Para final de Arquitectura de Computadoras

UNIDAD 1:

Qué es un Bus, tipos de buses, temporización y métodos de arbitraje.

Un bus es un camino de comunicación entre dos o más dispositivos, se trata de un medio de transmisión compartido. Al bus se conectan varios dispositivos, así cualquier señal transmitida por uno está disponible para que los otros puedan acceder a ella. Usualmente, un bus está constituido de varias líneas, cada una es capaz de transmitir señales binarias.

- Clasificación de buses:

Bus de sistema: el bus que conecta los componentes principales (CPU, memoria, E/S).

Bus de datos: transmite datos, se puede transferir un bit a la vez, el número de líneas determina cuantos bits se pueden transmitir en simultaneo.

Bus de direcciones: identifica el origen o destino de los datos, la anchura del bus determina la máxima capacidad de memoria posible en el sistema.

Bus de control: controla el acceso y el uso de líneas de datos y direcciones, puesto que estas líneas son compartidas por todos los componentes.

- Tipos de líneas del bus:

Las líneas del bus pueden ser de dos tipos, **dedicadas** o **multiplexadas**.

Las dedicadas se usan para una sola función, como por ejemplo una línea de direcciones. Las

multiplexadas permiten ahorrar espacio y costo, ya que se usa la misma línea para varias funciones, como por ejemplo para direcciones y datos. Usando otra línea como control, primero se coloca una dirección en el bus, y luego de un tiempo se usa para transmitir los datos.

- Temporización:

Forma en la que se coordinan los eventos en el bus. Pueden ser sincrónico o asincrónico.

Con temporización asincrónica, la presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo.

Con temporización sincrónica, la presencia de un evento en el bus esta determinada por un reloj, el bus incluye una línea de reloj a través de la que se transmite una secuencia en la que se alternan intervalos regulares de igual duración. Un único intervalo de 1 seguido de 0 se conoce como ciclo de reloj y define el time slot. Todos los dispositivos pueden leerlo y se establece que todos los eventos comiencen al principio del ciclo del reloj.

- Método de arbitraje:

Es el método con el que se supervisa el uso del bus, ya que normalmente son varios

los dispositivos que necesitan usarlo y se requiere que una sola unidad a la vez transmita en el bus. Los modos pueden ser **centralizados** donde un único dispositivo hardware, determinado controlador de bus, es responsable de asignar tiempos en el buso **distribuido,** en su lugar no existe controlador si no que cada módulo dispone de la lógica necesaria para el control de acceso, los módulos actúan conjuntamente para compartir el bus.

¿Qué es la segmentación del cauce de instrucción?

La segmentación de cauce o “pipeline” es una forma efectiva de organizar el hardware de la CPU para realizar más de una tarea al mismo tiempo. Consiste en dividir el ciclo de ejecución de las instrucciones en un conjunto de diferentes etapas que permitan ir haciendo operaciones paralelamente, así el procesador no tiene que esperar a terminar de ejecutar toda una sola instrucción para empezar a ejecutar las demás. Como es una técnica de mejora de hardware es invisible al programador y tiene como regla uniformizar sus etapas al tiempo de la más larga, consiguiendo beneficios como la mejora de rendimiento y el incremento de productividad (no el tiempo de ejecución de la instrucción, este si es el mismo).

Motivos de retardo en segmentación de cauce, tres tipos de atascos:

* Por dependencia de datos: ocurren cuando dos instrucciones se comunican por medio de un dato (ej.: una lo produce y la otra lo usa). Existen 3 clases de estas dependencias.

- Lectura después de Escritura (RAW, dependencia verdadera): una instrucción necesita leer un dato que una instrucción anterior no escribió.

- Escritura después de Lectura (WAR, antidependencia): una instrucción modifica un valor antes de que otra anterior que lo tiene que leer, lo lea.

- Escritura después de Escritura (WAW, dependencia en salida): una instrucción necesita escribir un registro que una instrucción anterior aún no escribió. Sólo se da si se deja que las instrucciones se adelanten unas a otras.

* Por dependencia de control: ocurren cuando la ejecución de una instrucción depende de cómo se ejecute otra (ej.: un salto y los 2 posibles caminos). Problema ocurrente en la etapa de decodificación, la cual evalúa si la condición de salto es verdadera o no. Si salta cuando no debería tiene que descartar la instrucción y cargar la que corresponde, generando un Branch Taken Stall (BTS).
* Por dependencia estructural: provocados por conflictos por los recursos. Dos o más instrucciones necesitan utilizar el mismo recurso simultáneamente en el mismo ciclo.

Segmentación del cauce: describa los métodos y técnicas utilizadas para disminuir o evitar las paradas que afectaran al funcionamiento de los cauces.

* Soluciones para los atascos por dependencia de datos:

Técnicas de hardware: CPU más compleja y costosa, es más potente y cómodo que las técnicas de software. Conocida como técnica de adelantamiento o forwarding, consiste en pasar el dato obtenido de la instrucción a las instrucciones que lo necesiten como operando. Si el dato está disponible a la salida de la ALU se lleva a la entrada de la etapa correspondiente sin esperar la escritura (buffer de resultados).

Técnicas de software: el programador o compilador es el que deberá reordenar o cambiar instrucciones que generan dependencias en el código sin afectar los resultados. Hay dos formas: Insertar instrucción NOP, el problema de esta es que genera retardo sin utilidad o reordenar las instrucciones y generar máxima separación con las instrucciones RAW.

* Soluciones para los atascos por dependencia de control:

Hay que tener en cuenta los dos tipos de saltos.

Incondicional: la dirección de destino se debe determinar lo más pronto posible, para reducir la penalización. Se puede adelantar la resolución del salto en la etapa de decodificación, donde en ella se decodifica y se sabe que es un salto.

Condicional: introduce riesgo adicional por la dependencia entre la condición de salto y el resultado de una instrucción previa.

Técnicas de hardware: consiste en predicciones de salto para evitar la parada.

Las estáticas son: predecir que nunca se salta: asume que el salto no se producirá (siempre capta la siguiente instrucción) o predecir que siempre se salta: asume que el salto se producirá (siempre capta la instrucción destino del salto) o predecir según el código de operación, hay instrucciones con más probabilidades de saltar.

Las dinámicas son: Branch Target Buffer (BTB), utiliza un buffer para predecir el salto, su estrategia: si saltó la vez anterior, se predice que va a saltar nuevamente. Si no saltó la vez anterior, predecir que no salta. Si el programa tiene muy pocos ciclos, puede no ser conveniente usar este método, ya que la penalización son siempre 4 atascos (2 por miss prediction + 1 por cargar la instrucción + 1 por actualización del buffer).

Técnicas de software: Delay Slot, cambia la manera en la que se ejecutan los saltos. Retarda el salto un ciclo más, o sea se ejecuta en la siguiente línea. Tiene como desventaja que los programas tienen que modificarse para que esta técnica funcione, hay que hacer un código específico para este método.

* Soluciones para los atascos por dependencia estructurales:

Replicar, segmentar o realizar turnos para el acceso a las unidades funcionales en conflicto.

• Duplicar los recursos de hardware necesarios como duplicar los recursos de la ALU.

• Separar la memoria de instrucciones de la de datos.

• Turnar el acceso al banco de registros. Ej. las escrituras en la primera mitad del ciclo y lectura en la segunda mitad del ciclo.

Subrutinas

Innovación en el desarrollo de lenguajes de programación, una subrutina o procedimiento es un programa con identidad propia que se incorpora en un programa más grande. Pueden invocarse desde cualquier punto del programa mediante un CALL y retornar al programa con el RETURN. Brindan reutilización de código, modularización, facilidad en lectura y comprensión, limita errores y permite independencia de variables si se utilizan parámetros.

Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el comportamiento con anidamiento de múltiples procedimientos/funciones.

Los parámetros nos permiten intercambiar datos con subrutinas, ofrecen independencia entre las variables del programa y las subrutinas o sea los procedimientos no quedan ligados a variables del programa.

Vía registros: en este método se pasan los argumentos a través de registros de la CPU, son de alta velocidad, pero el número de registros es la principal limitación. Además, es importante documentar que registros se usan.

Vía memoria: los argumentos se almacenan en posiciones de memoria, no hay tal limitación, pero puede ser menos eficiente debido al tiempo necesario para acceder a la RAM. Se suele utilizar cuando hay demasiados argumentos o cuando se trabajan con tipos de datos grandes.

Vía pila: es el método más ampliamente usado, considerado el verdadero “pasaje de parámetros”. Es independiente de memoria y registros.

Paso 1: antes de llamar a la función, se deben apilar los parámetros y la dirección de retorno.

Paso 2: los argumentos se colocan en la pila en orden. El control salta a la función y comienza a ejecutar su código, durante la ejecución el puntero de la pila se utiliza para acceder a lo almacenado en la pila.

Para almacenar y recuperar valores de la pila se realiza mediante operaciones push (empujar) y pop (sacar).

Aquí se puede llamar a otra subrutina o regresar a la anterior, si se vuelve a hacer una llamada también se debe apilar los parámetros a pasar a la subrutina y la dirección de retorno de esta. La subrutina llamada repite los pasos anteriores más el que sigue. Sino se hace otra llamada a subrutina se deberá seguir solo con el último paso:

Paso 3: retorno de la función, cuando esta ha completado su ejecución, el valor de retorno (si lo hay) se coloca en la pila. El puntero de la pila se mueve de nuevo al estado antes de la llamada, liberando el espacio reservado. Para regresar correctamente del procedimiento es necesario desapilar todo lo que apilo y volver a cargar a cargar el valor del puntero base que tenía antes de entrar a la subrutina.

Sistemas embebidos

Electrónica y software dentro de un producto, es un sistema de computación basado en un microprocesador, diseñado para realizar una o algunas pocas funciones dedicadas, posee una memoria pequeña. En contraposición a una computadora de uso general los sistemas embebidos se usan para cubrir necesidades específicas. Ejemplos de estos sistemas lo podemos encontrar en videocámaras, hornos microondas, lavarropas, sensores, sistemas de seguridad, y demás sistemas automatizados.

UNIDAD 2:

¿Qué es una interrupción? Mencionar cuales son las fuentes/tipos de interrupción.

Una interrupción es un mecanismo, que prácticamente todas las computadoras disponen, mediante el que otros dispositivos pueden interrumpir el procesamiento normal de la CPU.

* Interrupciones por software:

Mecanismo mediante el cual los programas pueden llamar al sistema operativo para que realice alguna funcionalidad con los dispositivos. Es similar a una subrutina, entonces a partir de una solicitud de interrupción, deja de ejecutarse el programa de usuario y comienza a ejecutarse el S.O que a su vez llama al código de un driver, y es el driver el que interactúa con el dispositivo.

* Interrupciones por hardware:

Son generadas por los dispositivos de E/S, son las “verdaderas” interrupciones, el sistema de cómputo tiene que manejar estos eventos no planeados que no están relacionadas con el proceso de ejecución del momento. Son conocidos como pedido de interrupción, alertan al procesador la necesidad de atención, fundamentales para permitir la comunicación y coordinación entre el hardware y software para responder eficientemente a eventos y condiciones específicas.

* Traps/ excepciones:

- Programa: generadas por alguna condición que se produce como resultado de la ejecución de una instrucción, tal como desbordamiento aritmético, división por cero, intento de ejecutar una instrucción maquina inexistente o intento de acceso fuera del espacio de memoria permitido.

- Temporización: generadas por un temporizador interno al procesador que le permite al sistema operativo realizar algunas funciones de manera regular.

- E/S: generadas por un controlador de entrada/ salida para indicar la finalización de una operación o para avisar ciertas condiciones de error.

- Fallo de hardware: generadas por un fallo tal como la falta de potencia de alimentación o fallo de paridad en memoria.

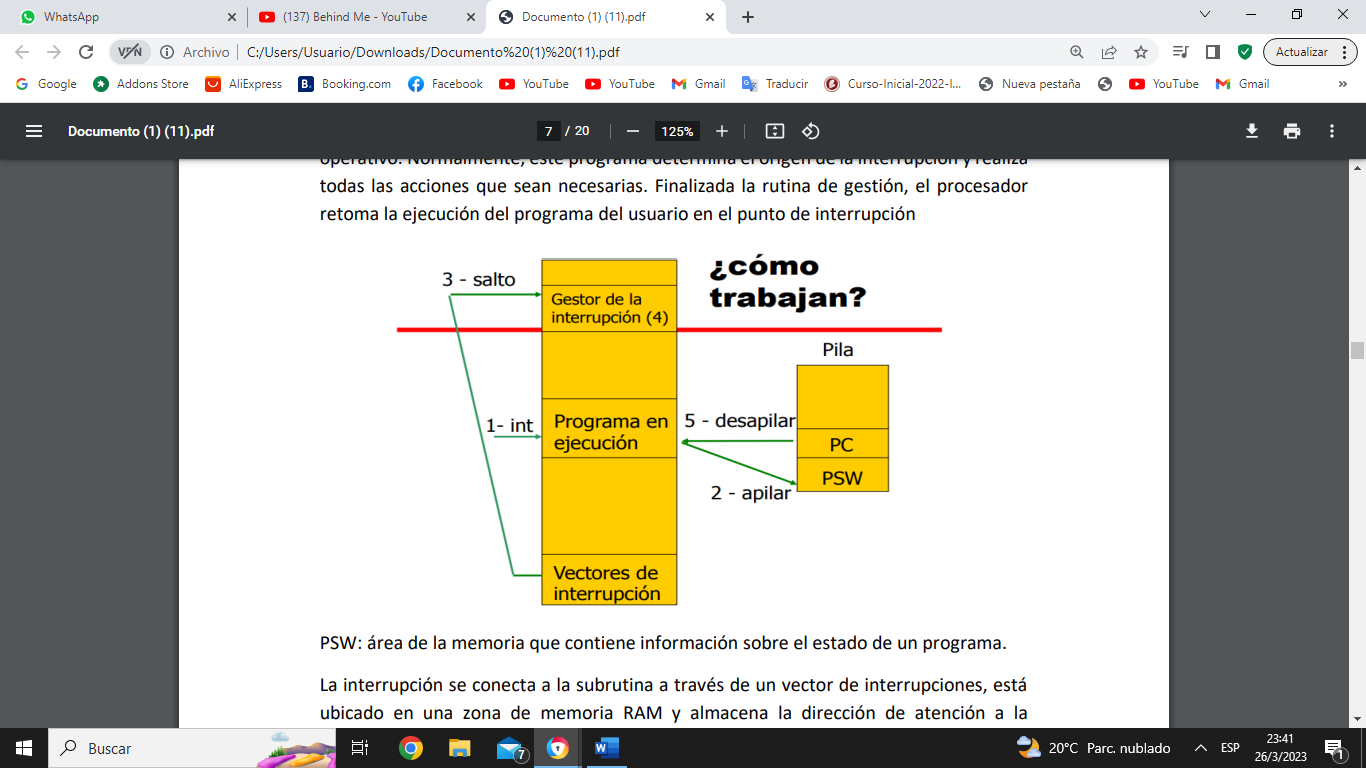
Describir el mecanismo de interrupción

Es un mecanismo en el cual un dispositivo “interrumpe” al procesador para solicitar su servicio cuando esté preparado para intercambiar datos con él. Esto permite que el procesador no tenga que esperar, comprobando repetidamente el estado del módulo, hasta que este pueda transmitir o recibir los datos. Dado que los dispositivos son mucho más lentos que el

procesador, no usar interrupciones haría caer el nivel de prestaciones de todo el sistema. Por esto la CPU repite esta secuencia:

Comprueba si hay alguna interrupción (la indica un flag), si no hay interrupción se capta la siguiente instrucción, si la hay 1ero suspende la ejecución del programa en curso y guarda su contexto, esto significa almacenar la dirección de la siguiente instrucción que se iba a ejecutar (contenido actual del Program Counter) y cualquier otro dato relacionado con la actividad en curso del programa. 2do carga el PC con la dirección de comienzo de una rutina de gestión de interrupción y a continuación, el CPU prosigue con el ciclo de captación y accede a la primera instrucción del programa de gestión de interrupción, que dará servicio a la interrupción.

Generalmente este programa forma parte del sistema operativo donde este determina el origen de la interrupción y realiza todas las acciones que sean necesarias. Finalizada la rutina de gestión, el procesador retoma la ejecución del programa de usuario en el punto que fue interrumpido.



La interrupción se conecta a la subrutina a través de un vector de interrupciones, donde almacena la dirección de atención a la interrupción. Los vectores de interrupción se almacenan en una tabla en ese espacio de memoria, cuando se atiende una petición de interrupción de número n, el sistema transfiere el control a la dirección indicada por elemento n-ésimo de dicha tabla.

Jerarquía de interrupciones. Describir el tratamiento de múltiples interrupciones.

Para las prioridades se consideran: interrupciones no enmascarables, cuando no pueden ignorarse, indican eventos peligrosos o de alta prioridad, por lo general las interrupciones de Hardware son las de prioridad más alta. Y enmascarables, cuando pueden ser ignoradas, con instrucciones podemos inhibir la posible solicitud.

- Si hay múltiples interrupciones, se pueden seguir dos alternativas:

1) desactivar las interrupciones mientras se está procesando una interrupción. Una interrupción inhabilitada significa que el procesador puede y debe ignorar la señal de petición. Si se produce una interrupción en ese momento se mantiene en pendiente y será examinada por el procesador una vez que estén habilitadas. Se manejan en orden secuencial.

2) definir prioridades, una interrupción de prioridad mas alta puede interrumpir a un gestor de interrupción menor. Cuando se ha gestionado la interrupción de mayor prioridad el procesador vuelve con la interrupción previa.

- Administrar múltiples interrupciones por hardware, reconocimiento de interrupciones.

Puede ocurrir que haya múltiples módulos de E/S, y también que cada módulo tenga múltiples dispositivos conectados. Debe disponerse de una técnica para discernir cuál dispositivo es el que generó la interrupción.

Múltiples líneas de interrupción: consiste en proporcionar varias líneas de interrupción entre el procesador y los módulos de E/S (es costoso, difícil de mantener, por lo tanto, se terminan usando varios dispositivos en cada línea y hay que usar uno de los otros métodos).

Consulta software (polling): Consiste en que cuando el procesador detecta una interrupción se produce una bifurcación a una rutina de servicio de interrupción que se encarga de consultar a cada módulo de entrada salida para determinar el módulo que ha provocado la interrupción. La principal desventaja es el tiempo que consume.

Interrupciones vectorizadas: en este método, el módulo de E/S debe en primer lugar disponer del control de bus antes de poder activar la línea de petición de interrupción. La línea de pedido es compartida, el dispositivo que quiere interrumpir además de la señal de pedido de interrupción debe colocar en el bus de datos un identificador (vector), la CPU emplea el vector como puntero para acceder a la rutina de servicio.

Describa las características del controlador de interrupciones PIC.

Si el procesador tiene una única entrada de pedido de interrupciones o si tenemos varios productores de interrupciones, se soluciona con el dispositivo controlador programable de interrupciones. Un PIC es un dispositivo controlador programable de instrucciones, usado para gestionar y priorizar las interrupciones generadas por dispositivos periféricos. Cuando el dispositivo tiene múltiples salidas de interrupción a imponer, las asignará un orden de prioridad relativa.

Dispone de los registros: EOI que le avisa al PIC que la interrupción fue atendida. IMR habilita o deshabilita las interrupciones. IRR indica cuales dispositivos externos desean interrumpir. ISR indica cual dispositivo está siendo atendido. INT 0, INT 1, INT2, INT3… para las interrupciones donde carga el valor del vector de interrupción correspondiente.

Describa las características que diferencian a los procesadores RISC respecto de los CISC.

CISC = procesador con conjunto de instrucciones complejas.

RISC = procesador con conjunto de instrucciones reducido.

|  |  |
| --- | --- |
| **CISC** | **RISC** |
| Características:  Mayor cantidad de instrucciones y más complejas.  Se suelen ejecutar en varios ciclos de reloj.  Estas instrucciones tienen una longitud variable.  Pueden soportar varios modos de direccionamiento.  Utiliza pocos registros.  Pretenden mejorar la eficiencia de la  ejecución, a través de secuencias detalladas en micro código, y da soporte a lenguajes de alto nivel más complejos. | Características:  Repertorio de instrucciones limitado y sencillo.  Solo toman un ciclo de reloj.  Estas instrucciones tienen una longitud fija (32 bits).  Tiene una limitación en la cantidad de modos de direccionamiento.  Utiliza gran número de registros de uso general.  Se hace un gran énfasis en el software lo principal es la optimización de la segmentación de instrucciones. |
| Procesadores cisc:  Los Intel 8086, 8088… | Procesadores risc:  MIPS, ARM, POWER PC, DEC APHA… |
| Ventajas:  Pocas instrucciones para ejecutar una tarea, esto genera un código corto así necesitando menos memoria RAM. El compilador requiere menos esfuerzo para traducir las instrucciones | Ventajas:  Mejor rendimiento gracias a su corto repertorio de instrucciones. Requiere una menor cantidad de hardware, esto hace que sean más económicos. Presenta menos consumo energético y calor. |
| Desventajas:  El rendimiento es afectado por necesitar muchos ciclos de reloj.  Diseño físico del procesador es más complicado, requiere más transistores, lo que conlleva mayor consumo, mayor requisito de espacio y mayor generación de temperatura. | Desventajas:  El rendimiento es variable según la aplicación que este ejecutando el procesador. Necesitan una memoria más rápida para guardar muchas instrucciones y ser capaz de ejecutarlas en un tiempo corto. |

Conclusión: En cuanto a definir cual arquitectura de set de instrucciones es mejor no hay una respuesta definitiva. Cada una tiene su enfoque particular y sus fuertes según la funcionalidad que tienen que cumplir, tampoco es comparable porque no existen dos máquinas cisc y risc semejantes, tampoco hay un conjunto de programas de prueba definitivo y es difícil separar los efectos del hardware. Además, las maquinas actuales son una mezcla de ambas arquitecturas.

UNIDAD 3:

E/S: Además del procesador y la memoria el tercer elemento clave de un sistema informativo es un conjunto de módulos E/S. Cada módulo se conecta al bus de sistema y controla uno o más dispositivos periféricos. Estos módulos contienen lógica para realizar una función de comunicación entre el periférico y el bus.

Razones por las que los dispositivos no se conectan directamente al bus del sistema:

- Gran variedad de periféricos con diversos métodos de funcionamiento.

- Diferencia de velocidad en la transferencia de datos, la de los periféricos menor al del CPU.

- Los periféricos suelen utilizar formatos de datos y longitudes de palabra diferentes a los de la computadora a la que están conectados.

Funciones principales de un módulo E/S

* Control y temporización

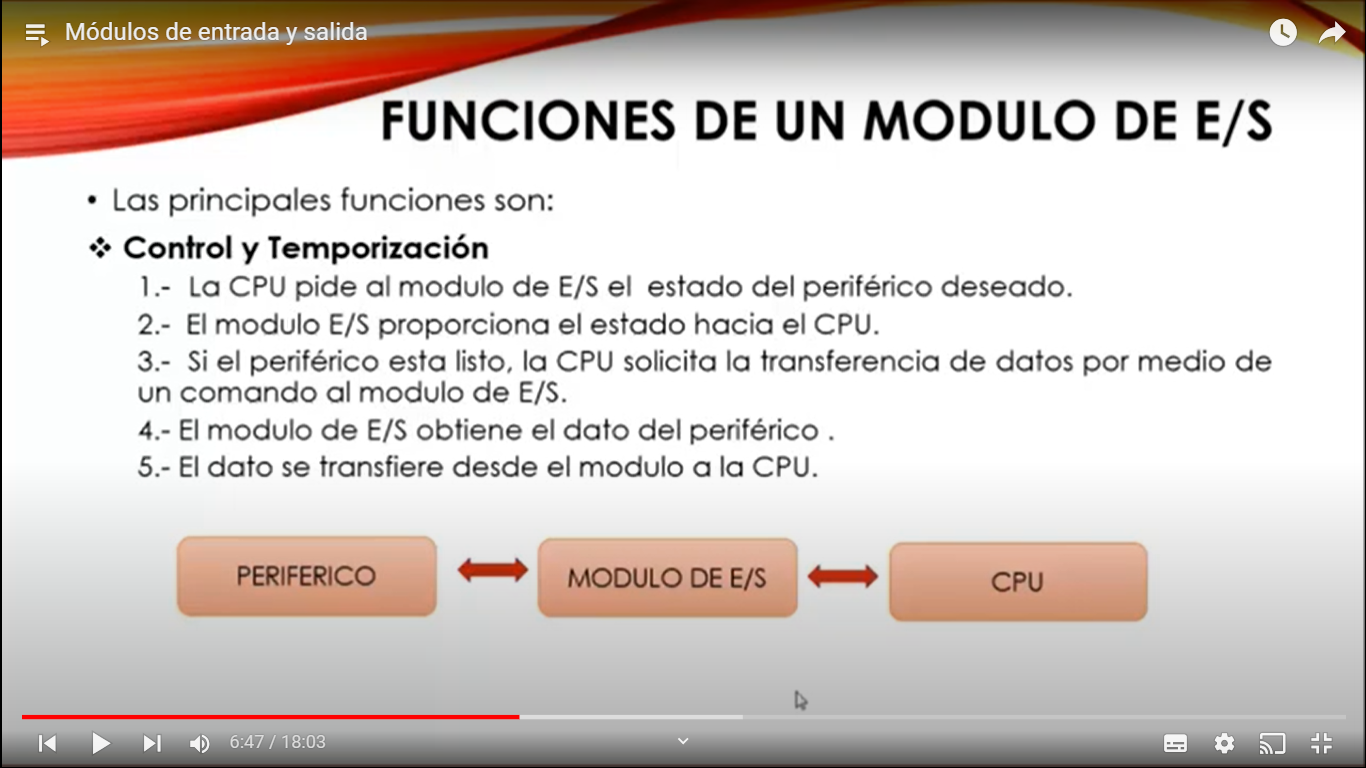
1) la CPU pide al módulo E/S el estado del periférico deseado.

2) el módulo E/S proporciona el estado al CPU.

3) si el periférico está listo para la transferencia de datos, la CPU solicita la transferencia por medio del comando al modulo E/S.

4) el módulo de E/S obtiene el dato del periférico.

5) el dato se transfiere desde el módulo a la CPU.



* Comunicación con el procesador y con los dispositivos

Decodificación e intercambio de órdenes, datos, información de estado, reconocimiento de dirección.

* Almacenamiento temporal de datos

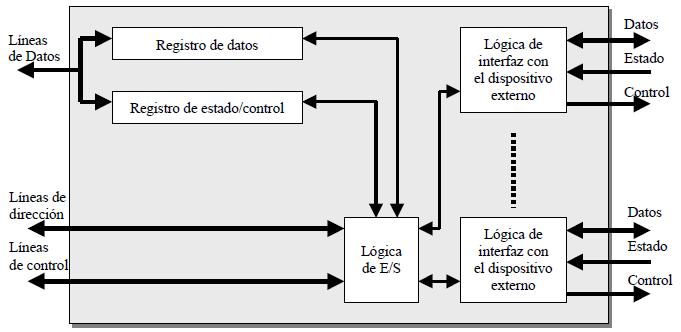
Los datos se almacenan temporalmente en el módulo E/S y luego se envían al periférico a la velocidad de este bus.

* Detección de errores

El modulo E/S es responsable de una detección de errores, puedes ser defectos mecánicos y eléctricos.

Estructura de E/S

El módulo se conecta con el resto de la máquina a través del bus del sistema. Los datos que se transfieren a y desde el módulo, se almacenan temporalmente en el registro de datos. Posee registro de estado que proporciona información del estado actual y que también puede funcionar como control, para recibir información de control del procesador. También contiene la lógica que interactúa con el procesador a través de una serie de líneas de control, usadas por el procesador para proporcionar las órdenes al módulo de E/S. Algunas de las líneas de control pueden ser usadas por el módulo E/S, por ejemplo, para señales de arbitraje y estado. El módulo debe ser capaz que reconocer y generar las direcciones asociadas a los dispositivos que controla. Cada módulo E/S tiene una dirección única, y si controla más de un dispositivo, posee la lógica específica para cada una de las interfaces de los dispositivos que controla.



Interfaz con el dispositivo interno, nos da acceso al banco de registros del módulo E/S.

Interfaz con el dispositivo externo, conexión con el periférico.

La lógica es la responsable del paso de la información entre el interfaz interno y externo.

Formas de direccionamiento de E/S

Cuando el procesador, la memoria y las E/S comparten un bus común, son posibles dos modos de direccionamiento:

* E/S asignada en memoria: los periféricos se representan como direcciones de memoria. Los programas acceden a estos dispositivos utilizando las mismas instrucciones de carga y almacenamiento que se utilizan para acceder a memoria principal. No hay ordenes específicas para E/S, es fácil de implementar, es de programación eficiente.
* E/S aislada: en este enfoque se asignan direcciones especificas paras las operaciones de E/S con un conjunto limitado de instrucciones que son diferentes de las utilizadas para acceder a memoria principal. Permite un mayor control sobre el acceso a los dispositivos y puede evitar accesos donde la memoria se sobrescribe por error. Mas robusto pero seguro en términos de programación.

Describa las posibles técnicas que pueden utilizarse para realizar las operaciones de E/S

* E/S programada: el procesador tiene que esperar un tiempo considerable a que el modulo de E/S este preparado para realizar la operación. El procesador espera comprobando repetidamente el estado del módulo, así degradándose significativamente el rendimiento de la CPU porque permanece ociosa hasta que el modulo termine.
* E/S por interrupción: la CPU no tiene que esperar la finalización de la tarea de E/S, puede seguir procesando. La CPU emite un comando de E/S mientras ejecuta instrucciones y es interrumpido por el módulo cuando ya haya terminado su tarea. La CPU puede trabajar con varios programas y va comprobando si hay alguna interrupción al final de cada ciclo de instrucción.

Si hay que mover grandes volúmenes de datos, se requiere una técnica más eficiente, el acceso directo a memoria. En este modo, el módulo de E/S y la memoria principal intercambian datos directamente.

* E/S por acceso directo a memoria (DMA): forma de transferencia que utiliza un dispositivo (CDMA) que controla los intercambios de datos entre la memoria principal y un periférico sin intervención de la CPU. Implica un módulo adicional en el bus del sistema capaz de imitar al procesador y tomar el control del sistema desde la CPU, este solo sirve para copiar datos, es menos complejo y su función es complementarla quitándole este trabajo.

Características funcionales de DMA

DMA (Direct Memory Access) es un módulo adicional en el bus de sistema, que tiene la capacidad de imitar al procesador y tomar el control del sistema, con la función de realizar transferencias de E/S de datos, de manera directa entre los periféricos y la memoria, a través del bus, mejorando la velocidad de la transferencia y haciendo el sistema más eficiente.

Etapas de una transferencia DMA: 1. Inicialización de la transferencia, la CPU configura el CDMA. 2. Realización de la transferencia, la CPU no interviene en nada, queda libre. 3. Finalización de la transferencia, la CPU debe atender la interrupción que se generó.

El CDMA debe actuar como maestro del bus durante la transferencia DMA y debe ser capaz de: 1. Solicitar el uso del bus mediante las señales y la lógica de arbitraje necesarias. 2. Especificar la dirección de memoria sobre la que se realiza la transferencia. 3. Generar las señales de control de bus. 4. Indicar tipo de operación: lectura/ escritura. 5. Sincronizar todo ya que su objetivo es no pisarse con la CPU.

Modos de transferencia DMA

* Por ráfagas (por bloques): cuando la CPU concede el bus, el DMAC no lo libera hasta haber finalizado la transferencia de todo el bloque de datos completo. Ventajas, la transferencia se realiza de forma rápida. Desventajas, durante el tiempo que dura la transferencia la CPU no puede utilizar el bus con memoria, lo que puede degradar el rendimiento del sistema.
* Por modo robo de ciclo (demanda): cuando la CPU concede el bus al DMAC, se realiza la transferencia de un byte y después el DMAC libera el bus, este también solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo. Ventajas, no degrada el rendimiento del programa. Desventajas, la transferencia tarda más tiempo en llevarse a cabo.

UNIDAD 4:

CACHÉ

Memoria caché: de tipo volátil, es un buffer especial de almacenamiento, pequeño y más veloz que otras memorias. Se ubica entre la CPU (o adentro de ella) y la memoria principal, es usada para guardar una copia de instrucciones y datos de la memoria que el procesador va a necesitar para un rápido acceso, la caché se va a encontrar en constante renovación.

Entonces, cada vez que el sistema quiere acceder a un nuevo dato, este es traído de la memoria principal a la caché, así cuando se necesite recurrir al mismo dato, el sistema se dirigirá directamente a la caché generando un proceso más rápido. Esto funciona debido a los principios de localidad. Principio de localidad temporal: los elementos de memoria referenciados recientemente volverán a serlos en el futuro. Principio de localidad espacial: los elementos de memoria cuyas direcciones están próximas a los últimos referenciados serán referenciados en el futuro.

(niveles de caché) La transferencia entre la CPU y la memoria caché es rápida, y a nivel transferencia de palabras. La transferencia entre la memoria caché y la memoria principal es más lenta y a nivel transferencia de bloques.

(tamaño) Cuanto mayor sea el caché mayor será el numero de compuertas involucradas en el direccionamiento del caché, generando que sea mas lento. Entonces las cache mas grandes son menos veloces, también limita su tamaño el área disponible en la placa. Por el rendimiento del cache es sensible y dependiente de la carga de trabajo es imposible llegar a un único tamaño óptimo de cache y debido a que hay menos líneas de caché que bloques de memoria principal, se necesita un algoritmo para mapear los bloques de memoria principal en líneas de caché.

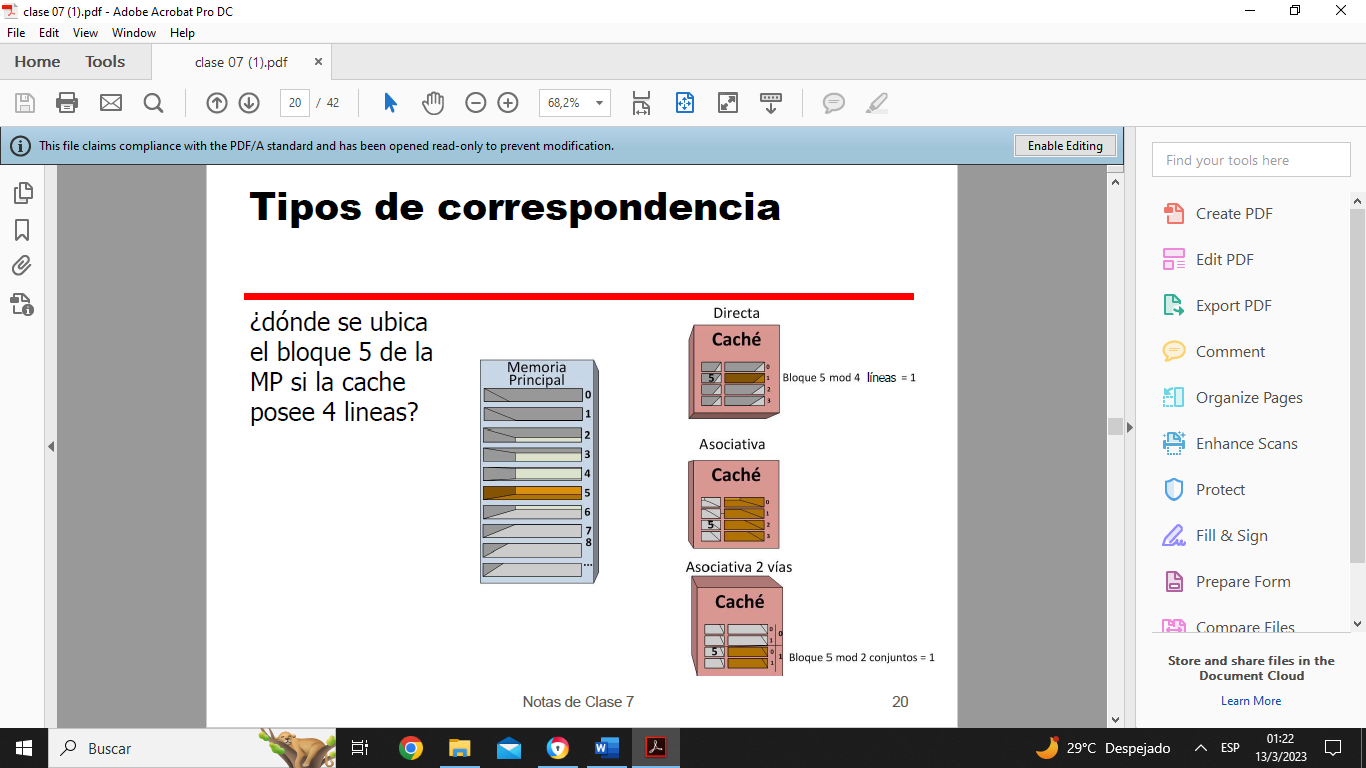
(transferencia de datos) Entonces la lectura de caché a memoria se realiza por bloques. Un bloque X contiene K palabras de memoria principal. La caché contiene C líneas de memoria donde cada una contiene un bloque de celdas de memoria principal, es decir su ancho es de K palabras.

Política de ubicación, tipo de función correspondencia

* Mapeo directo: es la técnica más sencilla, consiste en hacer corresponder cada bloque de la memoria principal a solo una línea posible de la cache. Es necesario etiquetar los datos para distinguirlos de otros bloques que pueden caber en esa línea.

La ventaja es que es muy sencilla de implementar. La desventaja es que hay una posición concreta de chache para cada bloque dado. Por ello, si un programa referencia repetidas veces a palabras de los bloques diferentes asignados en la misma línea dichos bloques se estarían intercambiando continuamente en la cache, y la tasa de aciertos seria baja.

* Mapeo asociativo: hay flexibilidad en cuanto a que bloque reemplazar cuando se lee un bloque en la caché. Soluciona el problema del mapeo directo permitiendo que cada bloque de memoria principal se cargue en cualquier línea de caché. La lógica de control de cache interpreta una dirección de memoria simplemente como una etiqueta y un campo de palabra. El campo etiqueta identifica explícitamente bloques de memoria y cada línea es examinada para una concordancia, desventaja búsqueda en cache es costosa.
* Mapeo asociativo de conjuntos: se toma lo mejor de los dos algoritmos anteriores, la cache se divide en un numero de conjuntos y cada conjunto contiene un numero de líneas. Un bloque dado mapea a cualquier línea del conjunto dado. No se busca la etiqueta por todas las líneas, sino solo en las líneas dentro del conjunto.



Política de reemplazo de bloque, algoritmo de sustitución

Cuando tengo correspondencia directa, no tengo optativa de elección de donde reemplazar, tengo que escribir el que ocupa el lugar del nuevo. Como ese dato corresponde a un único lugar de la cache no hay alternativa.

Cuando tengo correspondencia asociativa, en este si tengo opciones, como los datos se pueden guardar en cualquier parte, se reemplaza según alguna de estas estrategias:

LRU: menos recientemente usado, sobre el dato que entró y hace mucho no fue usado. FIFO: se escribe sobre el dato más antiguo. LFU: se escribe sobre el dato menos frecuentemente usado. Aleatorio: sobrescribo cualquier dato.

Política de escritura

En una estructura en la que más de un procesador tiene un a cache y la memoria principal es compartida, si se modifican los datos de una cache, se invalida no solamente la palabra correspondiente de memoria principal, sino también la misma palabra en otras caches. Un sistema que evite este problema se dice que mantiene la coherencia de cache. Entre las posibles aproximaciones a la coherencia de cache se incluyen:

 Vigilancia del bus con escritura inmediata: cada controlador de cache monitoriza las líneas de direcciones para detectar operaciones de escritura en memoria, por parte de otros maestros en el bus.

Si otro maestro escribe en una posición de memoria compartida que también reside en la memoria cache, el controlador de cache invalida el elemento de la cache. Esta estrategia depende del uso de una política de escritura inmediata por parte de todos los controladores de cache.

 Transparencia hardware: se utiliza hardware adicional para asegurar que todas las actualizaciones de memoria principal vía cache quedan reflejadas en todas las cache.

 Memoria excluida de cache: solo una porción de memoria principal se comparte con más de un procesador y esta se diseña como no transferible a cache. En un sistema de este tipo todos los accesos a la memoria compartida son fallos de cache porque la memoria compartida nunca se copia a cache.

Cache. Describir las políticas de escritura (en acierto y en fallo).

* Política de escritura en acierto (se encuentra en la cache el dato solicitado)
* Escritura inmediata: se actualizan simultáneamente la posición de cache y la de memoria principal para evitar inconsistencias.
* Post escritura: la información solo se actualiza en la caché, es un método más rápido. Se marca como actualizada con un bit de “sucio”. La memoria principal se actualiza en el reemplazo y puede pasar que contenga información errónea en algún momento.

* Política de escritura en fallo (cuando el dato solicitado no se encuentra en la caché)
* Write Allocate: se comporta de forma habitual, la información se lleva de la memoria principal a la caché y ahí se sobrescriben los datos en la caché.
* No-write Allocate: acá no nos preocupamos por la cache, el bloque no se lleva a la memoria caché, se escribe directamente en la memoria principal.

UNIDAD 5:

¿Qué características posee un procesador superescalar?

Un procesador superescalar es aquel que usa múltiples cauces de instrucciones independientes. Cada cauce consta de múltiples etapas, de modo que puede tratar varias instrucciones a la vez. El hecho de que haya varios cauces introduce un nuevo nivel de paralelismo, permitiendo que varios flujos de instrucciones se procesen simultáneamente. Saca provecho de lo que se conoce como “paralelismo al nivel de instrucciones”, que hace referencia al grado en que las instrucciones de un programa pueden ejecutarse en paralelo. Un procesador superescalar capta varias instrucciones a la vez, y a continuación, intenta encontrar instrucciones cercanas que sean independientes entre sí, y puedan, por consiguiente, ejecutarse en paralelo. Si la entrada de una instrucción depende de la salida de una instrucción anterior, la segunda instrucción no puede completar su ejecución al mismo tiempo ni antes que la primera, pero una vez que se han identificado tales dependencias, el procesador puede emitir y completar instrucciones en un orden diferente al del código de máquina original.

¿Qué características posee un procesador supersegmentado?

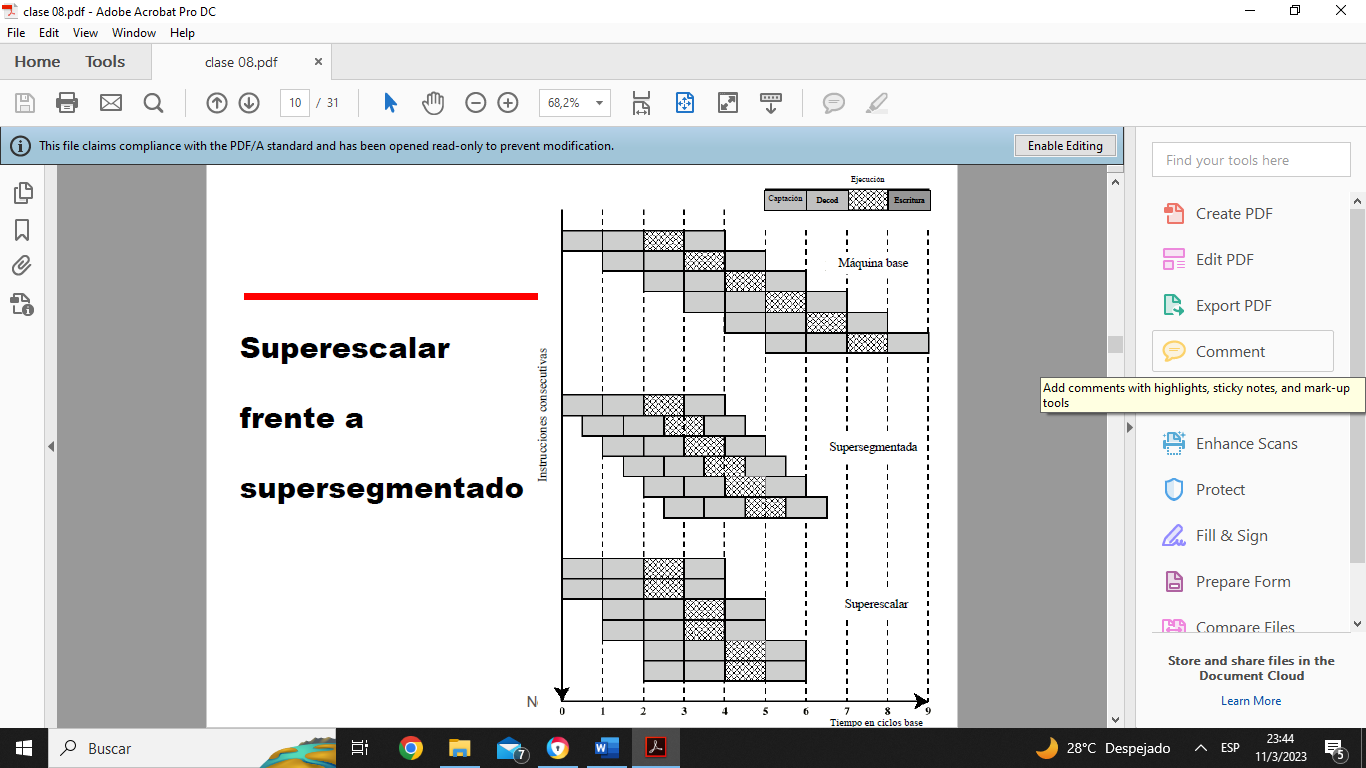
La supersegmentación aprovecha el hecho de que muchas etapas del cauce realizan tareas que requieren menos de medio ciclo de reloj. De este modo, doblando la velocidad de reloj interna se permite la realización de dos tareas en un ciclo de reloj. Obtendremos mayores prestaciones subdividiendo el ciclo de reloj en subintervalos. Resulta una mayor frecuencia del ciclo de reloj. División de las etapas “macro” del cauce segmentado en subetapas más pequeñas (y, por tanto, más rápidas) y se transmiten los datos a la mayor velocidad del ciclo de reloj. El tiempo para las instrucciones individuales no varía. Aumenta el grado del paralelismo. Incrementa la aceleración percibida.

¿Qué características posee un procesador supersegmentado frente a un superescalar?

La supersegmentacion aprovecha el hecho de que muchas etapas del cauce realizan tareas que requieren menos de medio ciclo de reloj. De este modo, doblando la velocidad del reloj interno, se permite la realización de dos tareas en un ciclo de reloj externo.

El superescalar, en cambio, tiene múltiples cauces de instrucciones independientes. Cada cauce consta de múltiples etapas, de modo que puede tratar varias instrucciones a la vez. El hecho de que haya varios cauces introduce un nuevo nivel de paralelismo, permitiendo que varios flujos de instrucciones se procesen simultáneamente. Un procesador superescalar saca provecho de lo que se conoce como paralelismo en las instrucciones, que hace referencia al grado en que las instrucciones de un programa pueden ejecutarse en paralelo.

Ambas técnicas, si trabajan ininterrumpidamente, pueden procesar la misma cantidad de instrucciones en el mismo tiempo. El procesador supersegmentado se queda atrás con respecto al procesador superescalar al comienzo del programa y en cada destino de un salto.



Clasificación de los sistemas paralelos según taxonomía de Flynn

Una secuencia de instrucciones y una secuencia de datos (SISD): un único procesador interpreta una única secuencia de instrucciones para operar con los datos almacenados en una única memoria. Los computadores monoprocesador caen dentro de esta categoría.

Una secuencia de instrucciones y múltiples secuencias de datos (SIMD): una única instrucción máquina controla paso a paso la ejecución simultánea y sincronizada de un cierto número de elementos de proceso. Cada elemento de proceso tiene una memoria asociada, de forma que cada instrucción es ejecutada por cada procesador con un conjunto de datos diferentes. Los procesadores vectoriales y los matriciales pertenecen a esta categoría.

Múltiples secuencias de instrucciones y una secuencia de datos (MISD): se transmite una secuencia de datos a un conjunto de procesadores, cada uno de los cuales ejecuta una secuencia de instrucciones diferente. Esta estructura nunca ha sido implementada.

Múltiples secuencias de instrucciones y múltiples secuencias de datos (MIMD): un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes. Se pueden dividir según la forma de comunicarse:

\* Memoria compartida: SMP. \*Memoria distribuida: clusters.

Características de un multiprocesador simétrico SMP

Computadora autónoma con las siguientes características:

Dos o más procesadores similares de capacidades comparables. Comparten la memoria principal y las E/S. Interconectados mediante un bus u otro tipo de sistema de interconexión. Tiempo de acceso a memoria similar para todos los procesadores (UMA). Todos los procesadores pueden desempeñar las mismas funciones. Sistema operativo integrado, que proporciona la interacción entre los procesadores y sus programas.

Ventajas potenciales de un SMP:

Mayores prestaciones, si el trabajo a realizar puede organizarse en paralelo. Buena disponibilidad, un fallo en un procesador no detendrá la computadora. Crecimiento incremental, se pueden añadir más procesadores. Escalado, en función de la cantidad de procesadores.

Cuidado, Bus compartido: La prestación está limitada por el tiempo de ciclo el bus. Cada procesador debería estar equipado con una memoria cache para mejorar las prestaciones, se reduciría el número de accesos. Se pueden producir problemas de coherencia de cache, este problema debe ser resuelto por el hardware con protocolos de sondeo y protocolos de directorio.

Características de clusters:

Computadoras completas interconectadas que trabajan conjuntamente como un único recurso, dan ilusión de que se trata de una única máquina. Cada computadora se denomina “nodo”. Prestaciones y disponibilidad elevadas. Aplicaciones propias de un servidor. Son la alternativa a los SMP.

Beneficios del cluster: Escalabilidad absoluta e incremental. Alta disponibilidad y mejor relación precio/ prestaciones.

