Resumen para final de Arquitectura de Computadoras

ALU - (UNIDAD ARITMÉTICO LÓGICO)

Unidad aritmética lógica encargada del procesamiento lógico y aritmético en la ejecución de procesos en tiempo de ejecución. De ella nace el concepto de Aritmética de un computador el cual entiende dos tipos de números: enteros y de coma flotante.

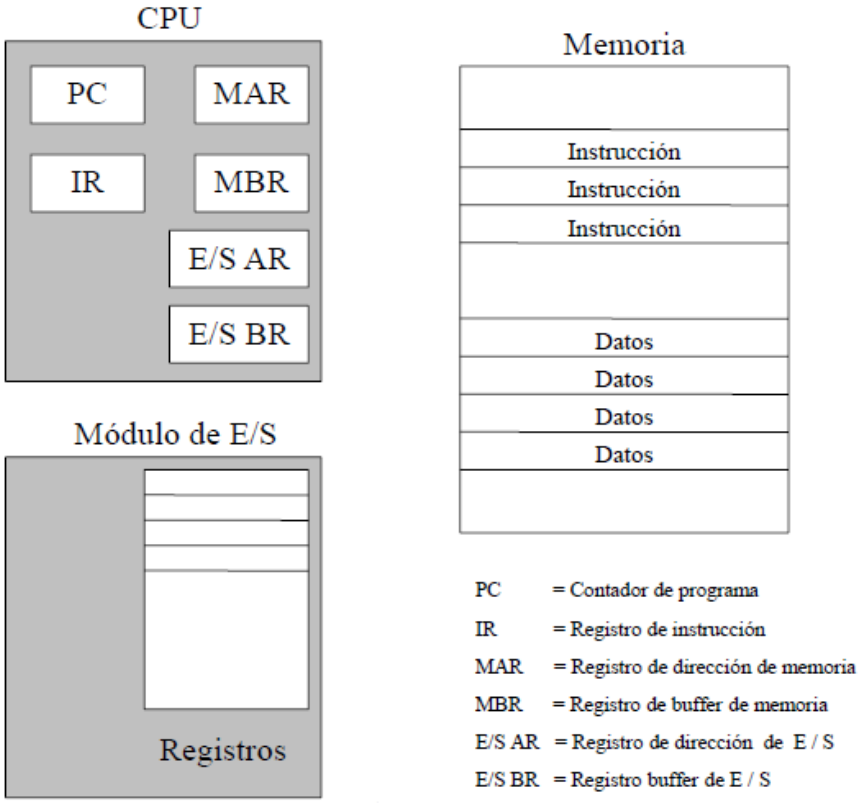
- Elementos de un computador suministran datos a la ALU (RAM, UC, REGISTROS, E/S)
- Núcleo de una computadora
- Activador de flags

ARQUITECTURA DE VON NEUMANN:

CPU: Compuesto por una unidad de control (UC) y una unidad aritmética lógica (ALU)

El flujo trataría de que deben ingresar datos e instrucciones que serían tareas a procesar para esta unidad para tener como fin una respuesta proporcionada mediante componentes de E/S

Para poder llegar a tal fin de procesamiento surgió la necesidad de poder almacenar estos datos e instrucciones a procesar de manera temporal a lo que es MEMORIA PRINCIPAL



PC = Contador de programa
IR = Registro de instrucción
MAR = Registro de dirección de memoria
MBR = Registro de buffer de memoria
E/S AR = Registro de dirección de E / S
E/S BR = Registro buffer de E / S

REPERTORIO O CONJUNTO DE INSTRUCCIONES:

Se nombra así al conjunto completo de instrucciones que se realizan en una CPU, existe el tipo que es CÓDIGO MÁQUINA o BINARIO el cual es representado simbólicamente por un conjunto de códigos de ensamblaje (ADD- SUB-MOV).

Todo lenguaje de alto nivel (lenguajes de programación) pasa un proceso de compilación y ensamblado para así llegar a este conjunto de instrucciones el cual la máquina podrá entender y ejecutar.

TIPOS DE INSTRUCCIONES:

Existen distintos tipos de instrucciones: Están los que serán procesados por la CPU, estas son las instrucciones aritméticas lógicas. Están las instrucciones que conllevarán al almacenamiento de datos, estas son las instrucciones de almacenamiento. Las de transferencia de datos, y que en algunos casos se realizan, instrucciones de testeo y flujo de programas.

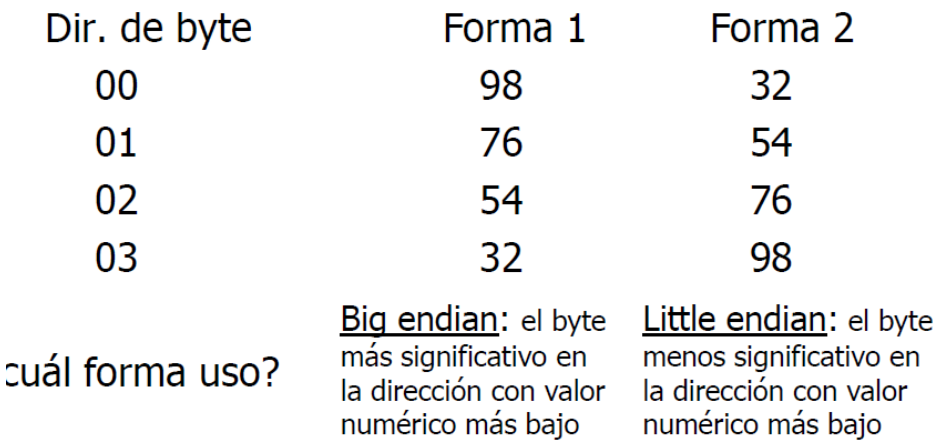
TIPOS DE OPERANDOS:

Los tipos de operandos que existen son: direcciones, números (punto fijo y punto flotante), caracteres ASCII y datos lógicos (0 y 1).

ORDEN DE ALMACENAMIENTO EN MEMORIA:

Existen dos modos de almacenamiento en memoria, Big endian y Little endian:

- BIG ENDIAN: el almacenamiento comienza desde el byte más significativo del número a almacenar en la dirección con valor número más bajo
- LITTLE ENDIAN: el almacenamiento comienza desde el byte menos significativo del número a almacenar en la dirección con valor número más bajo, es decir, al revés



TIPOS DE OPERACIONES:

Operaciones que se pueden realizar:

- Transferencia de datos
- Operaciones aritméticas
- Operaciones Lógicas

- Operaciones de conversión de datos
- Operaciones de entrada/salida
- Control de sistema
- Control de flujo
- Transferencia de datos

Para poder realizar esta operación es necesario tener en cuenta la dirección origen, la dirección destino, el tamaño de lo que se va a transferir y el modo de direccionamiento en el que se realizara. La transferencia puede ser de registro a registro, de registro a memoria, de memoria a registro.

Cuando resulta de registro a registro, la transferencia es más sencilla, ya que se reducen el paso de cosas que tiene que realizar el procesador y termina siendo una transferencia de reg. a reg. realizándose de manera interna al procesador, pero cuando es de reg. a memoria, el procesador tiene que hacer más pasos para realizarlo:

Calcular la dirección de memoria basándose en el modo de direccionamiento que se definió

Si la dirección hace referencia a memoria virtual, habrá que realizar la traducción para poder obtener la dirección real de la posición de memoria a donde se quiere transferir o a donde se quiere llegar

Validar también si lo que se quiere transferir está en caché, y si no es así cursar el módulo de memoria (RAM)

- Aritméticas

Involucra a las operaciones ADD- SUB-DIV-MUL-INC-DEC-NEG(CA2) y ABS

Muchas veces para realizar estas operaciones el procesador requiere hacer operaciones de transferencias para buscar en memoria los operandos o para saber dónde almacenar el resultado después de haber realizado la ALU la operación.

- Lógicas

Operaciones como AND-OR-XOR-NOT

- Conversión de datos

Conversión de datos en formato ASCII a EBCDIC (anda a saber qué formato es ese)

- Entrada/salida

Como indica su nombre, operación de entrada y salida de datos IN-OUT los cuales pueden ser realizados mediante la instrucción MOV para mover datos o cuando se hace uso del DMA

- Control de flujo

Como ese caso de en ejecución de instrucciones realizar un salto a otra instrucción con el comando JMP o después de una condición con el comando JZ o cuando se llama a una subrutina con el comando CALL para después retornar de ella como RET

Resultan importantes contar con este tipo de instrucciones por cuestiones de modularizar en la realización de tareas, permitiendo cubrir los casos donde hay que realizar más de una vez una misma tarea o n – veces.

- Control de sistemas

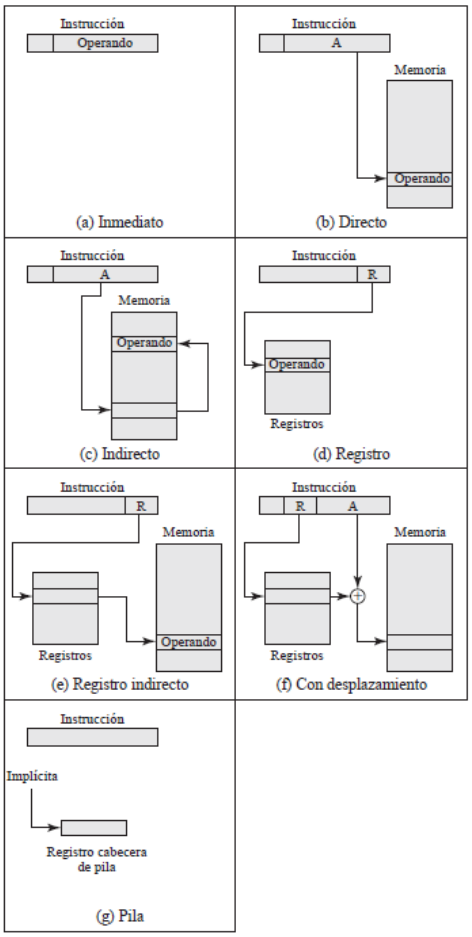
Estas instrucciones para el control de sistemas suelen ocurrir cuando el procesador se encuentra en un estado de privilegio o cuando un programa tiene acceso a una zona específica de memoria privilegiada y normalmente estas instrucciones suelen estar reservadas para el uso exclusivo del SO.

Estas instrucciones están relacionadas con la lectura y alteración de registros de control o claves de protección.

MODOS DE DIRECCIONAMIENTO:

Todos los modos de direccionamiento:

- Direccionamiento Inmediato
- Direccionamiento Directo
- Direccionamiento Indirecto
- Direccionamiento de Registros
- Direccionamiento Indirecto con Registros
- Direccionamiento con Desplazamiento
- Direccionamiento con Pila



- Direccionamiento Inmediato:

Direccionamiento más simple de todos en donde en realidad el o los operandos que se necesitan para una operación están ya presenten en las mismas instrucciones, por lo tanto, ya no hay que ir a buscarlos.

Desventaja de esto es que el elemento ubicado en la instrucción debe estar contenido en la capacidad que tiene el campo que almacena la instrucción, es decir, que no puede pesar más de lo permitido por instrucción.

- Direccionamiento Directo:

En las instrucciones se especifica la dirección de memoria en donde habrá que ir a buscar el operando en cuestión. Por lo tanto, surge por lo menos un ciclo de memoria por la búsqueda.
Un problema que surgía con esto es que la capacidad que tenían los campos de direcciones a veces no era suficiente para poder almacenar la dirección completa de un operador.

- Direccionamiento Indirecto:

A modo de solución, se almacenaba en el campo de direcciones una dirección que direccionara a una posición de memoria donde estaría almacenada la dirección completa del operador que se estaba buscando, de ahí se dice que se realiza un direccionamiento indirecto por la doble redirección como mínimo que se tenía que realizar.

- Direccionamiento a Registros:

Similar a lo que es el direccionamiento directo, pero en vez de direccionar a una posición de memoria principal, se direcciona a un registro que es donde estaría almacenado el operando en cuestión.
Las ventajas son más que las desventajas. Ventajas como que el espacio ocupado que se necesita para poder direccionar a un registro es menor que a un espacio de memoria principal, además de que el direccionamiento resulta más rápido en performance, ya que no hay que acceder en ningún momento a memoria y los registros se encuentran ubicados internamente en los procesadores, pero sigue generando el problema sobre la cantidad de direcciones que se puede almacenar en el campo de direcciones de una instrucción, sigue siendo limitado. En un uso masivo de direcciones a registros, generaría problemas el almacenamiento de estos en un mismo campo.

- Direccionamiento Indirecto con Registros:

Similar a lo que es el direccionamiento indirecto, con las mismas limitaciones y ventajas con la única diferencia de que en este caso surge un acceso menos a memoria para obtener el que almacena la dirección de donde se ubica la dirección completa, ya que lo tiene almacenado un registro.

- Direccionamiento con Desplazamiento:

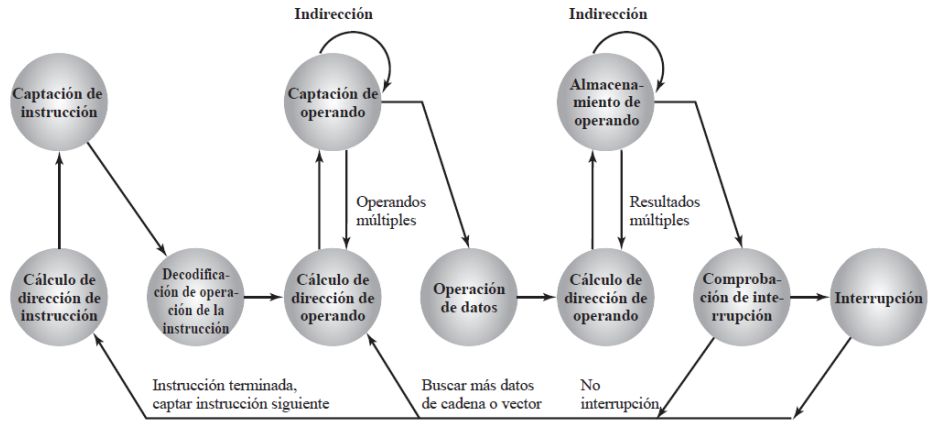
Resulta en una combinación entre el direccionamiento directo con el direccionamiento con registros. Se necesitan dos campos de direcciones donde, por un lado, tener la referencia al registro que almacenara el contenido que, al sumarlo con dirección obtenida del direccionamiento directo, se pueda obtener la dirección completa del operador que se está buscando

- Direccionamiento en Pila:

Suele tener un uso implícito, muchas veces se usa la pila para tener referencias a registros que almacenan operandos que en cuestión son los que el procesador está buscando para sus operaciones, y como la pila es un espacio de memoria reservado, las direcciones terminan siendo direccionamientos indirectos a registros

CICLO DE LAS INSTRUCCIONES:

Consta de los siguientes sub-ciclos, uno es la captación, la ejecución y la interrupción.



- Ciclo de Captación:

La dirección de la instrucción que se debe captar se encuentra ubicada en el registro PC, una vez ubicada se dirige al registro IR de instrucción donde para empezar con la ejecución antes se incrementa el registro PC para poder ubicar la instrucción siguiente a no sé qué se indique lo contrario (interrupción)

- Ciclo de ejecución:

Proceso de ejecución de instrucciones previamente captadas.

Tipos:

Entre procesador y memoria:

Usualmente, se realizan transferencia de datos

Entre procesador y módulos de E/S:

Para transferencias desde CPU hasta módulos de E/S

Solo procesador:

Procesamiento aritmético lógico de datos

De control:

Dado usualmente para alterar o modificar secuencias de ejecución

Híbridas o por combinación:

Donde se puede hacer un conjunto de ejecuciones que suele ser más común entre conjuntos de instrucciones

La ejecución de una instrucción solo se puede detener si ocurre algo con la máquina, si ocurre algún error que sea irrecuperable y perjudique al ciclo de ejecución o se ejecute alguna instrucción que detenga al computador.
Esto pasaría a tratarse en la gestión de interrupciones que surgen cuando se están en el ciclo de ejecución, así cuando ocurra algo inesperado, daría comienzo a un clico de interrupciones para poder llegar a tratarlo y continuar con el ciclo de ejecución

Subrutinas

Programas autocontenidos que brindan reusabilidad y modularización, donde se les pueden pasar parámetros para cumplir una cierta tarea, ya sea por valor (copia de datos) o por referencia (direcciones reales a memoria de datos). En código de ensamblaje pueden ser llamados con la instrucción CALL.

Con respecto al pasaje de parámetros:

Cuando es con pasaje vía registros surgen la precaria cantidad de registros que se pueden usar, ya que no se cuentan con muchas y además de que hay que documentar lo que se almacenan en ellas debido a su interpretación
Con respecto al pasaje vía memoria, se cuentan con más capacidad para el pasaje y utilización, también con espacios de almacenamiento de memoria para declarar los parámetros, pero resulta ser un problema su estandarización debido a qué es de uso compartido entre los diferentes procesos y aplicaciones, entre otros aspectos.
Y por último y él más uso es por vía pila, se maneja un registro SP (Stack Pointer) el cual resulta ser independiente de memoria y registros, la pila es un espacio reservado de memoria con una estructura First Come First Served (FCFS) donde el primer elemento en apilarse en la pila, será el apuntado por el SP y, por ende, el primero en desapilar cuando alguna instrucción lo requiera utilizando las instrucciones PUSH para apilar y POP para desapilar como comandos de lenguaje de ensamblado.

Interrupciones:

Las interrupciones ocurren de manera interna o externa a la CPU y muchas veces interrumpen el flujo de ejecución de ella. Y en muchos casos surgen ya sea por:

Interrupción por desbordamiento aritmético (overflow) o división por cero, esto sería de manera interna. Por algo programado, es decir, por un temporizador interno del procesador, por operaciones de E/S que hay que cubrir o por atención de errores de hardware que ocurren de manera inesperada.
Cuando las interrupciones están habilitadas para ser atendidas en tiempo de ejecución, se hace un chequeo de si hay interrupciones por atender, si no lo hay se sigue con el flujo normal. Recién hasta cuando se detecta una interrupción, y en caso de ser no enmascarable empieza el ciclo de interrupción donde primero hay que guardar el contexto, modificar el PC con la dirección que apunta una subrutina encargada del manejo de interrupciones, según sea el tipo de interrupción, será necesario atender por completo o no para retornar al ciclo de ejecución de instrucciones.

De todas estas posibles interrupciones, se delega en el gestor de interrupciones la tarea de poder intervenir y poder llevar a cabo el flujo sin que se perjudique el entorno de ejecución que se estaba llevando a cabo, y así, una vez realizada la interrupción poder seguir el flujo normal. Los pasos que realiza este gestor se basan en salvar el estado del procesador una vez ocurrida la interrupción, tratar la interrupción y una vez finalizado encargarse de volver al contexto anterior restaurando el estado guardado (el que fue apilado) y así el procesador continuará con la ejecución normal del programa.
Las interrupciones se subdividen en una jerarquía donde existen unas más importantes que otras.

- No Enmascarables:

  Eventos de alta prioridad que requieren atención inmediata
- Enmascarables:

  Eventos con más accesibilidad al trato con instrucciones

Interrupciones por hardware:
Dentro de las no enmascarables están las interrupciones por hardware las cuales se consideran las más prioritarias e importantes de tratar cuando surgen. Suelen ser eventos no planeados o asincrónicos. Conocidas como INTERRUPT REQUEST.
Traps/Excepciones:
Son interrupciones por hardware que a priori el procesador los definió ante posibles casos donde haya que realizar una interrupción en tiempo de ejecución: Ejemplos básicos, operaciones algorítmicas o lógicas que generen errores de cálculo. Fallas en la ejecución de programas o como otro caso, acceso a memoria de lugares protegidos sin acceso
Interrupciones por software:
Las enmascarables son las interrupciones por software que en la mayoría de los casos tienen una manera de poder tratarlos y poder continuar con el procesamiento normal sin necesidad de tener que detener la ejecución del programa.
Interrupciones Múltiples:
Con las interrupciones múltiples hablan del caso en donde se podría llegar a recibir varias interrupciones en un mismo ciclo de ejecución que requieran ser atendidas con un cierto criterio.
Una forma de manejar esta situación es atender a la primera interrupción recibida e inmediatamente deshabilitar las interrupciones hasta finalizar con el ciclo de interrupción de la misma en cuestión. Por supuesto las interrupciones recibidas se siguen almacenando, pero no se les brindan atención y se mantienen en estado pendiente hasta que el gestor finalice. Es claro que esto resulta ineficiente por la desatención a interrupciones que podrían resultar críticas de corta duración de atención, ya que se podrían perder los datos, así como también ausencia de prioridades entre las interrupciones.

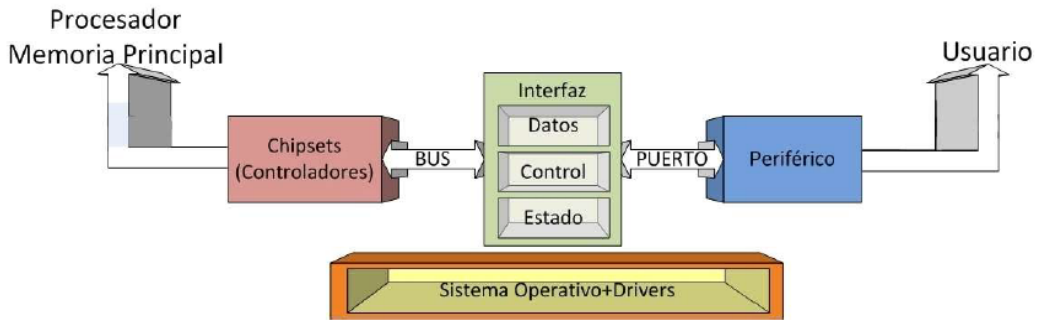
Otra forma de atender interrupciones múltiples es la manera de otorgar una prioridad a priori para que cuando se recibe una interrupción con prioridad X e inmediatamente se recibe una con mayor atención, automáticamente se apila el contexto del ciclo en cuestión de la interrupción que se está tratando y se da prioridad de atención a la interrupción con mayor prioridad. En relación con los módulos de E/S surgen distintas formas de tratar las interrupciones, están las INTERRUPCIONES MULTINIVEL, caso donde por cada módulo de E/S hay una conexión directa con el procesador, independientes entre sí, para poder emitir interrupciones cuando se les plazca, esto resulta sencillo, pero caro a la vez por las líneas de comunicación que estarían ocupadas solo para algo tan sencillo que es emitir señales.

Como solución a este caso multinivel, surgen la manera de mantener conectados entre sí a los módulos de E/S, así, cuando alguno de ellos emita una interrupción lo haga desde una misma línea de comunicación y después el procesador se encargue de verificar, es decir, chequear el registro de cada módulo averiguando quién fue el que lo emitió y así poder responder con el gestor correspondiente a la interrupción emitida, esto se llamó

INTERRUPCIONES DE LÍNEA ÚNICA. Obviamente, esto resultó ineficiente por el tiempo que debe tomarse el procesador para verificar, así que se pensó en otro método que se basó en mejorarlo. Ahora cuando un módulo emita una interrupción, este enviaría un vector que funciona como identificador, para que el procesador o controlador de interrupciones (PIC) sepa cómo gestionarlo sin necesidad de preguntarle a cada módulo, es decir, que con este vector sabría como atender adecuadamente para tal interrupción buscando la rutina correspondiente en el vector de interrupciones e identificaría el periférico qué lo emitió. Esto se denominó INTERRUPCIONES VECTORIZADAS.

Entrada/Salida

Al surgir una gran variedad de dispositivos externos con distintas propiedades, algunos más rápidos para la transferencia de datos y otros no comparándolos con el de un mismo procesador en cuestión, entre otras diferentes capacidades surgen los módulos de entrada/salida. Estos son la interfaz entre el procesador y memoria principal con los distintos periféricos que puede haber conectados a un computador.



Estos módulos se definen de la siguiente manera mediante el cual cumplen su rol para poder mantener un control y timing.

- Control y temporización
- Interpretación de señales, es decir, lograr la comunicación entre los distintos autores
- Control en la manera en que se realizan las transferencias, ya sea tanto en velocidad como en formato, mantener una compatibilidad
- Detectar errores y poder hacer algo al respecto
- Dar reportes de estado entre los distintos autores

Así como los módulos conecta al procesador con varios periféricos, existe otra interfaz que es la que conecta al periférico con el módulo: este es el llamado PUERTO
A través de este se realizan envíos de 3 señales como las más destacadas:
Señales de control: donde se deja en claro la función u operación que se va a llevar a cabo (LECTURA o ENTRADA, ESCRITURA o SALIDA)
Señales de estado: mediante el cual se notifica el estado de cada uno de los autores (READY-NOT READY)
Señales de datos: Como lo dice su nombre, a través de estas señales se transportarán los datos que sean necesarios enviar como un conjunto de bits los cuales podrán ser enviados en ambas direcciones según sea el caso de la operación a realizar

Otras propiedades de un módulo:

- Transductor: encargado de la recepción y conversión de datos con el fin de lograr la interpretación correcta de los datos para ambos autores
- Control lógico: Encargado del manejo de direcciones
- Buffer: brinda adaptabilidad y almacenamiento temporal

Para poder realizar las operaciones tienen que realizar distintas tareas para poder lograrlo:
Direccionamiento: E/S mapeada a memoria y E/S aislada

- E/S mapeada a memoria se basa en compartir los mismos espacios de direcciones entre los dispositivos y la memoria donde no haya diferencias ni de estos espacios ni de los comandos a utilizar para la E/S
- E/S aislada consta de discriminar los espacios de direcciones tanto para los dispositivos E/S y la memoria de manera que las órdenes para la E/S se vuelven especificas lo que hace que resulte limitante.

DISTINTAS TÉCNICAS DE GESTIÓN DE E/S

- E/S Programada con espera de respuesta

La CPU mantiene el control total de la operación de E/S que hay que realizar. Ordena al módulo y dispositivo que debe ejecutarse y que hacer a través de una orden. Periódicamente, la CPU pregunta por los bits de estado del módulo que indican si la E/S finalizó, ya que este no tiene la capacidad de notificar cuando terminó. Mientras tanto la CPU se mantiene ociosa a la espera, lo cual resulta ineficiente.

- E/S con interrupciones

A través de este procesamiento de E/S la CPU ya no chequea reiteradas veces los bits de estado que le indican que la operación de E/S ordenada al módulo finalizó. En este caso puede continuar con la ejecución del programa en simultáneo verificando en cada finalización de cada instrucción si hay alguna interrupción de finalización de E/S pendiente, cuando llega el caso, guarda el contexto, interrumpe el proceso y realiza el inicio de la gestión de la interrupción correspondiente para así hacerse de los datos que esperaba del dispositivo en caso de que la operación fuera de entrada y poder continuar con el programa retornando al contexto anterior.

- E/S con DMA

El DMA se encarga de todo el procesamiento de E/S para que la CPU no esté al pendiente de la ejecución de la operación y/o gestión de interrupciones, lo cual en transferencias masivas de E/S provocaría que este se mantuviera ocupada en gran parte para la ejecución de este tipo de operaciones haciendo que su performance se vea perjudicada.
La técnica aplicada por el DMA beneficia a la gestión de este tipo de operaciones haciendo que este tome el control total de la transferencia de E/S por lo cual este cuenta con un controlador (DMAC) el cual toma el control del bus del sistema de forma arbitraria para no tener conflictos con la CPU al momento de realizar la operación e indica las direcciones implicadas para el almacenamiento de los datos a transferir, así como también el tipo de operación que se va a realizar (in/out).
Como única tarea por parte de la CPU es inicializar al DMA y al periférico indicando los parámetros de transferencia tales como la el tipo de transferencia que tienen que realizar, la cantidad de bits o palabras que deben transferir, la ubicación a memoria o de memoria que se ve implicada en la operación y parámetros de control o de n de canales en caso de que se cuenten con DMA con varios canales. Una vez que realiza esto, la CPU se desliga del procesamiento de E/S
Una vez que el periférico indica que está ready, se inicia con la transferencia. Para esto el DMAC debe solicitar acceso al bus del sistema para llevar esto a cabo donde dependerá del tipo de transferencia indicado previamente, donde tendrá una coordinación de uso del bus con la CPU ya sea hasta llegar a transferir una cantidad determinada de palabras (TRANSFERENCIA POR ROBO DE CICLO), es decir, realizando varios ciclos de transferencia o hasta finalizar completamente la operación (TRANSFERENCIA MODO RAFAGA) que solo constaría de un ciclo de transferencia.
La transferencia de robo de ciclo consta en el uso del bus del sistema entre la CPU y el DMAC donde a priori se defina la cantidad fija de palabras a transferir por parte del DMAC, donde una vez llegada a esa cantidad se libere el bus y pase a la CPU y así sucesivamente. Esta forma si bien reduce la performance de la CPU resulta la forma más eficiente, ya que con la transferencia modo ráfaga la CPU genera más tiempo de espera.

Definición buscada, también lo suelen llamar modos de ubicación de E/S con Arquitectura Von Neumann:

En una arquitectura de Von Neumann, los dispositivos de entrada/salida (E/S) se ubican mediante dos modos: el modo de E/S programada y el modo de E/S por interrupción.
En el modo de E/S programada, el procesador controla directamente la transferencia de datos entre los dispositivos de E/S y la memoria principal. El programa del usuario es responsable de realizar las operaciones de E/S necesarias para leer o escribir datos desde o hacia los dispositivos de E/S.
En el modo de E/S por interrupción, el procesador es interrumpido por un dispositivo de E/S cuando se requiere una transferencia de datos. El procesador suspende temporalmente la ejecución del programa actual y atiende la interrupción, transfiriendo los datos necesarios entre el dispositivo de E/S y la memoria principal. Luego, el procesador reanuda la ejecución del programa interrumpido.
En ambos modos, la transferencia de datos se realiza a través del bus de datos de la arquitectura de Von Neumann, que conecta la memoria principal y los dispositivos de E/S. La memoria actúa como un intermediario para todas las transferencias de datos entre los dispositivos de E/S y la CPU, independientemente del modo de E/S utilizado.

El modo de E/S por DMA (Acceso Directo a Memoria, por sus siglas en inglés) también es un modo adicional de ubicación de E/S en una arquitectura de Von Neumann.
En este modo, se utiliza un controlador de DMA para transferir datos directamente entre los dispositivos de E/S y la memoria principal, sin la intervención del procesador. El controlador de DMA se encarga de iniciar la transferencia de datos y supervisar, mientras el procesador continúa ejecutando otros programas. Una vez que se completa la transferencia, el controlador de DMA interrumpe al procesador para indicar que se ha completado la operación de E/S.
El modo de E/S por DMA es especialmente útil para transferir grandes cantidades de datos entre dispositivos de E/S y memoria principal, ya que permite que el procesador se libere de realizar estas operaciones y se concentre en otras tareas. Esto puede mejorar significativamente el rendimiento y la eficiencia del sistema en general.

SEGMENTACIÓN DEL CAUCE (PIPELINE)

Manera de organizar la ejecución de procesos entre el hardware para llevar a cabo su procesamiento simultáneo, es decir, varios procesos a la vez descomponiéndose en fases o cauces permitiendo así el paralelismo entre las instrucciones que ejecuta un procesador.
Por ejemplo, en una segmentación de cinco etapas típica, las instrucciones pasan por las siguientes etapas en orden: buscar la instrucción, decodificar la instrucción, calcular operaciones, ejecutar la instrucción y escribir los resultados en memoria o en los registros. Cada etapa realiza una tarea específica, y la instrucción en procesamiento se mueve de una etapa a otra en cada ciclo de reloj.
La segmentación entonces se vuelve una manera de poder organizar el hardware con respecto a su diseño, pero aun así no dejan de depender del tipo de instrucciones que tengan que procesar es decir estos procesadores segmentación depende de su repertorio por lo que el procesador incrementa su productividad por el paralelismo, pero el tiempo de ejecución de las instrucciones sigue siendo la misma
Por ciclo los procesadores realizan las siguientes tareas:

- Búsqueda
- Decodificación
- Ejecución
- Acceso a Memoria
- Almacenamiento

No siempre todas las instrucciones necesitan pasar por todas las etapas, entonces esto genera que no haya un ciclo completo normal. Por lo tanto, el pipeline puede llegar a tener

Atascos del Cauce

Los atascos son situaciones que impiden que las instrucciones se ejecuten en el ciclo que le corresponde.
Distintos tipos de ATASCOS:

Estructurales:

Este caso sucede cuando las dos o más instrucciones necesitan hacer uso de un mismo recurso de hardware en un cauce. Por ejemplo, cuando ocurre un ciclo de búsqueda F y por otro lado una instrucción necesita realizar un acceso a memoria M, en estos casos, la instrucción con mayor prioridad en el pipeline realiza el uso del recurso haciendo que el otro pierda un ciclo para que este se libere.

Por dependencia de datos:

Cuando las instrucciones imponen una comunicación entre ellas mediante datos y esta no llega a estar disponible en el ciclo que se lo requiere por lo que genera ese atasco.

Cuando las instrucciones requieren el uso de datos que en algunos casos están siendo usados por otras instrucciones o están siendo accedidas por lo que esto repercute en la espera hasta su liberación.

Este tipo de atascos se clasifican en 3 tipos:

- Dependencia de datos verdadera (RAW): se produce cuando una instrucción intenta leer un operando antes de que la instrucción que lo escribe haya escrito ese operando en un registro o en memoria.
- Dependencia de datos de salida (WAW): se produce cuando dos instrucciones intentan escribir en el mismo registro al mismo tiempo. Esto puede provocar una intercalación de los datos escritos.
- Dependencia de control o antidependencia (WAR): se produce cuando una instrucción intenta escribir en un registro o en memoria antes de que una instrucción anterior haya terminado de leer los datos de ese registro o memoria.

Por dependencia de control:

Estos tipos de atasco ocurren cuando una instrucción en una etapa del cauce necesita esperar a que se complete la ejecución de una instrucción anterior en otra etapa del cauce antes de poder continuar su propia ejecución, por la cual se ve condicionada respecto a su resultado, es decir, cuando una instrucción condicional (por ejemplo, una instrucción de salto) depende del resultado de otra instrucción en el cauce. Si la instrucción condicional se evalúa como verdadera, se debe saltar a otra ubicación de la memoria para buscar la siguiente instrucción; si la instrucción condicional se evalúa como falsa, la siguiente instrucción en el cauce debe ser la instrucción siguiente en orden secuencial.

Si la instrucción condicional se encuentra en una etapa posterior en el cauce que la instrucción cuyo resultado se utiliza en la evaluación, puede producirse un atasco por dependencia de control, ya que la instrucción condicional debe esperar a que se complete la ejecución de la instrucción anterior antes de evaluar su condición. Esto puede provocar una disminución del rendimiento del procesador y afectar a la velocidad de procesamiento.

SOLUCIONES A LOS ATASCOS:

Soluciones para atascos estructurales:

Duplicación de recursos:

Aumentar los recursos o unidades para evitar qué dos instrucciones compitan por ellos. Lo ineficiente suele ser qué requiere más procesamiento la duplicación de recursos, por lo qué podría perjudicar la eficiencia del procesador.

Separación en memoria de instrucciones y datos:

Una técnica de diseño qué podría dar solución a estos atascos podría ser la separación en distintos lugares de memoria las instrucciones y los datos. Donde el procesador podría tener acceso a estos de forma paralela sin conflictos.

Mecanismo de sincronización de acceso al banco de registros:

> Y por último buscar un mecanismo de tal forma qué haya un acceso sincronizado al banco de registros. De esta manera la disponibilidad sería mayor. Como ejemplo sería qué en la primera mitad de los ciclos del reloj sean todas instrucciones de escritura y en la segunda mitad todas de lectura.

Soluciones para atascos de datos:

Para este tipo de atasco, en casos donde se tiene riesgos RAW, es decir, cuando se pretende leer cuando aún no ha sido escrito, existe dos tipos de técnicas como solución:

Técnica de detección de dependencia por software y por hardware

Por Software:

Para este tipo de atascos es necesario detectarlo a tiempo para poder prevenir qué pase. Se suele contar con unidades qué se encargan de esta detección o bien se cuenta con un compilador más complejo qué puede realizar esta tarea.

Una posible solución consiste en realizar la inserción de instrucciones (NOP), así poder generar un retardo de aquellas instrucciones que no deberían ejecutar su cauce porque generarían dependencia de datos.

Otra solución es la reordenación de las instrucciones para qué de alguna manera se mantengan separadas en el cauce, las que podrían generar conflictos teniendo en cuenta qué su ejecución no se dé “fuera de orden”.

Por Hardware:

La solución para este caso es el adelantamiento de operandos (forwarding). Donde las instrucciones cuentan con ellos apenas hayan sido calculados en su cauce, es decir, sin haber sido almacenados en registros o memorias. De esta manera las instrucciones pueden hacerse de este operando en menor tiempo, evitando así esperar su almacenamiento.
Esta posible solución también es aplicable para operandos de instrucciones o instrucciones de salto.

Soluciones para atascos de control:

Soluciones para casos típicos donde una instrucción espera la ejecución de otra instrucción, el cual aún no ha finalizado debido a la presencia de un salto condicional.

Predicción de saltos: esta técnica intenta predecir si un salto condicional se tomará o no antes de que se calcule la dirección de salto. Si la predicción es correcta, se evita el atasco de control y se sigue ejecutando el programa. Si la predicción es incorrecta, se deshace el trabajo realizado y se reanuda la ejecución del programa desde la nueva dirección de salto.

Predicción de saltos dinámica: esta técnica utiliza la historia de los saltos anteriores para predecir el comportamiento de los saltos futuros. Al analizar los patrones de comportamiento de los saltos, se pueden hacer predicciones más precisas y reducir el número de atascos de control. Por ejemplo, si un salto condicional se toma la mayoría de las veces, el procesador puede hacer una predicción de que se tomará de nuevo en el futuro y comenzar a ejecutar las instrucciones después de ese salto antes de que la dirección de salto sea conocida.

Saltos retardados: esta técnica permite que el procesador continúe ejecutando instrucciones después de un salto incondicional, lo que reduce el impacto de los atascos de control. El procesador retiene las instrucciones después del salto y las ejecuta en orden después de que el salto se haya completado.

Desenrollado de bucles: esta técnica consiste en duplicar o triplicar las instrucciones dentro de un bucle, de manera que se reduzca el número de saltos y se reduzcan los atascos de control.

COMPUTADORAS DE REPERTORIO REDUCIDO DE INSTRUCCIONES (RISC)

Es un tipo de arquitectura de procesadores qué se caracterizan por tener un conjunto de instrucciones reducido, lo que los hace más eficientes y rápidos.

Características principales:

- Repertorio reducido de instrucciones: Al contar con pocas instrucciones y sencillas, hace qué el procesamiento de los mismos sea más rápido y por ende su ejecución. Lo que hace qué este tipo de procesadores sean rápidos y eficiente en la ejecución
- Gran cantidad de registros de propósito general: Para el almacenamiento temporal en ejecución, cuentan con una gran cantidad de registros, así como también compiladores, qué gestionan de una manera eficiente, favoreciendo así la performance, tratando de qué sean pocos los cambios de localidad, como principal problema, qué se necesiten realizar y por ende, qué la transferencia de datos de registros usados sea menor.
- Énfasis en la segmentación de instrucciones: Estos procesadores funcionan con pipeline extenso, esto quiere decir, qué subdividen en varias etapas la ejecución de las instrucciones, y junto con distintos mecanismos, gestionan la ejecución de estos de una manera más eficiente y simultánea.
- Predicción de bifurcaciones: Utilizan técnicas para la especulación de bifurcaciones en las instrucciones, mejorando el rendimiento del procesador, permitiendo saber la ruta a donde tendrán qué bifurcar las instrucciones sin aún haber llegado a esa etapa.

Después de una cierta cantidad de estudios, entre ellos, el más conocido qué fue el estudio de Patterson donde analizada ambos procesadores RISC y CISC debido a qué este último si bien fue una solución para la época, tenía problemas de diseño tales como la complejidad en el software, la ineficiencia en la ejecución de instrucciones y la dificultad en la implementación de nuevas instrucciones, dado qué estos procesadores contaban con una gran cantidad y hacia qué sea complejo y difícil escribir y/o depurar los programas y había

casos donde solo había instrucciones qué eran propias de algunos procesadores CISC, por lo que si el programa se llegaba a ejecutar en otro procesador qué no lo fuera, había qué garantizar qué sea capaz de ejecutar el programa completo o volver a escribir el programa acorde a la compatibilidad del procesador qué se usará.

Lo que media en estos estudios era la frecuencia dinámica relativa, en donde se veía cuantas veces era usado en ejecución una instrucción o mejor dicho con qué frecuencia para poder comprender cómo se está usando el procesador en tiempo de ejecución.
Por ejemplo, si se determina que un conjunto particular de instrucciones se utiliza con frecuencia, esto puede indicar que es importante optimizar el procesador para ejecutar esas instrucciones de manera más eficiente.

Como solución de los cambios de localidad en los registros usados, está el

Cambio de Ventana de Registros:

Es una técnica de diseño en los procesadores RISC para poder intentar reducir los costos por cambios de localidad en tiempo de ejecución. A través de ella se almacenan los datos temporales y parámetros en pequeños conjuntos de registros qué necesiten los procedimientos o funciones de un programa. Por lo qué al momento de la ejecución, los procesadores harían cambios de ventanas de estos conjuntos para poder acceder a los datos sin necesidad de realizar cambios de localidad.

Es decir qué los registros qué se vayan a utilizar en estas ventanas funcionan o se usan de forma reservada a la función o procedimiento qué se le asocie.

Ventanas de registros solapadas:

Mismo concepto de lo qué son las ventanas de registros pero en este caso entre ventanas comparten registros temporales de forma subyacente. De esta manera se podría hacer hacer uso de menos registros por ventana, y por ende, al momento del uso de estos registros se podrían evitar transferencias de datos agregando así una optimización al uso de estas ventanas.

Ventanas de registros en buffer circular:

Con las propiedades qué tiene un buffer circular, de esta manera se podría optimizar el uso y asignación de las ventanas a los distintos procedimientos y funciones por la forma en qué las almacena, haciendo posible qué las ventanas puedan usarse de forma rotativa.

Cuando se produce una llamada a una función o procedimiento, la ventana de registros asociada se carga desde el buffer circular en el conjunto de registros disponible. Cuando se completa la función o procedimiento, la ventana de registros se guarda en el buffer circular y se carga la siguiente ventana de registros.

RISC vs CISC:

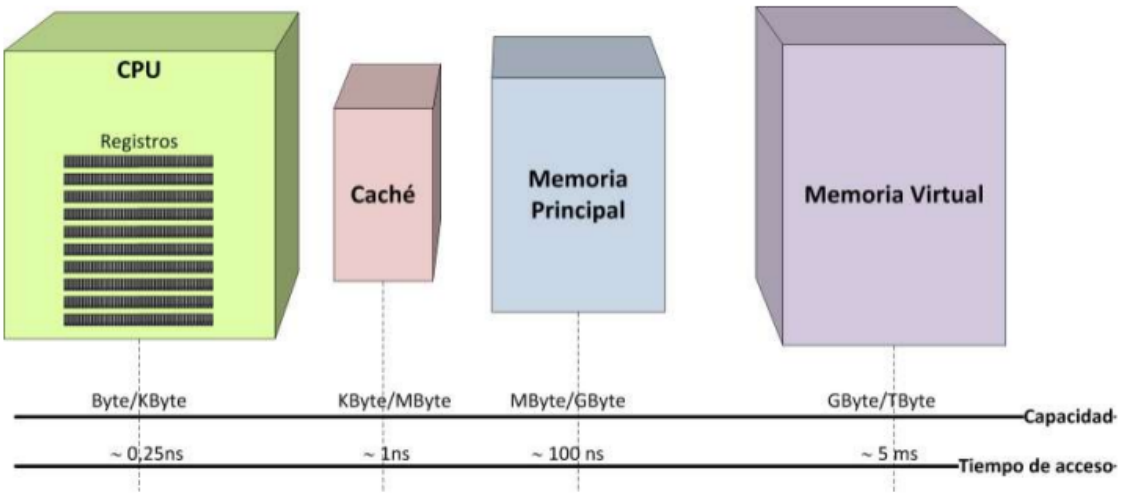
Por un lado, los partidarios de la arquitectura RISC argumentan que su diseño simplificado permite una mayor eficiencia y rendimiento en la mayoría de las aplicaciones, ya que se centra en la ejecución rápida de un número limitado de instrucciones básicas. Además, la arquitectura RISC reduce la complejidad del hardware, lo que puede reducir el costo y la complejidad del diseño del procesador y mejorar la eficiencia energética.
Por otro lado, los partidarios de la arquitectura CISC argumentan que su conjunto de instrucciones complejas permite realizar operaciones más complejas en una sola

instrucción, lo que puede reducir la latencia y el ancho de banda de la memoria. Además, la arquitectura CISC puede tener una mayor flexibilidad en términos de direccionamiento de memoria, lo que puede ser útil en aplicaciones que requieren un acceso frecuente a datos en diferentes partes de la memoria.

MEMORIA

La memoria adopta una jerarquía acorde a las propiedades de cada tipo de memoria qué existen, propiedades como tamaño, velocidad, capacidad y acceso.

Jerarquía de memoria:



¿Por qué se adoptó esta jerarquía en las memorias?

El porqué está definido en el principio de localidad de referencia qué define dos conceptos:

Localidad Temporal:

La localidad temporal se refiere a la tendencia de que los programas acceden varias veces a los mismos datos o instrucciones en un corto período de tiempo. Por ejemplo, si un programa accede a un dato en un ciclo de su ejecución, es probable que acceda al mismo dato nuevamente en ciclos posteriores. La localidad temporal se puede aprovechar para mejorar el rendimiento del sistema mediante la utilización de caché, ya que los datos e instrucciones accedidos recientemente se almacenan en la caché, y los accesos posteriores a estos datos e instrucciones pueden ser satisfechos rápidamente desde la caché.

Localidad Espacial:

La localidad espacial, por otro lado, se refiere a la tendencia de que los programas accedan a datos o instrucciones que se encuentran cerca en el espacio de direcciones. Por ejemplo, si un programa accede a un dato en una dirección de memoria específica, es probable que acceda también a datos cercanos en direcciones de memoria adyacentes. La localidad espacial se puede aprovechar para mejorar el rendimiento del sistema mediante la utilización de técnicas de prefetching o precarga, en las cuales se anticipa el acceso a datos o instrucciones cercanos y se traen a la caché antes de que sean necesarios.

En resumen, la localidad temporal se refiere a la tendencia de acceder varias veces a los mismos datos o instrucciones en un corto período de tiempo, mientras que la localidad espacial se refiere a la tendencia de acceder a datos o instrucciones cercanos en el espacio de direcciones. Ambas pueden ser aprovechadas para mejorar el rendimiento del sistema mediante el uso de técnicas de caché y prefetching.

Mecanismo de acceso a memoria:

1. El procesador emite una dirección de memoria que especifica la ubicación de los datos o instrucciones que desea acceder.

2. El controlador de memoria recibe la dirección de memoria y la decodifica para determinar la ubicación física en la memoria principal.

3. Si la ubicación de memoria solicitada está en la caché, el controlador de caché busca los datos o instrucciones solicitados en la caché y los devuelve al procesador. Si la ubicación de memoria solicitada no está en la caché, el controlador de memoria principal busca los datos o instrucciones solicitados en la memoria principal y los devuelve al procesador.

4. Una vez que se han recuperado los datos o instrucciones solicitados, el procesador los procesa según sea necesario (por ejemplo, ejecutando una instrucción o almacenando un dato).

Memoria Caché:

Pequeña cantidad de memorias de acceso rápido. Compuesto por líneas donde adentro se divide en etiquetas y marcos de almacenamiento. Posicionados entre la CPU y la memoria principal.

Cuando la CPU requiere algún dato, a quien primero se consulta es a la cache, donde si este lo contiene lo brindara al procesador, pero sino, se delega la búsqueda a memoria principal o en los niveles inferiores de la jerarquía, donde extrae el contenido del bloque a donde apunta la dirección en cuestión y se copia en la caché donde después se entrega al procesador y se almacena para futuras solicitudes del mismo recurso.

La caché usa las etiquetas para poder identificar los bloques qué almacena en relación a la memoria principal.

Entre la CPU y la caché se transfieren PALABRAS y entre caché y la memoria principal se transfieren BLOQUES cuando no son encontrados en caché a lo qué se le llama FALLOS.

Estos fallos son atendidos por hardware donde la CPU detiene su procesamiento hasta tener el recurso solicitado disponible para poder continuar.

Diseño de la memoria Caché

Aspectos qué se tienen en cuenta:

- La organización (tamaño, costos y niveles de caché)
- Políticas de ubicación
- Políticas de reemplazo
- Políticas de escritura

Dentro del concepto de organización de caché se incluyen de los aspectos de tamaño, costo y niveles de una memoria caché:

En su organización surgen tres tipo de correspondencia con respecto a cómo almacena los bloques de memoria en caché:

- Correspondencia Directa
- Correspondencia Totalmente Asociativa
- Correspondencia Asociativa en Conjunto

Correspondencia Directa:

Un bloque de memoria se asigna/mapea a una sola línea de caché, por lo tanto, cuando este bloque se actualice en memoria, la misma línea de caché asignada también será actualizada

Correspondencia Asociativa:

En este caso un bloque de memoria puede ser asignado o mapeado en cualquier línea de caché, por lo qué para este caso se trata de manera distinta la metodología de localización del bloque almacenado, los bloques ahora cuentan con una etiqueta qué sirve para poder ser ubicados en caché, lo qué hace qué sea más eficiente pero podría conllevar costos por hardware

Correspondencia Asociativa en Conjuntos:

En este tipo de correspondencia la caché se divide en bloques, por lo qué el mapeo con la memoria pasa a ser de a conjuntos de líneas (bloques) donde se le da el mismo tratamiento de etiquetado para identificar el bloque en caché.

Flujo de la organización de cache con correspondencia directa:

Cuando se realiza una lectura en MP, la dirección de memoria se divide en tres:
Etiqueta, Índice y Desplazamiento.

La etiqueta identifica al bloque de memoria qué corresponde a la dirección de memoria qué se solicitó.
El índice especifica la ubicación en caché donde deberá ser almacenado el bloque de memoria.
Y el desplazamiento especifica la ubicación dentro del bloque de memoria del operando solicitado.

La caché utiliza el índice para poder localizar su ubicación en caché y si las etiquetas coinciden, se retorna el dato solicitado desde la caché. En caso contrario, se carga desde memoria principal el bloque solicitado a caché en lo posición qué indica el índice y se actualiza la etiqueta con la nueva etiqueta del bloque solicitado para lo cual esto se realiza mediante un algoritmo de sustitución y como tarea final se retorna el dato solicitado.

Cuando surge una operación de escritura en memoria principal, la caché localiza el dato modificado y lo actualiza para mantener una consistencia.

Flujo de la organización de cache con correspondencia asociativa:

Las direcciones solo contienen etiqueta y desplazamiento:

Cuando se requiere un bloque de memoria, la búsqueda se hace mediante la etiqueta en caché (costoso) hasta encontrar una coincidencia. Esto debido a qué resulta ineficiente mantener o utilizar un índice actualizado cuando cualquier bloque puede almacenar en cualquier línea de caché.

Si se encuentra el bloque solicitado (acceso hit) se retorna el contenido del bloque al procesador. Si no es así, (acceso miss) se busca el bloque solicitado en memoria y se intenta almacenar en caché siempre y cuando sea posible, es decir, haya espacio. De no haber espacio se recurre a la utilización de algoritmo de reemplazo:

- LRU: Se basa en reemplazar el lugar de caché del bloque qué ha sido el menos reciente usado para lo qué lo define entonces es el tiempo de la última vez qué fue utilizado.
  Cada vez que se accede a un bloque, se actualiza el contador de tiempo a la hora actual. Cuando se produce un fallo de caché, se busca el bloque que tiene el contador de tiempo más antiguo (es decir, el que se ha utilizado menos recientemente) para ser reemplazado.

- FIFO: En este caso la caché adopta una arquitectura de cola donde cuando se genere un fallo en la caché, el primer o los primeros bloques en ser almacenados serán los elegidos para ser reemplazados.

- LFU: Se basa en reemplazar el lugar de caché del bloque, qué ha sido el menos frecuentemente usado, donde en este caso se define por la frecuencia o cantidad de veces que fue usado el bloque. Cuando se produce un fallo de caché, se busca el bloque que tiene el contador de frecuencia más bajo (es decir, el que se ha utilizado menos frecuentemente) para ser reemplazado.

- Aleatorio: Basado simplemente en reemplazar bloques al azar siempre qué se genere un fallo de caché (menos eficiente).

Tanto en LRU como en LFU puede llegar a ser costoso mantener actualizados los contadores tanto de tiempo como de frecuencia, así como también cuando los contadores entre bloques tienen las mismas cantidades. Para este caso se suelen combinar ambos algoritmos para poder llegar a definir de una forma más eficiente el o los bloques qué serán reemplazados.

Para el caso de FIFO, muchas veces puede no ser eficiente ya qué puede haber bloques importantes qué hay qué prevenir de reemplazo y para este algoritmo, esos aspectos no los tiene en cuenta así como tampoco la frecuencia y el uso, por lo qué LRU o LFU resultan ser más eficientes a diferencia de este.

Una vez realizado el reemplazo y almacenamiento en caché del bloque de memoria, se procede a devolver el contenido del bloque al procesador.

Flujo de la organización de cache con correspondencia asociativa en conjuntos:

Adopta las mejores características de las otras correspondencias. Un bloque de memoria es asignado a un conjunto de líneas en caché donde se almacenará cuando se requiera, esta metodología busca tener un reemplazo no directo dentro de este conjunto, buscando así aumentar la tasa de aciertos, optimizando la búsqueda en caché para el bloque qué requiere el procesador.

Políticas de escritura:

Lo que buscan las políticas de escritura es ante diferentes situaciones donde se accede a memoria principal, tratar en caché qué los bloques relacionados mantengan la misma consistencia de información, o sea, los datos se mantengan iguales en caso de modificaciones.

Las siguientes políticas son:

- Write-Through (escritura directa): Esta política pone como prioridad la integridad de los datos, por lo que si se almacena en caché, también se hará en memoria principal.

  Puede resultar conflictivo en caso de tener varias CPU con varios cachés, donde mantener esta integridad requiere de un gran tráfico de información a memoria principal, por lo que podría llegar a retrasar la escritura de los datos en tiempo de ejecución.

- Write-back (escritura-diferida): Esta política pone como prioridad el menor acceso a memoria posible, mejorando así la performance y reduciendo la cantidad de accesos, almacenando los datos en caché y solo actualizando en memoria principal cuando se necesite hacer un reemplazo.

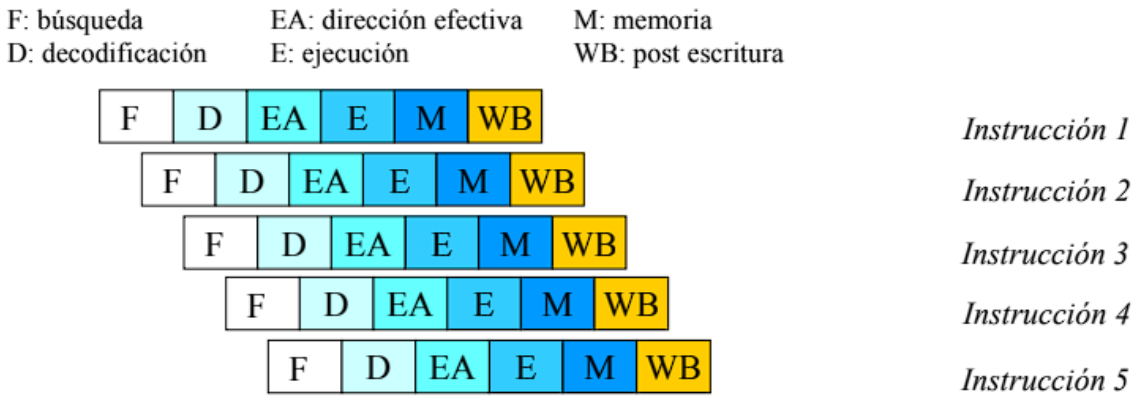
  Esto puede hacer qué en algún momento los datos en memoria principal sean erróneos o incorrectos.

Procesadores Superescalares:

Debido al avance lógico en los diseños de cauces segmentados, dieron lugar a dos técnicas de ejecución de mayores prestaciones: Procesadores Superescalares y Procesadores Supersegmentados

Procesadores Supersegmentados:

El enfoque qué tiene este tipo de diseño se basa en recortar los ciclos en sub-ciclos, haciendo qué se obtengan pipelines más extensos y por ende se requiera de más procesadores y recursos para poder llevar a cabo la ejecución paralela de más ciclos a la vez, agregando un grado más de complejidad en su diseño, escalabilidad, predicción de saltos y manejo de instrucciones fuera de orden ya qué es, una evolución de los procesadores superescalares con nivel más en las segmentaciones de instrucciones:



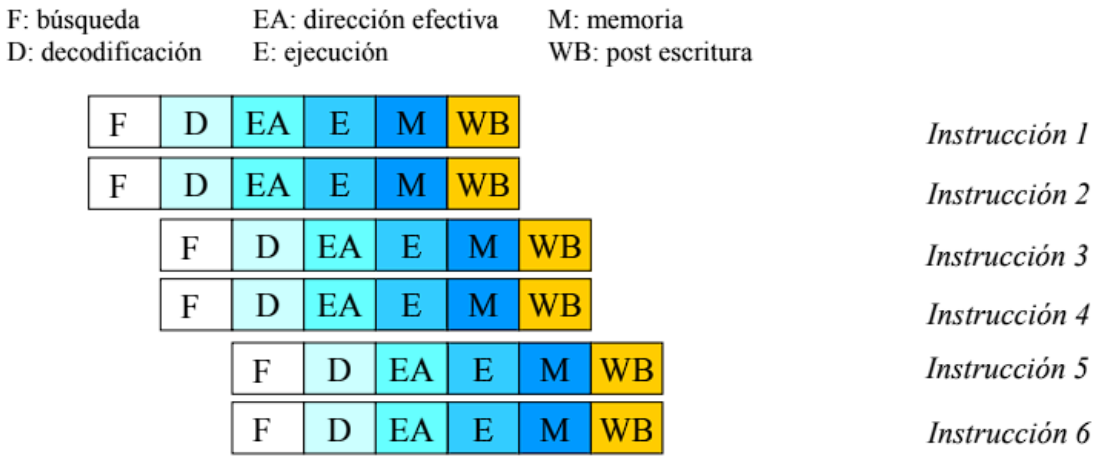
De esta manera se realizan estas etapas agregando otro grado de paralelismo entre sub-etapas.

Procesadores Superescalares:

Los procesadores superescalares son una clase de procesadores que emplean técnicas avanzadas de ejecución especulativa y paralelismo de instrucciones para mejorar el rendimiento. Estos procesadores son capaces de ejecutar varias instrucciones de manera simultánea, aprovechando el paralelismo que existe entre ellas, lo que permite que se ejecuten más instrucciones por ciclo de reloj. Además, los procesadores superescalares también suelen incluir mecanismos de predicción de salto avanzados y técnicas de renombre de registros para maximizar la capacidad de paralelismo en la ejecución de las instrucciones. Sin embargo, la implementación de estas técnicas puede llevar a la aparición de riesgos y problemas de rendimiento que deben ser gestionados adecuadamente.

Con este tipo de diseño de procesadores, el objetivo principal es realizar la ejecución de varias instrucciones, es decir, finalizar más de una instrucción a la vez, por lo que hace qué uno llegue a pensar qué esto generará, qué se requieran más recursos a tal punto de duplicar partes de CPU o ALU para una ejecución paralela o almacenar en memoria distintos datos en un mismo ciclo:

Ejemplo en diseño con 2 instrucciones en simultáneo/paralelo en un mismo ciclo:



- Uso de ventana de instrucciones

Limitaciones:

Si los recursos dependen de varias instrucciones y podrían llegar a generar algunas de estas limitaciones:

- Dependencia de datos verdadera (RAW)
- Dependencia relativa al procedimiento (similar a RAW pero entre procedimientos separados)
- Conflicto en los recursos (necesidad de acceso de varios procesos a la vez a un mismo recurso)
- Dependencia de salida (WAW)
- Antidependencia(WAR)

Concepto de paralelismo de instrucciones:

Es una medida qué representa la capacidad qué tienen los procesadores de poder ejecutar varias instrucciones de forma paralela. En esta medida se tienen en cuenta 3 cosas:

- Cantidad de instrucciones captadas por ciclo
- Cantidad de unidades funcionales o disponibles para el procesamiento
- El mecanismo qué se implementará para poder localizar las instrucciones independientes, lo cuales pueden ser:
    - Identificar paralelismo y ejecutar en paralelo en conjunto de instrucciones.

    - Usar renombre de registros: Técnica que permite que varias instrucciones en paralelo utilicen el mismo registro sin interferir entre sí. Esto se logra mediante la creación de copias físicas de los registros de la CPU, lo que permite que varias instrucciones utilicen el mismo nombre de registro sin que se produzcan conflictos.

    - Uso de ventana de instrucciones: Es un conjunto de instrucciones que están listas para ser ejecutadas y que se encuentran en diferentes etapas del proceso de ejecución, desde la captación hasta la ejecución. Las ventanas de instrucciones permiten que varias instrucciones se capturen, decodifiquen, renombren, reordenen y se emitan en paralelo

Políticas de emisión de superescalares:

Las políticas de emisión son un conjunto de reglas que determinan qué instrucciones se deben emitir y en qué orden:

- Emisión y finalización en orden: Las instrucciones se ejecutan en el mismo orden en el que fueron recibidos. Es más simple y seguro, pero menos eficiente en términos de paralelismo al no permitir la ejecución simultánea de varias instrucciones.

- Emisión en orden y finalización desordenada: Las instrucciones es emiten en el orden en el que aparecen en el programa y se pueden ejecutar en algún orden, qué resulte beneficioso para el paralelismo. Esto significa que, aunque las instrucciones se emiten en orden, el procesador no espera a que una instrucción se complete antes de emitir la siguiente. Los resultados obtenidos de las instrucciones se

almacenan en buffer para poder ser brindados a cualquier instrucción qué lo requiera.

- Emisión y finalización desordenada: Las instrucciones se emiten en el orden en el que fueron recibidas y pueden ser ejecutadas de forma diferente de manera qué se pueda aprovechar el paralelismo.

Para que la ejecución fuera de orden sea posible, el procesador utiliza una unidad de reserva (o buffer) llamada ventana de instrucciones, donde almacena las instrucciones emitidas. Cuando los operandos necesarios para una instrucción están disponibles, la instrucción se envía a una unidad de ejecución para su procesamiento.

Mecanismo de renombre de registros:

Técnica utilizada en procesadores superescalares para evitar las dependencias de datos. Consiste en asignar un número de registro temporal a cada operando en una instrucción en el momento en que se decodifica, en lugar de utilizar el número de registro físico original. Estos registros temporales se conocen como registros renombrados y se utilizan para calcular el resultado de la operación, es decir, qué ellos se almacenan los operados de las instrucciones. Con el uso de estos registros (qué implican más uso de hardware y de recursos) resultan beneficioso para el tratamiento de dependencias de salida y antidependencias, dándole enfoque solo al tratado de dependencias verdaderas.

Excepciones en procesadores superescalares:

Para este tipo de procesadores resulta ser un aspecto crítico a tener en cuenta, ya qué la ejecución fuera de orden puede complicar la detección o manejo de excepciones

En un procesador superescalar, las instrucciones se ejecutan fuera de orden, lo que significa que pueden ocurrir excepciones en un orden diferente al de la secuencia original de instrucciones. Además, como los procesadores superescalares ejecutan múltiples instrucciones al mismo tiempo, es posible que varias excepciones ocurran al mismo tiempo.

Existen mecanismos para poder gestionar las excepciones de una manera eficiente:

Especulación: los procesadores superescalares pueden especular sobre la ejecución de instrucciones, lo que significa que pueden ejecutar instrucciones antes de que se sepa si son necesarias. Si una excepción ocurre mientras se están ejecutando instrucciones especulativas, el procesador puede deshacer las instrucciones especulativas y continuar la ejecución normal.

Buffer de excepciones: los procesadores superescalares pueden tener un buffer de excepciones, que almacena información sobre las excepciones que ocurren durante la

ejecución. Cuando se detecta una excepción, se guarda la información de la excepción en el buffer y se continúa la ejecución de instrucciones no afectadas.

Reordenamiento de instrucciones: los procesadores superescalares pueden reordenar las instrucciones de tal manera que se minimice el impacto de las excepciones. Por ejemplo, si una instrucción tiene una alta probabilidad de generar una excepción, el procesador puede reordenar las instrucciones para que se ejecuten otras instrucciones antes de ella.

Predicción de excepciones: los procesadores superescalares pueden predecir si una excepción ocurrirá durante la ejecución de una instrucción y tomar medidas para evitar la excepción antes de que ocurra. Por ejemplo, si se sabe que una instrucción genera una excepción cuando se divide por cero, el procesador puede verificar si el divisor es cero antes de ejecutar la instrucción.

Procesamiento Paralelo:

La organización de las distintas computadoras, en lo que es el sistema de procesamiento paralelo cada vez con mejores prestaciones, están definidas en distintas categorías a lo que se llamó Taxonomía de Flynn:SISD, SIMD, MISD, MIMD

SISD (Single Instruction, Single Data):

Esta categoría define los procesadores qué ejecutan una sola secuencia de instrucción y qué procesan un solo conjunto de datos, lo cual es común en Monoprocesadores. Es la arquitectura más simple, ineficiente si se quiere lograr paralelismo, eficiente ante tareas simples y poco procesamiento de datos.

SIMD (Single Instruction, Multiple Data)

Esta categoría define una arquitectura de un solo conjunto de instrucciones con la capacidad de manejo de varios conjuntos de datos de forma simultánea, es decir, un conjunto por procesador que tiene un sector de memoria dedicado.
Esta categoría se relaciona con procesadores vectoriales y matriciales, donde cada uno, con su grado de complejidad, tienen las capacidades de manejar varias cantidades de datos por instrucción, ya sea en vectores o matrices de una manera eficiente y paralela.

MISD(Multiple Instruction, Single Data)

Categoría menos usada donde se trata de representar una arquitectura donde se transmitan entre todos los procesadores distintos secuencias de instrucciones donde a su vez utilicen un solo conjunto de datos

MIMD (Multiple Instruction, Multiple Data):

Conjunto de procesadores qué ejecutan conjuntos de instrucciones diferentes y utilizan conjuntos de datos diferentes por procesador, con acceso o a una memoria compartida o distribuida.
Se pueden sub-clasificar por las formas de comunicación y gestión de memoria:

Memoria Compartida: Multiprocesadores Simétricos SMP y sistemas NUMA

- Multiprocesadores Simétricos (SMP):

  Conjunto de procesadores físicos qué comparten una sola memoria y tienen conexión a través de un mismo bus de sistema. En este tipo de sistema, todos los procesadores tienen la capacidad de poder realizar las mismas acciones o tareas, ya qué el SO y el software los ven como un mismo recurso.

  Además, manejan un tiempo de acceso a memoria similar para todos los procesadores, a lo que se define como una arquitectura de memoria UMA.

  Sus ventajas son: Mayores prestaciones en lo que es la organización para poder realizar tareas de forma paralela. Una falla en alguno de los procesadores no detendrá la ejecución. Escalable en la integración de nuevos procesadores con las mismas propiedades para poder contar con más unidades.

  Sus desventajas se puede ver del lado del bus de uso compartido. Si bien todos los procesadores tienen acceso a una misma memoria física, al momento qué se requiera un acceso múltiple entre varios procesadores, se puede ver afectado el rendimiento por el acceso al bus que comparten y esto se puede seguir prolongando si la cantidad de unidades aumentara, claramente. Como posible solución ya se implementan métodos para que cada procesador pueda estar equipado con un caché jerárquica, para qué así se reduzcan la cantidad de acceso y reduzca así también la latencia, mejorando así el rendimiento.

  Si bien el uso compartido de recursos resulta ineficiente para SMP, existen métodos para poder optimizar y dar solución a estos casos.

- Sistemas UMA (UMA, NUMA, CC-NUMA):

  UMA: Define una arquitectura donde todos los procesadores tienen acceso a una misma memoria física mediante un mismo bus de sistema con un mismo tiempo de acceso a memoria hacia todas las regiones de memoria.

  NUMA: Mismo concepto de UMA, pero con la diferencia de qué el tiempo de acceso a memoria deja de ser uniforme para todos los procesadores, ya qué se hace uso de red para poder acceder a una memoria compartida, lo cual hará qué varíe el acceso en cada uno de los procesadores. Con respecto a la coherencia de caché, se realiza la comunicación entre los procesadores en caso de fallas mediante la red de interconexión qué mantienen los procesadores, es decir, a nivel de software.

  CC-NUMA: Mismo concepto de NUMA, pero con la diferencia qué para mantener la coherencia de las cachés, las comunicaciones de los procesadores se realizan hacia la memoria central compartida en caso de fallas y no entre

procesadores, es decir, qué se hace uso de un controlador (SMC) de memoria, qué se encarga de las comunicaciones entre los procesadores para poder mantener la coherencia, o sea, qué pasa a ser todo a nivel de hardware.

Memoria Distribuida: CLÚSTER

- Clúster: Tipo de sistema de procesamiento distribuido conformado por múltiples nodos interconectados mediante redes de alta velocidad donde cada uno hace uso y gestión de su propia memoria, es decir, cada uno gestiona su propio espacio de direcciones. Gran capacidad de prestaciones y escalabilidad, evolución de sistemas SMP.