es el formato en el que se pueden escribir las instrucciones.

Hay instrucciones tipo I (Inmediato), R (), J (Jump).

**Problemas:**

Dado que no todas las instrucciones necesitan todas las etapas, las etapas F y M requieren acceso a memoria, existen los saltos de control (los cuales cambian la siguiente instrucción), entre otras cosas esto puede generar errores en el programa.

**Atasco del cauce (stall)**

Se genera cuando se impide la ejecución de la instrucción en el ciclo que le corresponde

**Estructural:** Provocado por conflicto de recursos (Ej. 2 acceso a Mem.).

**Por dependencia de datos:** Cuando 2 instrucciones utilizan el mismo dato (WAR).

**Por dependencia de control:** Cuando la ejecución de una instrucción depende de cómo se ejecute otra (Cuando se realiza un salto, con 2 posibles caminos).

**Stall por dependencia de datos:**

**RAW (lectura después de escritura) (Dependencia Verdadera):** Cuando una instrucción intenta leer un dato que la instrucción anterior tiene que modificar

**WAW (escritura después de escritura) (Dependencia de Salida):** Una instrucción escribe un dato después que una posterior (Ej. P.F.).

**WAR (escritura después de lectura) (Anti-dependencia):** Una instrucción modifica un valor antes que una posterior lo lea.

**Soluciones:**

**Riesgos estructurales: (simple)  
 -**Duplicación de recursos de Hardware (Se agrega un Sumador y restador a la ALU).  
 -Separación de memoria de instrucciones y datos.  
 -Turnar acceso a Registros (Escritura en primera mitad de ciclo, lectura 2nda).

**Riesgos RAW:** Mediante una unidad de riesgos o un compilador se detecta si:  
 Es por HARDWARE:  
 -Mediante fowarding.[[1]](#footnote-1)

Es por SOFTWARE:  
 -Instrucciones NOP.  
 -Reordenación del código.

**RIESGOS DE CONTROL:** Se separa en 2 tipos de riesgos

**Incondicionales:** La dirección destino debe ser calculada pero es fija, un solo resultado es posible.

**Condicionales:** La dirección destino va a variar según el resultado de la instrucción previa. Por lo que le programa puede proseguir en 2 caminos.

**Soluciones a Riesgos de Control:**

Se separan en 2 grupos de soluciones, soluciones por HARDWARE (predicción de saltos para evitar la parada) o por SOFTWARE (salto retardado o con relleno de ranura de retardo (nop)).

**HARDWARE:**

**Detención del cauce:** Se basa en que después de un salto siempre se retrasa un ciclo el inicio de la próxima instrucción.

**Métodos de predicción de salto:**

**Mejora de ruta de datos:** Se adelanta la llegada del resultado, sacado el resultado directamente de la salida de la ALU, de esta manera se consigue un atraso de 1 ciclo.

**Técnica estática:**

Consiste en la “predicción” de que nunca va asaltar, o siempre va a saltar. Es decir  
 que siempre va empezar a hacer la siguiente operación del caso que sea.

**Técnica dinámica:**

**Conmutador:** Guarda el resultado de la última vez que se ejecutó el salto, bueno para bucles.

**Tabla de historia de saltos** (BTB, branch-target buffer):  
 Posee una pequeña memoria muy rápida donde se guardan direcciones de la instrucción de bifurcación tomada, instrucción o dirección destino e historia de uso.

**Predicción según código de operación:** Hay instrucciones que son más probables a saltar que otras, este sistema llega a un 75% de acierto.

**Flujos múltiples:** Se pre-capta todos los cauces y sus respectivos saltos, posteriormente se elige el correcto. Posee la desventaja que provoca retardos en el acceso al bus y registros.

**Pre-captar el destino de salto:** Se pre-capta la instrucción destino al salto así como también las siguientes instrucciones a la bifurcación, hasta que se ejecute la instrucción bifurcación.

**Buffer de bucles:** El buffer trabaja en la etapa de captación de instrucción del cauce. Consiste en que comprueba el buffer antes de hacer la captación de memoria. Muy eficaz ara pequeños bucles y saltos.

**SOFTWARE:**

**Salto retardado:** (delay slot)Se basa en adelantar una instrucción (que no repercuta al salto) para ir realizando alguna operación en el tiempo que se espera el resultado para el salto, en caso de que no se pueda adelantar una instrucción se coloca un NOP.

**SEGMENTACION DE MEMORIA**

Siempre se busca la máxima capacidad de memoria más rápida posible.

Es decir que sería ideal si se pudiese tener una memoria infinita con la velocidad de los registros, pero si tenemos muchos registros deberíamos utilizar más bytes para las direcciones de los registros, lo cual ralentizaría la velocidad de lectura hacia los mismos.

Por lo tanto lo que se hizo fue crear la jerarquía de memoria con diferentes tipos de memoria que se van balanceando entre capacidad y velocidad. En estas se van copiando a sus hijos pequeñas partes de memoria según se vaya necesitando.

**JERARQUIA DE MEMORIA**

Registros: Se encuentran en el CPU con un tamaño de byte / KB, velocidad 0,25 ns.

Cache: Se encuentran en la CPU o chip pero con capacidad KB/MB y velocidad 1 ns.

Memoria: Conocido como RAM con una capacidad (MB/GB), velocidad 100ns.

Memoria Virtual: La virtualización del HDD/SDD (GB/TB) pero con una velocidad 5 ms.

Para saber que copiar a las memorias más rápidas se utilizan los **principios de localidad de referencia** que cumplen todos los **programas**.  
 **Principio de localidad temporal:** Dice que los elementos (datos o instrucciones) a los que se accedió recientemente son muy probables de volver a ser accedidos en un futuro próximo.

**Principio de localidad Espacial:** Dice que los elementos recientemente ejecutados o próximos a un elemento ejecutado son muy probables de ser ejecutados en la brevedad.

Con estos principios se copia, con un 90% de efectividad, la próxima instrucción a los sub-hijos; Y el 10% restante de las veces se tiene que acceder a la memoria padre (más lenta) en busca de la instrucción requerida.

El CPU y Cache se manejan mediante el bus de datos usualmente 64 bits (Windows 64 o 32 etc.), decimos que se pasan **palabras**. Mientras que entre la memoria y la cache se pasan **bloques**, ósea mas cantidad de datos dependiendo los principios de localidad. Cuanto más chica y más cerca del procesador se encuentre la unidad de almacenamiento, datos más chicos se les pasa.   
Siempre se les pasa a sus hijos el dato y la dirección de memoria (etiqueta) donde está almacenado el dato. Así mismo para saber de que forma copiar los datos de memoria a cache se utilizan ciertos métodos de correspondencia.

**Métodos de correspondencia**

**Correspondencia totalmente asociativa:** El controlador de cache le asignara el lugar que encuentre disponible, es decir que no hay una regla que indique donde se va a guardar. No posee índice.  
 El problema es que para encontrar un bloque se deben recorrer todo la cache.

**Correspondencia Directa:** Cada bloque posee un único lugar en cache, siempre el mismo. Dicha ubicación está determinada por N° Dir. Bloque mod N° líneas cache = Dir. de cache para ese bloque. Si coincide con una dirección previamente escrita, se tapa al anterior.  
 La desventaja es que para programas de repetición se estaría constantemente sobrescribiendo los bloques, resultando ineficiente.

**Correspondencia asociativa por conjunto:** (Asociativa 2/4 vías)Es un punto medio entre las 2, esto se hace teniendo **conjuntos** (que son un grupo de líneas de cache, por lo general cada conjunto son entre 2 y 4 bloques) y para saber a qué conjunto pertenece el bloque se hace N° Dir. Bloque mod N° conjuntos en cache = N° conjunto de dicho bloque.  
 De esta forma solo se deberá buscar entre los 2/4 conjuntos que poseen el el n° de bloque buscado y al poseer varias líneas no se estaría sobre escribiendo tanto.

**Tipo Correspondencia**

El tipo de correspondencia es como están guardados las direcciones de memoria de los datos.  
Con la **ETIQUETA** más el **INDICE** se consigue el bloque.   
El **BO** es la dirección precisa de cada elemento del bloque.  
La suma de todos nos da la dirección en memoria del dato.

Para este Ej. Usamos bloques de 4 datos y 4 lineas, Por lo tanto una capacidad máxima de 16 datos.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Totalmente asociativa** | | | | |  |
| **Etiqueta** | **00** | **01** | **10** | **11** | Se usan **6 bits** para la dirección. |
| **101011** | 1A | FF | 32 | 4C | Dir. **10101100** = 1A |
|  |  |  |  |  | Dir. **10101101** = FF |
|  |  |  |  |  | Dir. **10101110** = 32 |
| **011000** | D5 | 00 | A3 | 02 | Dir. **10101111** = 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Directa** | | | | |  |
|  | **Etiqueta** | **00** | **01** | **10** | **11** | Se usan **4 bits** para la dirección. |
| **00** | **0010** | 1A | FF | 32 | 4C | Dir. **00100000** = 1A |
| **01** |  |  |  |  |  | Dir. **00100001** = 1A |
| **10** |  |  |  |  |  | Dir. **00100010** = 1A |
| **11** | **1100** | D5 | 00 | A3 | 02 | Dir. **00100011** = 1A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Asociativa por conjunto** | | | | |  |
|  | **Etiqueta** | **00** | **01** | **10** | **11** | Se usan **5 bits** para la dirección y tiene **2 conjuntos** por índice. |
| **0** | **00101** | 1A | FF | 32 | 4C | Dir. **00101000** = 1A Dir. **11000000** = D5 |
| **1** |  |  |  |  |  | Dir. **00101001** = FF Dir. **11000001** = 00 |
|  |  |  |  |  |  | Dir. **00101010** = 32 Dir. **11000010** = A3 |
| **0** | **11000** | D5 | 00 | A3 | 02 | Dir. **00101011** = 4C Dir. **11000011** = 02 |
| **1** |  |  |  |  |  |  |

**Políticas de remplazo**

Para remplazar los bloques y decidir por cual se crearon los algoritmos de sustitución.  
Dichos algoritmos están ejecutados por el **controlador de memoria cache**

**Correspondencia directa:** A cada bloque le corresponde un único lugar por lo que simplemente lo remplaza.

**Correspondencia asociativa:**

**LRU:** (Last Recently Used)Se lleva un conteo de cuanto se va usando cada bloque, el últimamente menos usado es el remplazado.

**FIFO:** (First In First Out)  
 El primero que entro es el primero que se va.

**LFU:** (Least Frecuently Used)  
 El que menos se haya usado es el remplazado.

**Aleatorio:** Se sustituye una línea al azar.

**Política de escritura**

**ESCRITURA EN ACIERTO:**   
Se denomina así cuando el dato a guardar se encuentra en la cache.

**WRITE-THROUGH:** Se actualiza simultáneamente la cache y la Mem. Principal.

**WRITE-BACK:** Solo se actualiza la cache, pero deja marcado que se realizó un cambio. Cuando se va a sobrescribir ahí se procede a copiar el contenido a la Mem. Principal.

**ESCRITURA EN FALLO:**Cuando el dato a guardar no se encuentra en la cache.

**WRITE ALLOCATE:** EL dato se lleva de la principal a la cache y se sobre escribe la misma.  
 Comúnmente se usa con WRITE-BACK.

**NO-WRITE ALLOCATE:** El bloque no se escribe en la cache, solo se escribe en la Mem. Principal.   
 Se suele usar con WRITE-THROUGH.

**TECNOLOGIAS DE PROCESADORES**

**Distintos tipos de PROCESADORES:**

**-Procesador Escalar:** Ejecución secuencial.

-**Procesador Escalar Segmentado:** Con segmentación del cauce (pipeline).

**-Procesador Súper-Escalares:** Se le denomina procesadores súper-escalares a los procesadores que tienen duplicadas algunas unidades funcionales de la CPU/ALU, de esta forma se logra llevar a cabo más de un instrucción simultáneamente.

**-Procesador Súper-Escalares Segmentados:** (se usan en la actualidad)Son Súper-Escalares pero con segmentación del cauce. Es decir vamos a tener una pipeline con 2 procesos de búsqueda simultáneos, 2 decodificación, 2 Ex, 2 M y 2 WB. Tiene un gran grado de paralelismo.

**-Procesador Súper-Segmentado:** (se usan en la actualidad)Son procesadores en los cuales muchas etapas van a tardar menos de un ciclo en la ejecución, por lo que en un ciclo se realizan por Ej. 2 decodificaciones. Es decir permite aumentar la frecuencia del ciclo del reloj. No tienen duplicadas las unidades funcionales. Tiene un gran grado de paralelismo.

Estos nuevos procesadores (Súper-Escalares y Súper-Segmentados) requieren un nueva forma de ordenar las instrucciones, requieren de un Paquete de Emisión del cual sus instrucciones deben poder ser ejecutadas simultaneamente.

Esto lograra una mayor velocidad pero generaran más riesgos estructurales, para esto se crearon métodos de Emisión de Instrucciones.

**Métodos de emisión de instrucciones:**

**Estáticas**:   
 Este grupo de instrucciones esta generado por el compilador. Este genera el paquete de emisión evitando riesgos de datos y control.

**Dinámicas:** En este caso el paquete es creado por el CPU.   
 Este modelo es usado por las Súper-Escalares.

**Estados de instrucción:  
 Distribuida:** Instrucción enviada a estación de reserva.

**Emitida:** Sale de estación de reserva haciaunidad funcional.

**Finalizada:** Sale de estación de reserva y pasa a buffer de reordenamiento con el resto de registros, donde todavía no es visible por el programador.

**Terminada:** Cuando ya ha sido escrito en un registro con los renombramientos necesarios, ya es visible al programador.

**Retirada:** Cuando, en caso de ser necesario, el dato ya es escrito en memoria.

En los procesadores segmentados la etapa ID (Instrucción Decodification) se particiona en 3 partes Pre-Decodificación, Decodificación (Dispatcher / ventana de instrucciones) y Emisión (Issue). Una vez que la instrucción llega a la ventana de instrucciones, esta se encarga de esperar a la disponibilidad de los operandos y unidades funcionales requeridas, si ambas se cumple la ventana realiza la emisión.

Puede ser el caso que muchas instrucciones cumplan lo requerido para ser emitidas, pero que no todas puedan ser emitidas en simultáneo, por lo que se creó la política de emisión.

**Políticas de emisión:**

**Emisión alineada:** NO pueden ingresar instrucciones a la ventana de instrucción hasta que la ventana haya emitido todas las instrucciones que tenia previamente.

**Emisión NO alineada:** Mientras exista un espacio en la ventana, pueden ingresar instrucciones.

**Emisión ordenada:** Las instrucciones se ejecutan ESTRICTAMENTE en el orden que entran a la ventana de instrucciones, si una se para, las siguientes también.

**Emisión desordenada** No existen bloqueos, las instrucciones se ejecutan cuando cumplan las condiciones requeridas (UF y operandos requeridos).

**TECNICAS DE OPTIMIZACIÓN**

Hay 3 técnicas fundamentales de optimización duplicación de recursos, política de ejecución y renombramiento de registros.

Duplicación de recursos:  
 Es la forma en la que funciona los procesadores Super-Escalares, el cual funciona con la duplicación de recursos y el dispatcher como cerebro.

Políticas de ejecución  
 Politica de emisión.  
 Politica de finalización.

Renombramiento de registros.

**Riesgos del Paralelismo:** DEPENDENCIA DE DATOS VERDADERA: (RAW)  
 DEPENDENCIA RELATIVA AL PROCEDIMIENTO:  
 CONFLICTOS DE RECURSOS:  
 DEPENDENCIA DE SALIDA: (WAW)  
 ANTIDEPENDENCIA: (WAR)  
Estos 2 últimos se pueden solucionar con el método de Renombres de registros, ya que son creados por conflictos de almacenamiento.

El Renombre de registros consiste en que cuando una instrucción modifica un registro, esta guarda el valor en un registro nuevo y actualiza la dirección de referencia hacia el nuevo registro, es decir se van re asignando dinámicamente los registros.

De esta forma solo queda riesgos de atascos RAW.

**Implementación Super-Escalar:**

-Estrategia de captación simultanea de múltiples instrucciones.  
 -Lógica para determinar dependencia verdadera entre registros.  
 -Mecanismos para iniciar múltiples instrucciones en paralelo.  
 -Recursos para la ejecución de múltiples instrucciones en paralelo.  
 -Mecanismos para entregar el estado del procesador en orden correcto.

En caso de excepciones imprecisas (errores), se terminan de ejecutar las instrucciones en proceso y se corta la fila de instrucciones. Una vez terminada las instrucciones en ejecución, se ejecuta la instrucción que dio error y posteriormente se reanuda el cauce.

**Los distintos buses**

Cuando más abajo más lento. El más rápido es el bus local, después el de sistema y por último el de expansión.

**Bus Local:** Es el que conecta la CPU con la cache y los controladores locales de E/S.

**Bus de Sistema:** Conecta la cache con la memoria y la interfaz del bus de expansión.

**Bus de Expansión:** Conecta la interfaz del bus de expansión con las entradas más lentas como la de red, la del modem, teclado, mouse, etc.

**Tipos de sistemas**

**Sistemas Sincrónicos:** Todos los dispositivos trabajan al compás del procesador, en caso de que haya uno más lento este tendrá que trabajar cuando pueda.

**Sistemas Asincrónicos:** El sistema va trabajando a una velocidad variable ya que se va esperando a la respuesta de cada dispositivo. Por lo tanto si tenemos un dispositivo muy lentos este va a ralentizar a todos.

Inclusión: Los datos almacenados en un nivel han de estar almacenados en los niveles inferiores a él.

Coherencia: Las copias de la misma información almacenada en los distintos niveles deben tener los mismos valores.

**MSX88 Vonsim (CISC)**

Sistema Von-Neumann:  
 Memoria compartida

**WIN-MIPS 64 (RISC)**

Sistema Harvard:

Se le llama reducido por que posee menos modos de direccionamiento que el CISC (load / store)

Memoria de datos y registros separados (32 registros uso general, 32 de P.F.)

Segmentación de instrucciones.

Instrucciones de longitud fija (32 bits).

Formatos de instrucciones R, I, Control / J.

1. Fowarding: Es una técnica simple que consiste en el adelantamiento del dato a la próxima instrucción, consiste en pasarle el dato a la próxima instrucción apenas se obtiene el resultado (En la fase … ) [↑](#footnote-ref-1)