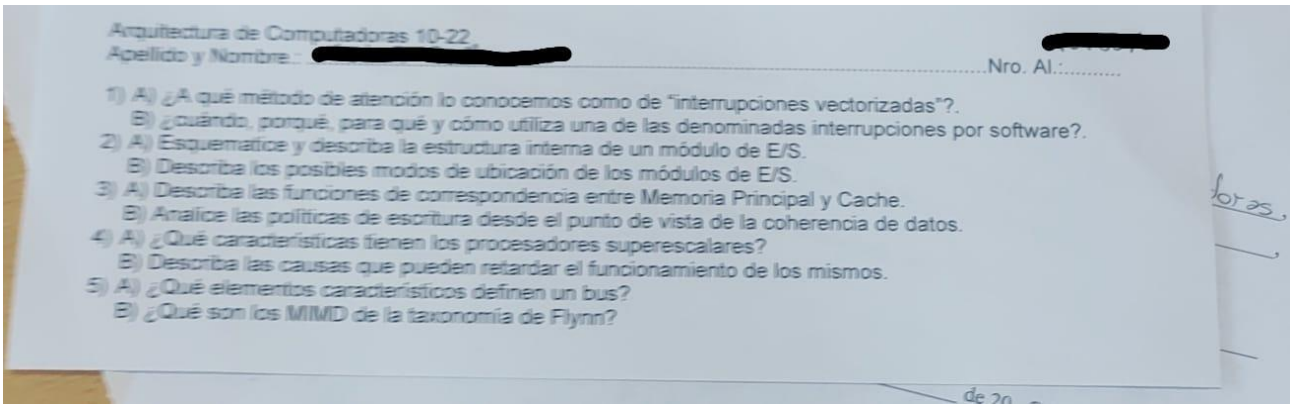
Resumen para el Final de Arquitectura de computadoras.

Temas:

1) **Interrupciones**
    a) definición
    b) ciclo de interrupción
    c) jerarquía de interrupciones
    d) int por hardware/software
    e) reconocimiento de interrupciones
    f) pic
2) **Módulo E/S**
    a) definición y estructura
    b) Funciones de un módulo de E/S (no creo que lo tomen)
    c) Tecnicas de gestion de E/S
    d) etapas y tipos de transferencia (no creo que lo tomen)
    e) Direccionamiento(modo de ubicación)?
3) **Segmentación de Instrucciones** (no vi que lo tomen muy seguido pero lo pueden tomar)
    a) definición
    b) tareas por ciclo
    c) atascos de cause
4) **RISC y CISC**
    a) características
    b) problemas en comparaciones
5) **Memoria caché**
    a) jerarquía de memoria
    b) diagrama
    c) definición
    d) diseño de la caché
    e) política de ubicación
    f) política de reemplazo
    g) política de escritura
6) **Buses**
    a) definición
    b) estructura
    c) tipos de buses
    d) métodos de arbitraje
    e) temporización
    f) PCI y SCSI
7) **Procesadores superescalares**
    a) definición
    b) DIAGRAMA
    c) paralelismo
    d) superescalar frente a supersegmentado (no creo que lo tomen)
    e) Políticas de emisión de instrucciones (no creo que lo tomen)
    f) Taxonomía de Flynn
8) **Otras definiciones varias**

Aclaración: Resumen hecho en Google docs por Juan Montalivet con contenido de pdfs de teoría, resúmenes de años anteriores y conclusiones propias.
No tenía pensado compartir este resumen y por eso está un poco (bastante) desprolijo.
Use arial 12 para la información más importante de recordar y arial 9 para información "no tan importante" pero que agrega detalles, si ven algo con otra tipografía por ej calibri 11 es porque lo copié de algún pdf u otro resumen.
Examen aprobado en mesa de octubre 2022 con nota 6 (2 de 10 preguntas sin responder por falta de tiempo 2hs) utilizando este resumen, suerte.


Arquitectura de Computadoras 10-22
Apellido y Nombre: [redacted] Nro. Al.:
1) A) ¿A qué método de atención lo conocemos como de "interrupciones vectorizadas"?.
B) ¿cuándo, porqué, para qué y cómo utiliza una de las denominadas interrupciones por software?.
2) A) Esquematice y describa la estructura interna de un módulo de E/S.
B) Describa los posibles modos de ubicación de los módulos de E/S.
3) A) Describa las funciones de correspondencia entre Memoria Principal y Cache.
B) Analice las políticas de escritura desde el punto de vista de la coherencia de datos.
4) A) ¿Qué características tienen los procesadores superescalares?
B) Describa las causas que pueden retardar el funcionamiento de los mismos.
5) A) ¿Qué elementos característicos definen un bus?
B) ¿Qué son los MIMD de la taxonomía de Flynn?
horas.
de 20

1
Interrupciones:
Mecanismo mediante el cual se puede interrumpir el procesamiento normal de la CPU. Son eventos que indican que existe una condición en algún lugar del sistema, que requiere la atención del procesador.

¿Qué se hace si se genera una interrupción? (el pic hace lo mismo)
Generalmente, para interrumpir la CPU, se transfiere el control del flujo de ejecución a un gestor/manejador de interrupciones que:
1- Suspende la ejecución del programa en curso.
2- Guarda el contexto actual, próxima instrucción y estado del procesador.
3- Carga el PC con la dirección de comienzo de una rutina de gestión de interrupción, se inhiben otras interrupciones.
4- Finalizada la rutina de gestión, restaura el estado original del procesador.
5- Retorna la ejecución normal del programa interrumpido.

Jerarquía de interrupciones:
-Enmascarables: pueden ser ignoradas mediante instrucciones que inhiben la posible solicitud.
-No Enmascarables: NO pueden ignorarse. eventos de alta prioridad.

Int por hardware:
Son generadas por disp de E/S. El sistema de cómputo tiene que manejar estos eventos "no planeados" o "asincrónicos" que no están relacionados con el proceso en ejecución en ese momento.

Traps/excepciones: Interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos (falla de prog, de hard, etc).

int por software: Son generalmente usadas para hacer llamadas a funciones del SO, esto es posible porque muchos procesadores tienen instrucciones explícitas que afectan al procesador de la misma manera que las interrupciones por hardware.

Mecanismo/Ciclo de interrupción
-Se comprueba si se ha solicitado alguna interrupción. (indicada por flag de pedido de interrupción.)
-Si no hay señal se capta la siguiente instrucción.
-Si hay algún pedido de interrupción pendiente:
1- Suspende la ejecución del programa en curso.
2- Guarda el contexto actual, próxima instrucción y estado del procesador.
3- Carga el PC con la dirección de comienzo de una rutina de gestión de interrupción, se inhiben otras interrupciones.
4- Finalizada la rutina de gestión, restaura el estado original del procesador.
5- Retorna la ejecución normal del programa interrumpido.

Conclusión:
Las interrupciones proporcionan una forma de mejorar la eficiencia del procesador. Los dispositivos externos suelen ser mucho más lentos que el procesador y esperar por las instrucciones de E/S sería un desperdicio de procesador. Con las interrupciones, el procesador puede dedicarse a ejecutar otras instrucciones mientras una operación de E/S está en curso.
La operación de E/S se realiza concurrentemente con Ía ejecución de instrucciones del programa de usuario. Cuando el dispositivo externo está listo para aceptar más datos del procesador, el módulo de E/S de este dispositivo externo envía una petición de interrupción al procesador. Éste suspende la operación del programa que estaba ejecutando y salta a un programa llamado “gestor de interrupción” que da servicio al dispositivo en cuestión y luego continúa con el programa original.

Reconocimiento de Interrupciones

Interrupciones multinivel

- Cada dispositivo que puede provocar interrupción tiene una entrada física de interrupción conectada a la CPU.
- Es muy sencillo, pero muy caro.

Línea de interrupción única

- Una sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos.
- Se debe “preguntar” a cada dispositivo si ha producido el pedido de interrupción (técnica Polling/encuesta)

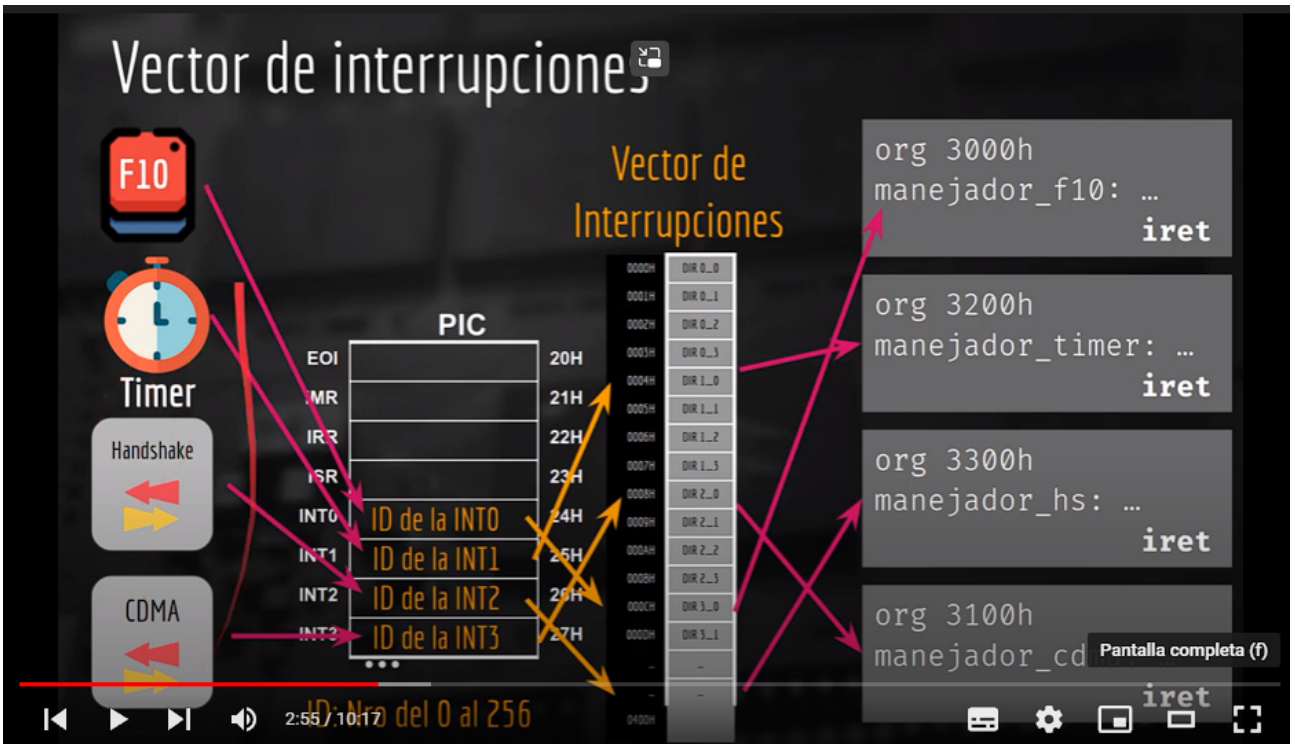
Interrupciones vectorizadas

vector de interrupciones definición: sector de la memoria (que se encuentra al principio de la memoria, por ejemplo los primeros 1024 bytes, de 0000h hasta 0400h) donde se guardan las DIRECCIONES de las subrutinas que atienden/manejan a las interrupciones.

Cuando se produce una interrupción “x” (timer por ejemplo) se busca, en el registro INT del PIC que corresponde a la interrupción “x” (INT0 … INT7), el ID de interrupción. Este id de interrupción se multiplica por 4 y da como resultado la dirección del vector de interrupciones en la cual se encuentra la dirección de la subrutina que atiende/maneja la interrupción producida.

dato: (los ids 0, 6, 7 no se pueden usar porque en esos lugares del vector de interrupciones se encuentran las direcciones de las subrutinas que atienden las int por software).

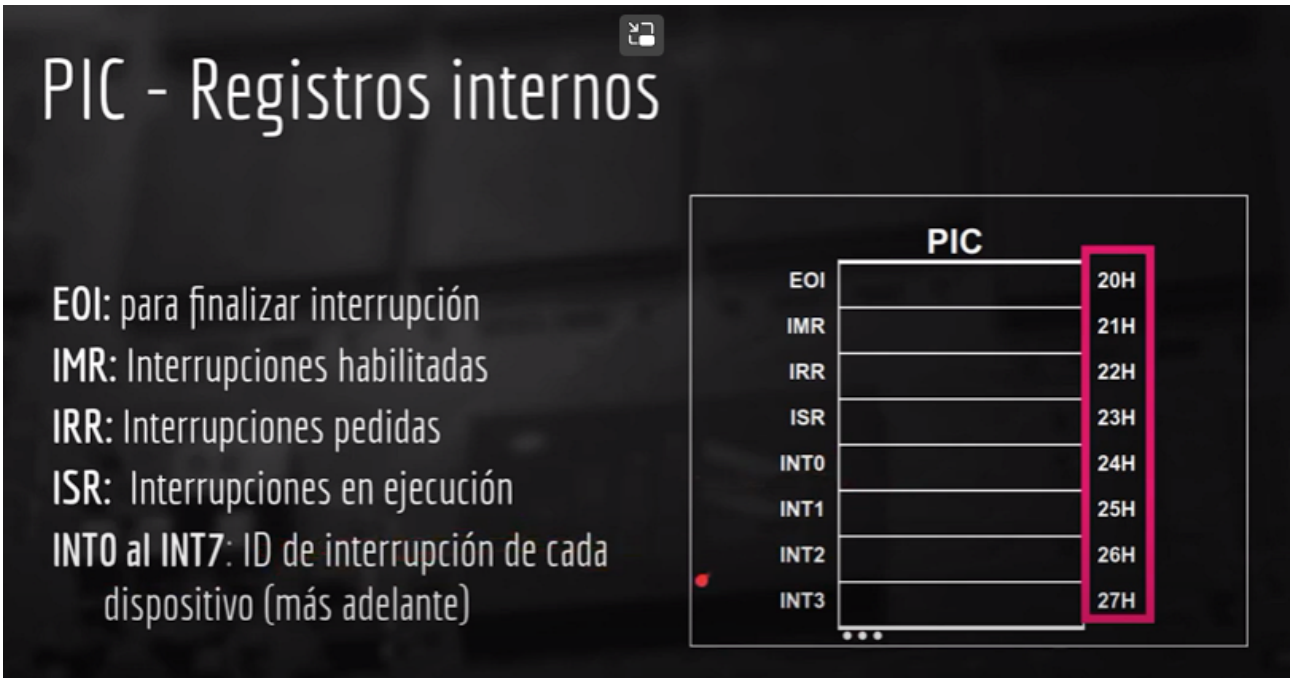
2.7 Vector de Interrupciones - YouTube

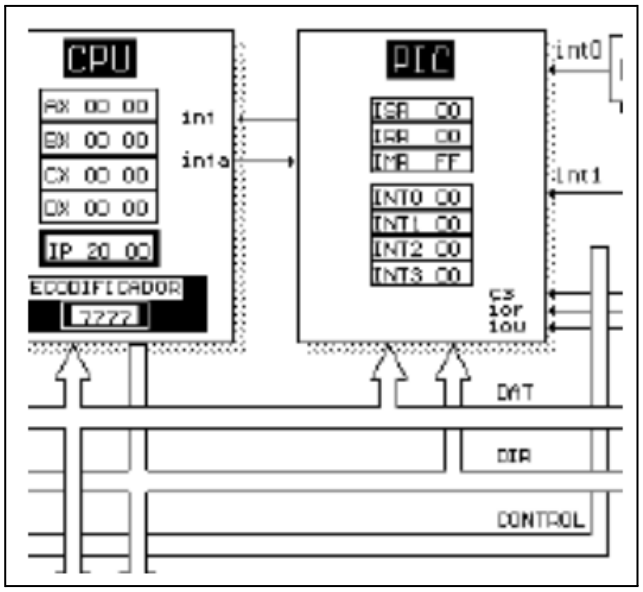


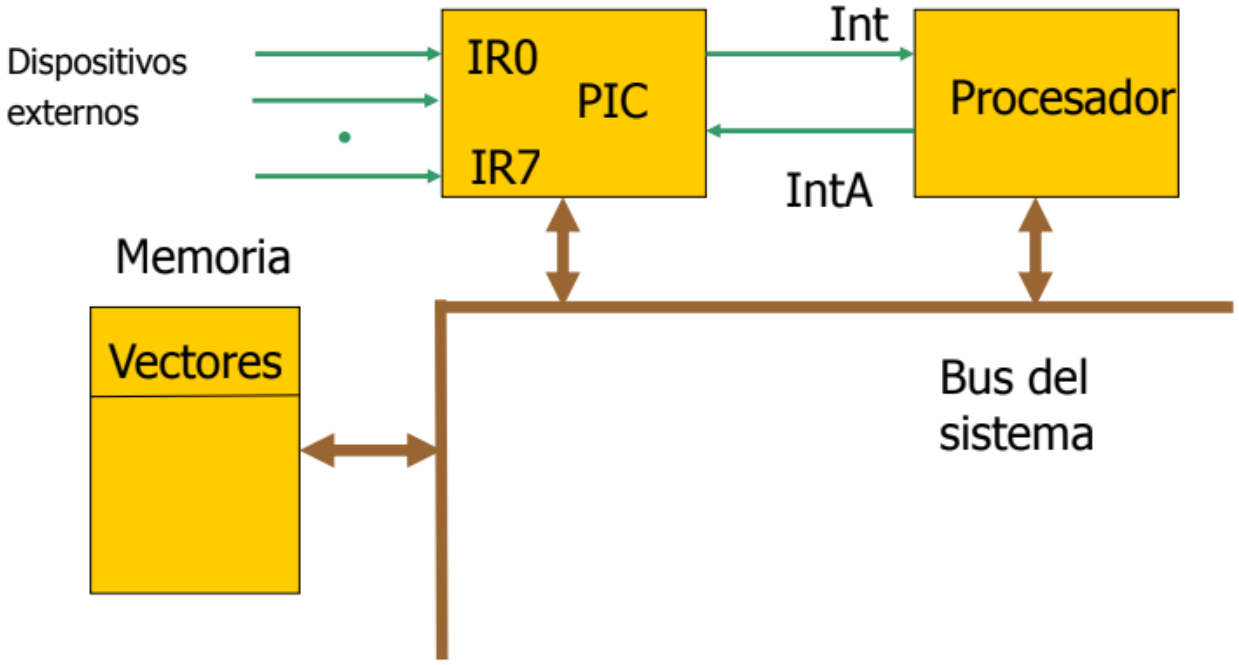
Controlador de interrupciones (PIC)
Es un dispositivo usado para combinar varias fuentes de interrupciones sobre una o más líneas del CPU.
puede manejar hasta 8 peticiones de interrupción independientes al mismo tiempo, numeradas de la 0 (INT0) a la 7 (INT7) Si más de una petición de interrupción se producen exactamente al mismo tiempo entonces el PIC las “pasa” a la CPU en un orden de prioridad, donde la petición por la entrada 0 tiene la prioridad más alta y la de la 7 la menor.

Estructura interna: (Registros internos)
-IRR (Interrupt Request Register): Registro de petición de interrupción, indica con bit en 1 las interrupciones demandadas hasta el momento.
-ISR (In Service Register): Registro de interrupción en servicio, indica con bit en 1 cuál es la interrupción que está siendo atendida.
-IMR (Interrupt Mask Register): Registro de maścara de interrupciones, permite el enmascaramiento selectivo de cada una de las entradas de interrupción, indicando con bit en 1 (Indica cuáles deben ser ignoradas). Tras un reset los bits de este registro quedarán en 0.
-EOI (End of Interruption): Fin de interrupción. Como consecuencia, se pone en 0 el bit del ISR correspondiente. sirve para indicarle al pic que ya fue atendida la interrupción y la cpu vuelve a la ejecución normal del programa
-INT0 … INT7: 8 registros, donde se cargan los id/identificadores de int.

8 registros, donde se cargan los id/identificadores de int, ids que indican la dirección del vector de interrupciones en donde se encuentra la dirección de inicio de subrutina manejadora de cada interrupción.



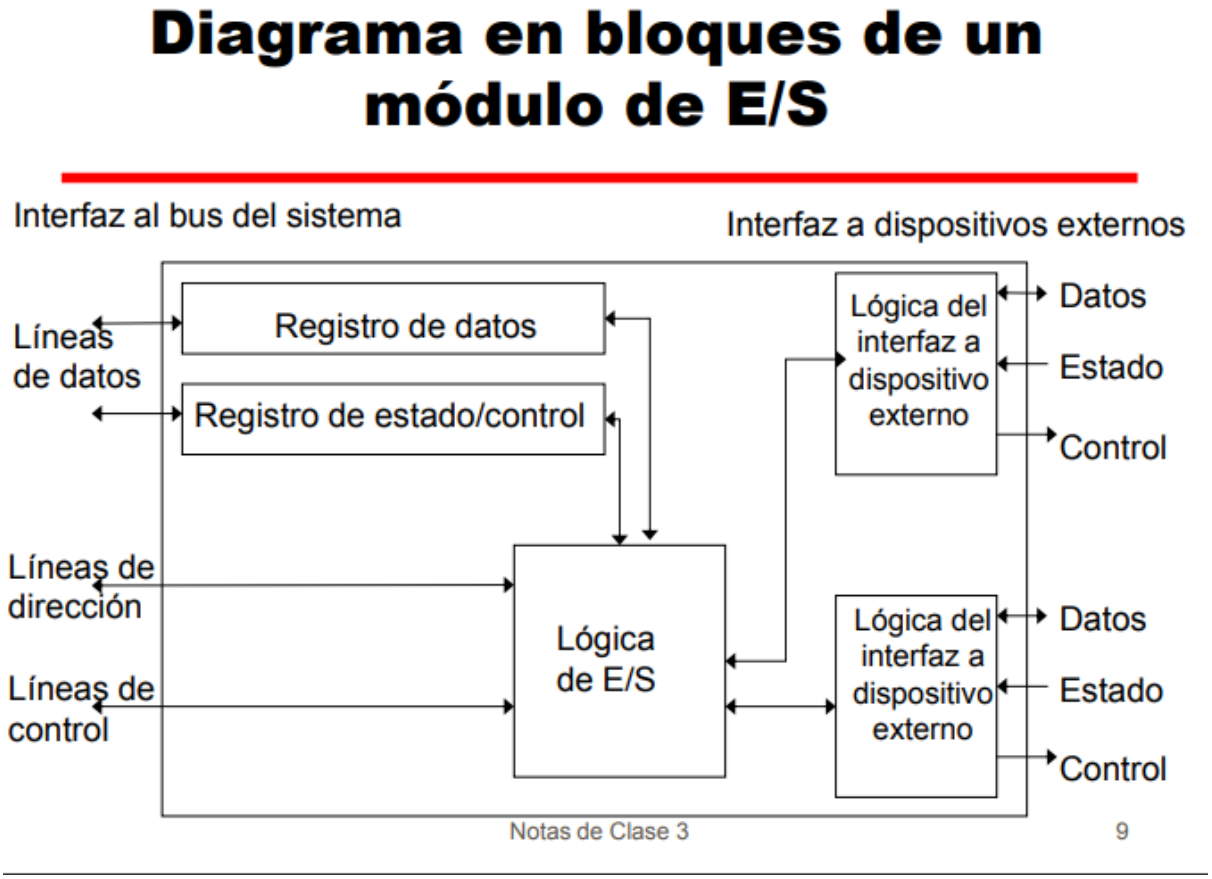




si queda alguna duda: [2.6 PIC - Dispositivo Interno - YouTube]

2
Módulo de E/S

Realiza la interfaz entre el procesador, la memoria (bus) y los periféricos.
El funcionamiento de un módulo de E/S permite que el procesador vea una amplia gama de dispositivos de una forma simplificada y oculta los detalles de temporización, formatos y electromecánica propios de los dispositivos para que el procesador pueda funcionar únicamente con órdenes de lectura y escritura / abrir y cerrar fichero.



(Explicación del diagrama:)
Estructura de un módulo de E/S El módulo se conecta al resto del computador a través de un conjunto de líneas (por ej, líneas del bus del sistema). Los datos que se transfieren a y desde el módulo se almacenan temporalmente en uno o más registros de datos. Además, puede haber uno o más registros de estado que proporcionan información del estado presente. Un registro de estado también puede funcionar como un registro de control, para recibir la información de control del procesador. La lógica que hay en el módulo interactúa con el procesador a través de una serie de lineas de control. Éstas son las que utiliza el procesador para proporcionar órdenes al módulo de E/S. El módulo también debe ser capaz de reconocer y generar las direcciones asociadas a los dispositivos que controla. Cada módulo de E/S tiene una dirección única o, si controla más de un dispositivo externo, un conjunto único de direcciones. Por último, el módulo de E/S posee la lógica específica para la interfaz con cada uno de los dispositivos que controla.

Funciones de un módulo de E/S: (esto me parece que no lo piden**)
1 Control y temporización de uno o más dispositivos externos
2 Interpretar las órdenes que recibe de CPU y transmitirlas al periférico
3 Comunicación con la CPU (registros) y Memoria
4 Controlar las transferencias de datos entre CPU y el periférico (convertir formatos, adaptar velocidades)
5 Comunicación con los dispositivos (periféricos)
6 Informar a la CPU del estado del periférico
7 Almacenamiento temporal (buffering) de datos

8 Detección de errores

yo lo resumiria en estos**:
-Interpretación y transmisión de órdenes que recibe de la CPU al periférico (realiza la conversión de formatos y velocidad).
-Control de dispositivos externos.
-Comunicación con CPU, Memoria y periféricos.
-Almacenamiento temporal de datos (buffering).
-Detección de errores.

Tecnicas de gestion de E/S: descripciones
E/S Programada con espera de respuesta: Cuando el procesador está ejecutando un programa y encuentra una instrucción relacionada con una E/S, ejecuta dicha instrucción mandando una orden al módulo de E/S. El módulo realiza la acción solicitada y activa los bits apropiados en el registro de estado pero no interrumpe al procesador, sino que éste es responsable de comprobar periódicamente el estado del módulo de E/S hasta que encuentre que la operación ha terminado (CPU espera ociosa, algo no deseable).
Los datos se intercambian entre la CPU y el módulo de E/S. La CPU tiene control directo sobre la operación de E/S: Comprobación de estado del dispositivo, el envío de orden de lectura o escritura y la transferencia del dato.

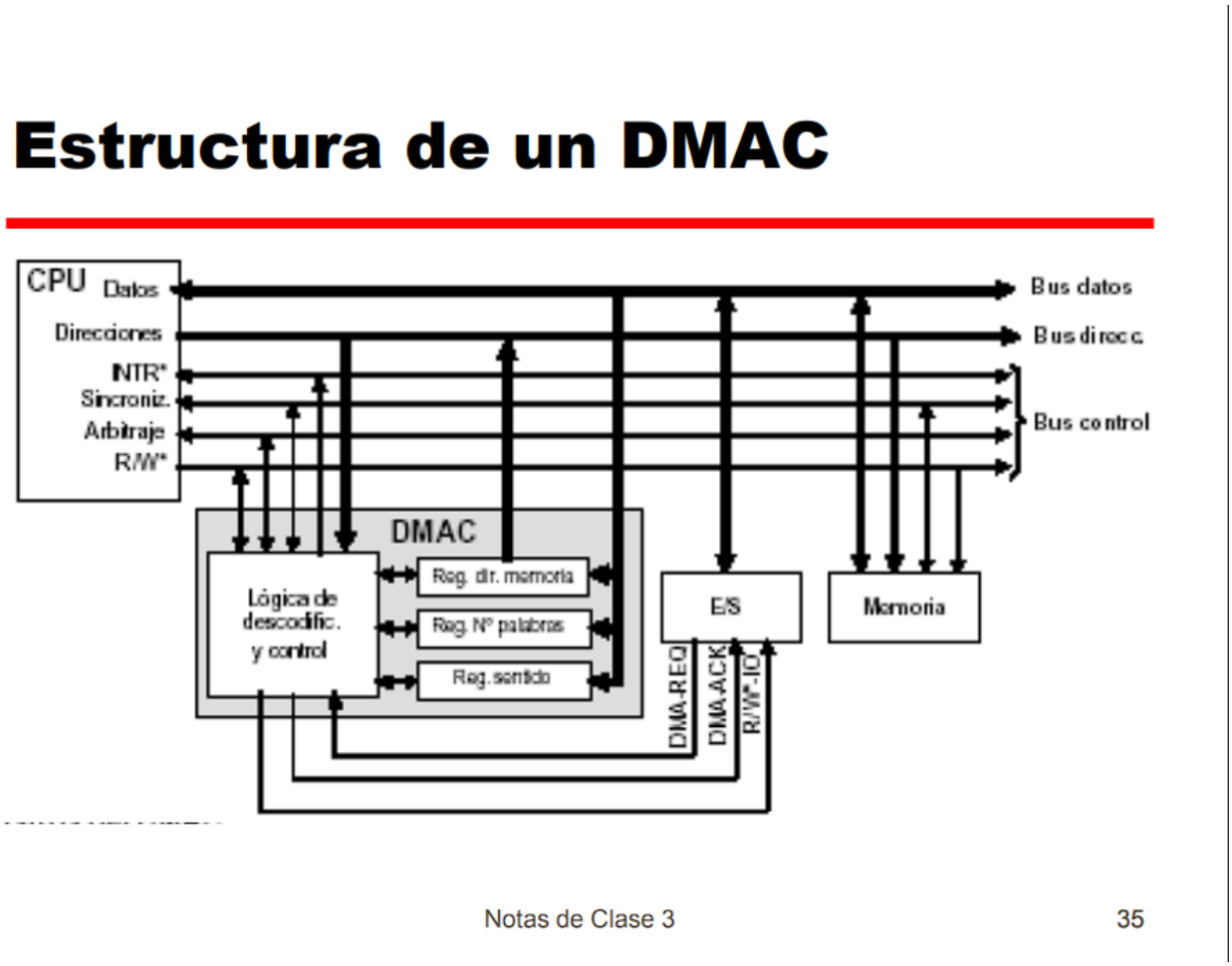
E/S con interrupciones: a diferencia de la e/s programada, en esta técnica, El módulo e/s envía un pedido de interrupción a la CPU cuando está listo, La CPU no tiene que esperar la finalización de la tarea de E/S, puede seguir procesando.
El procesador ejecuta una orden de E/S, puede continuar ejecutando otras instrucciones y es interrumpido por el módulo de E/S cuando éste ha terminado su trabajo. Ahí el procesador ejecuta la transferencia de datos y continúa con el procesamiento previo.

Las operaciones de E/S mediante interrupciones son más “efectivas” que las programadas. Pero ambas necesitan la intervención directa de la CPU, la cual permanece ocupada mucho tiempo durante la operación, y es responsable de extraer los datos de la memoria principal en una salida y de almacenar los datos en memoria principal en una entrada

E/S con acceso directo a memoria (DMA):
Con la tecnica DMA los periféricos y la memoria principal intercambian datos directamente, sin la intervención del procesador, aumentando el rendimiento de aplicaciones. Permite a dispositivos de diferentes velocidades comunicarse sin someter a la CPU a una carga masiva de interrupciones. Si el volumen a transferir es grande, esta técnica es la más eficiente.

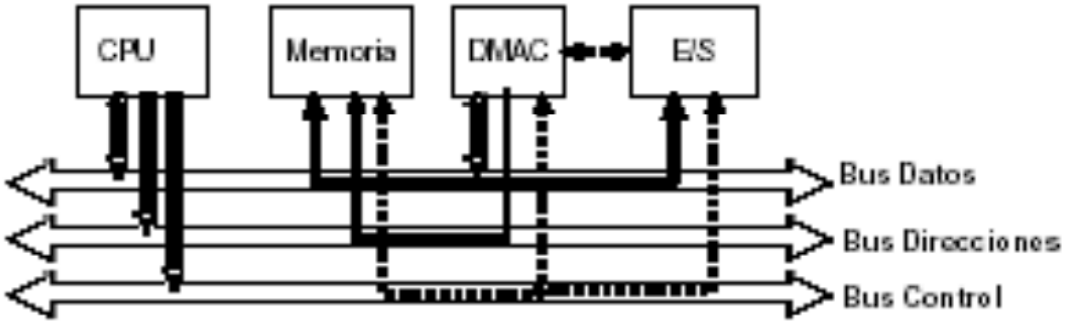
DMA requiere un módulo/controlador en el bus del sistema: El controlador de DMA (DMAC).
El DMAC actúa como maestro de bus para transferir datos a y desde memoria a través del bus del sistema. Para hacerlo, debe ser capaz de solicitar el uso del bus mediante la lógica de arbitraje y señales de control necesarias ya que la CPU liberará el bus cuando no lo necesite o si es forzado por el DMAC (robo de ciclo o cycle stealing).

Por dentro El controlador de DMA (DMAC) esta compuesta por el controlador de datos, registro de datos(numero de palabras), registro de direcciones y la lógica de decodificación y control(que registra el sentido de la transferencia, sea lectura o escritura)



Acceso directo a memoria (DMA)

El controlador de DMA es un dispositivo capaz de controlar una transferencia de datos entre un periférico y memoria sin intervención de la CPU.



Notas de Clase 3 33

(para la estructura del dma podria hacer el segundo dibujo pero con los componentes internos del dmac del primer dibujo) (de dma se podria resumir tambien: funcionamiento, tipos de transferencia problemas que puede haber con dma (coherencia de datos))

Etapas de transferencia DMA

INICIALIZACIÓN DE LA TRANSFERENCIA: La CPU debe enviar a la interfaz del periférico y al DMAC los parámetros de la transferencia. Después de la inicialización la CPU retorna a sus tareas y ya no se preocupa más de la evolución de la transferencia.

REALIZACIÓN DE LA TRANSFERENCIA: Cuando el periférico está listo para realizar la transferencia se lo indica al DMAC, este pide control del bus y se realiza la transferencia entre el periférico y la memoria. Después de cada transferencia se actualizan los registros DMAC (número de bytes a transferir y dirección de memoria).

FINALIZACIÓN DE LA TRANSFERENCIA: El DMAC libera el bus y devuelve el control a la CPU. El DMAC suele activar una señal de interrupción para indicar a la CPU la finalización de las operaciones de E/S solicitada.

Tipos de transferencia

Si el DMAC toma el control del bus durante los intervalos de tiempo en los que la CPU no hace uso del mismo, el rendimiento del sistema no sufrirá degradación alguna.

DMA modo ráfaga (burst): el DMAC solicita el control del bus a la CPU. Cuando la CPU concede el bus el DMAC no lo libera hasta haber finalizado la transferencia del bloque de datos completo.

- Ventaja: transferencia rápida.
- Desventaja: puede degradar el rendimiento del sistema.

DMA modo robo de ciclo (cycle-stealling): el DMAC solicita el control del bus a la CPU. Cuando la CPU concede el bus al DMAC se realiza la transferencia de una palabra y luego el DMAC libera el bus. El DMAC solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo.

- Ventaja: No se degrada el rendimiento del sistema.
- Desventaja: la transferencia tarda más tiempo.

• Si bien el trabajo de la CPU es lento, no será tanto como si ella realizara la transferencia. Por lo tanto, para transferencia de E/S de múltiples palabras, es la técnica más eficiente.

Direccionamiento

(PREGUNTAR SI DIRECCIONAMIENTO ES LO MISMO QUE “MODO DE UBICACION”**)
Las Operaciones de Entrada ó Salida, requieren Direccionamiento y Pueden ser de dos tipos, Tanto las E/S mapeada en memoria y aislada se usan comúnmente.

E/S Mapeada en memoria: Se necesita solo una línea de lectura y solo una línea de escritura en el bus. Existe un único espacio de direcciones para las posiciones de memoria y los dispositivos de E/S.
E/S Aislada: Puesto que el espacio de direcciones de E/S está aislado del de memoria, este se conoce como E/S aislada. Los puertos de E/S solo son accesibles mediante una orden especifica de E/S, que activa las ordenes de E/S del bus. La ventaja es que se puede utilizar un amplio repertorio de instrucciones permitiendo una programación más eficiente.

3
Segmentación de Instrucciones

La segmentación de cauce (pipelining) es una forma de organizar el hardware de la CPU para realizar más de una operación al mismo tiempo, consiste en descomponer el proceso de ejecución de las instrucciones en etapas que permitan una ejecución simultánea.
Las instrucciones pasan por varios estados en el proceso de ejecución, por lo tanto, varias instrucciones pueden ser procesadas simultáneamente (cada una en estados distintos).

Características

La segmentación es una técnica de mejora de prestaciones a nivel de diseño de hardware. Es invisible al programador. El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones (RISC). Incrementa la productividad, pero no reduce el tiempo de ejecución de la instrucción.

Tareas a realizar por ciclo

- busqueda. f
- decodificacion. d
- ejecucion. x
- acceso a memoria. m
- almacenamiento. w

(se podria agregar mas info pero no veo que pregunten mucho por las tareas por ciclo **)

Atascos de un cauce (stall)

Situaciones que impiden a la siguiente instrucción que se ejecute en el ciclo correspondiente.

- Estructurales: Provocados por conflictos por los recursos (dos o más instrucciones necesitan usar el mismo recurso de HW en el mismo ciclo)
- Por dependencia de datos: Dos instrucciones se comunican por medio de un dato que no está disponible cuando se necesita en una etapa determinada del cauce.
- Por dependencia de control: La ejecución de una instrucción depende de cómo se ejecute otra (ej.: un salto, varios posibles caminos) Existe una penalización por salto.

Si resolvemos con paradas del cauce(o detención de cause?), se disminuye el rendimiento.

Soluciones a riesgos estructurales
- Duplicación de recursos hardware
- Separación en memorias de instrucciones y datos
- Turnar el acceso al banco de registros

Soluciones a riesgos de datos

Tipos de dependencias de datos (lo tomaran?)**

- LECTURA DESPUÉS DE ESCRITURA (RAW, dependencia verdadera) una instrucción genera un dato que lee otra posterior.
- ESCRITURA DESPUÉS DE ESCRITURA (WAW, dependencia en salida) una instrucción escribe un dato después que otra posterior, esto solo se da si se deja que las instrucciones se adelanten unas a las otras.
- ESCRITURA DESPUÉS DE LECTURA (WAR, Antidependencia) una instrucción modifica un valor antes de que otra anterior que lo tiene que leer, lo lea. Esto, no se puede dar en nuestro cauce simple.

Soluciones para riesgos RAW: Se debe determinar cómo y cuándo aparecen esos riesgos. Será necesaria una unidad de detección de riesgos y/o compilador más complejo.

-A nivel de Hardware: adelantamiento de operandos (forwarding) consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando sin esperar a la escritura.
-A nivel Software: instrucciones NOP o reordenación de código.

Soluciones a riesgos de control:
Tratamiento de saltos, soluciones/aproximaciones a instrucciones de bifurcación condicional, las cuales, Hasta que la instrucción no se ejecuta realmente, es imposible determinar si el salto se producirá o no.
- flujos múltiples
- pre captar el destino del salto
- buffer de bucles
- Predicción de saltos (para evitar la parada): técnica hardware
- Salto retardado (o de relleno de ranura de retardo): técnica software

Cuando hay una bifurcación o salto, hasta que no se ejecuta la instrucción no hay forma de saber qué instrucción vendrá a continuación. Por lo general se carga la siguiente instrucción y continúa. Si el salto no se produce se obtiene el máximo provecho en rendimiento. Si el salto se

produce, se presenta una penalización en las prestaciones sufrida por no haber podido predecir el salto.
Las instrucciones de salto pueden ser:

- **Incondicional**: La dirección de destino se debe determinar lo más pronto posible, dentro del cauce, para reducir la penalización.
- **Condicional**: Introduce riesgo adicional por la dependencia entre la condición de salto y el resultado de una instrucción previa

Tratamiento: modificación sencilla de la ruta de datos para reducir la cantidad de paradas a un solo ciclo. Adelantar la resolución de los saltos a la etapa D (decodificación):

- En ella se decodifica y se sabe que es un salto
- Se puede evaluar la condición de salto (con restador)
- Se puede calcular la dirección de salto (con sumador)

(yo me acordaria solo eso, ya acordarse cada una de las técnicas me parece una lo cu ra**)

Técnicas para el tratamiento de saltos condicionales:

- Flujos múltiples: Duplicar las partes iniciales del cauce y dejar que éste capte las dos instrucciones utilizando los dos caminos.
- Precaptar el destino del salto: Se precapta la instrucción del salto además de la siguiente a la bifurcación. Se guarda esta instrucción hasta que se ejecute la instrucción de bifurcación. Si se produce el salto, el destino ya habrá sido precaptado.
- Buffer de bucles: es una memoria pequeña de gran velocidad, gestionada por la etapa de captación de instrucción del cauce, que contiene, secuencialmente, las n instrucciones captadas más recientemente. Si se va a producir un salto, el hardware comprueba si el destino del salto está en el buffer.
- Predicción de saltos (para evitar la parada): técnica hardware
    - Técnicas estáticas: no dependen de la historia de la ejecución
        - Predecir que nunca se salta: Asume que el salto no se producirá y siempre capta la siguiente instrucción
        - Predecir que siempre se salta: Asume que el salto se producirá y siempre capta la instrucción destino del salto
        - Predecir según el código de operación: El procesador asume que el salto se producirá para ciertos códigos de operación de bifurcación y no para otros
    - Técnicas dinámicas: dependen de la historia de la ejecución, intentan mejorar la exactitud de predicción
        - Conmutador saltar/no saltar: Basado en la historia de las instrucciones, eficaz para los bucles
        - Tabla de historia de saltos (branch-target buffer): Pequeña caché asociada a la etapa de captación de instrucción del cauce. Cada elemento de la tabla consta de tres campos:
            - Dirección de una instrucción de bifurcación
            - Información de la instrucción destino (dirección del destino o instrucción destino)
            - Número de bits de estado (historia de uso de la instrucción)
- Salto retardado (o de relleno de ranura de retardo): técnica software
    - Se pueden mejorar las prestaciones de un cauce reordenando automáticamente las instrucciones de un programa, de forma que las instrucciones de salto tengan lugar después de lo realmente deseado (el compilador introduce instrucciones que se ejecutarán en cualquier caso después de la instrucción de salto).

RISC (Reduced Instruction Set Computer): Repertorio reducido de instrucciones

Estudios del comportamiento de la ejecución de los programas escritos en lenguajes de alto nivel, proporcionaron la orientación para diseñar el computador de repertorio reducido de instrucciones (reduced instruction set computer, RISC).

Características:

1) un repertorio de instrucciones limitado y sencillo.
2) un número grande de registros de uso general, o la utilización de un compilador que optimice el uso de éstos reduciendo las referencias a memoria para buscar operandos.
3) énfasis en la optimización de la segmentación de instrucciones.

RISC permite una segmentación eficiente porque hay menos operaciones llevadas a cabo por instrucción y estas son más previsibles. También se presta a la técnica de salto retardado, en la cual las instrucciones de salto se reubican entre otras instrucciones para mejorar la eficiencia del cauce.

CISC Computador de repertorio complejo de instrucciones.

Características:

1) Repertorio de instrucción amplio, con un gran número de instrucciones complejas.
2) Permite operaciones complejas entre operandos situados en la memoria o en registros internos.

La tendencia CISC ha sido motivada principalmente por dos razones:

• Simplificar los compiladores: el trabajo del escritor de compiladores es generar una secuencia de instrucciones máquina para cada sentencia de lenguaje de alto nivel (HLL). Si existen instrucciones máquina que se parezcan a sentencias del HLL, la tarea se simplifica. Sin embargo las instrucciones máquina complejas son con frecuencia difíciles de aprovechar.

• Mejorar las prestaciones: se espera que un CISC produzca programas más pequeños y rápidos que ocupen menos memoria, así que se ahorra este recurso, pero la memoria hoy en día es relativamente barata por lo que no es una gran ventaja.

Por lo tanto, no está nada claro que la tendencia hacia repertorios de instrucciones de complejidad creciente sea apropiada.

No existe una clara barrera diferenciadora.

Al momento de comparar computadores risc y cisc existen problemas en las comparaciones por ej:

-No existe un par de máquinas RISC y CISC directamente comparables.
-Es difícil separar los efectos del hardware de los del compilador.
-La mayoría de las máquinas son una mezcla de risc y cisc.

Los diseños RISC pueden sacar provecho de la inclusión de algunas características CISC y los diseños CISC pueden sacar provecho de la inclusión de algunas características RISC. El resultado es que **no existe una clara barrera diferenciadora.**

RISC vs. CISC

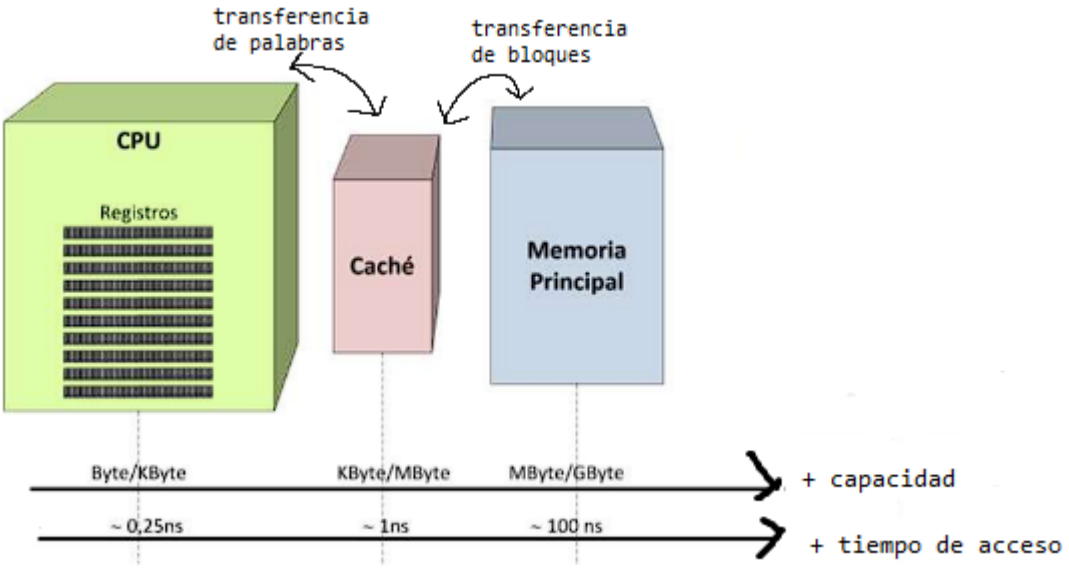
CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

no se de donde salio ese cuadro comparativo**

5
Memoria cache

Jerarquía de memoria:
Los programadores desean acceder a cantidades ilimitadas de memoria rápida !! Solución práctica: Jerarquía de memoria

La jerarquía de memoria usa diferentes tipos de memoria y de diferentes velocidades, está organizada en niveles que son ubicados en distintos lugares físicos, fabricados con tecnologías diferentes que se gestionan de manera independiente.
con el objetivo de que la velocidad del sistema sea, aproximadamente, la del nivel más rápido al costo del nivel más barato.
A medida que nos alejamos de la CPU, cada nivel inferior es más grande, más lento y más barato que el nivel previo (o superior) en la jerarquía.



ejemplo:
Teniendo una memoria principal relativamente grande y más lenta, junto con una memoria cache más pequeña y rápida se logra que la velocidad de la memoria sea lo más rápida posible, consiguiendo al mismo tiempo un tamaño grande.

¿Por qué funciona la jerarquía?: localidad de referencias

•Principio de Localidad Temporal: los elementos de memoria referenciados recientemente (datos o instrucciones), volverán a serlo en un futuro próximo => subo la palabra de nivel
•Principio de Localidad Espacial: los elementos de memoria cuyas direcciones están próximas a los últimos referenciados serán referenciados. => subo un bloque (con la palabra) de nivel

La caché es una memoria que dentro de una jerarquía de memoria generalmente se ubica entre la memoria principal y la CPU y contiene una copia de partes de la memoria principal. Cuando el procesador intenta leer una palabra de memoria, se hace una comprobación para determinar si la palabra está en la cache. Si es así, se entrega dicha palabra al procesador. Si no, un bloque de memoria principal, consistente en un cierto número de palabras, se transfiere a la caché y, después, la palabra es entregada al procesador. Debido al fenómeno de localidad de las referencias, cuando un bloque de datos es captado por la caché para satisfacer una referencia a memoria simple, es probable que se hagan referencias futuras a otras palabras del mismo bloque.

Diseño de la caché, elementos a tener en cuenta:
-Organización (tamaño y cantidad, niveles y costos) [no encontre ninguna explicación sobre este elemento y en los vídeos de teoría de runco no lo explica!]**
-Política de ubicación: Tipo de función de correspondencia
-Política de reemplazo: Algoritmo de sustitución
-Política de escritura.

-Política de ubicación:
•Función de correspondencia: Como hay menos líneas de caché que bloques de memoria principal, se necesita un algoritmo que haga corresponder bloques de memoria principal a líneas de caché. se necesita algún algoritmo que haga corresponder bloques de memoria principal a líneas de caché, Pueden usarse 3 técnicas:

1-Correspondencia directa: consiste en hacer corresponder cada bloque de memoria principal a sólo una línea posible de caché. Cada dirección de memoria principal se divide en etiqueta, línea y palabra.

2-Correspondencia asociativa: Permite que cada bloque de memoria principal pueda cargarse en cualquier línea de la caché. cualquier bloque puede ser reemplazado cuando se va a escribir uno nuevo en la caché. La lógica de control de la caché interpreta una dirección de memoria como una etiqueta y un campo de palabra.

3-Correspondencia asociativa por conjuntos: Un bloque puede almacenarse en un conjunto restringido de lugares en la caché. La caché se divide en X conjuntos, cada uno de Y líneas, un bloque Z puede asignarse en cualquiera de Y líneas del conjunto.

[estas son las definiciones MÍNIMAS podría agregar las desventajas de cada método]**

-Política de reemplazo:
•Algoritmos de sustitución: cuando se introduce un nuevo bloque en la caché, debe sustituirse uno de los bloques existentes. Para el caso de correspondencia directa, sólo hay una posible línea para cada bloque particular y no hay elección posible. Para las técnicas asociativas, se requieren algoritmos de sustitución.

1-LRU (least-recently used, menos recientemente usado): se sustituye el bloque que se ha mantenido en la caché por más tiempo sin haber sido referenciado.

2-FIFO (first-in-first-out, primero en entrar-primero en salir): se sustituye aquel bloque del conjunto que ha estado más tiempo en la caché.

3-LFU (least-frecuenty used, menos frecuentemente usado): se sustituye aquel bloque del conjunto que ha experimentado menos referencias.

4-Aleatoria: se sustituye al azar.

Estos algoritmos deben implementarse en hardware (controlador de caché) para conseguir velocidad.

-Política de escritura:

definicion según runco 18/11/20: “cuando tengo que actualizar un dato, donde lo cambio manteniendo coherencia? porque tengo el mismo dato repetido en distintos niveles”
osea: políticas de actualización de datos (repetidos en distintos niveles) manteniendo coherencia**

Cuando la CPU escribe sobre una línea de caché, el bloque de memoria principal correspondiente debe ser actualizado en algún momento. Se debe evitar la inconsistencia de memorias.
estrategias:

Política de escritura en acierto (cuando tengo que escribir algo en la cache, elemento a escribir/cambiar se encuentra en la memoria caché):

• **ESCRITURA INMEDIATA (Write-through):** se actualizan simultáneamente la posición de la caché y de la memoria principal. Se genera mucho tráfico y retrasa la escritura. Se mantiene la coherencia en todo momento. Suele combinarse con la técnica “no carga en escritura” (no-write allocate).

• **POST-ESCRITURA (Write-back):** la información sólo se actualiza en la caché y esa información se marca con un bit de ACTUALIZAR o “sucio”. Cuando un bloque sea desalojado de la caché, se comprueba este bit y si está activo, se escribe la información de dicho bloque en memoria principal. (la memoria principal puede contener información errónea en algún momento). Suele combinarse con la técnica carga en escritura (write allocate).

Política de escritura en fallo (elemento a escribir NO se encuentra en la memoria cache):

• **CARGA EN ESCRITURA (write allocate):** la información se lleva de la memoria principal a la caché. Se sobrescribe en la caché. Habitual con write-back.

• **NO CARGA EN ESCRITURA (no-write allocate):** el bloque no se lleva a la memoria caché. Se escribe directamente en la memoria principal. Habitual con escritura inmediata (Write-through).

6
Buses

Un bus es un camino de comunicación entre dos o más dispositivos, un medio de transmisión compartido en donde cualquier señal transmitida por un dispositivo está disponible para que otros dispositivos conectados al bus puedan acceder a ella. Solo un dispositivo puede transmitir con éxito en un momento dado (para que sus señales no se solapen y se distorsionen).
Los computadores poseen distintos tipos de buses que proporcionan comunicación entre sus componentes a distintos niveles dentro de la jerarquía del sistema.
El bus conecta a los componentes principales de la computadora memoria, procesador, E/S) se denomina BUS DEL SISTEMA.

Estructura del Bus

Un bus está constituido por varias líneas capaces de transmitir señales binarias, a las cuales se les asigna una función particular y se pueden clasificar en tres grupos:

1- Bus (lineas) de datos: Se utilizan un “camino” para transmitir datos entre los módulos del sistema.

- Transmiten datos, a este nivel no existe diferencia entre “datos” e “instrucciones”.
- El ancho del bus es un factor clave a la hora de determinar las prestaciones (8, 16, 32, 64 bits).

2- Bus (lineas) de direcciones: Identifica la fuente o destino de un “dato”

- El ancho del bus de direcciones determina la máxima capacidad de memoria posible del sistema

3- Bus (lineas) de control: Se utiliza para controlar el acceso y el uso de las líneas de datos y de direcciones.

- Transmite información de señales de control y temporización

Tipos de buses

Dedicadas: está permanentemente asignada a una función o a un subconjunto físico de componentes del computador.

Multiplexadas: Uso de las mismas líneas para usos diferentes. Ventaja: uso de menos líneas, lo cual ahorra espacio y costes. Desventaja: Necesita una circuitería más compleja en cada módulo, además de que pueden reducirse las prestaciones debido a que los eventos que deben compartir las mismas líneas no pueden producirse en paralelo.

Método de arbitraje

Puesto que, en un instante dado, sólo una unidad puede transmitir a través del bus, se requiere algún método de arbitraje. Los diversos métodos se pueden clasificar como **centralizados o distribuidos**.

En un esquema centralizado, un único dispositivo hardware, denominado controlador del bus o árbitro, es responsable de asignar tiempos en el bus.

En un esquema distribuido, no existe un controlador central. En su lugar, cada módulo dispone de lógica para controlar el acceso, y los módulos actúan conjuntamente para compartir el bus.

En ambos métodos de arbitraje, el propósito de designar un dispositivo el procesador o un módulo de E/S como maestro del bus. El maestro podría entonces iniciar una transferencia de dato (lectura o escritura) con otro dispositivo que actúa como esclavo en este intercambio concreto.

Temporización

La temporización es la forma en la que se coordinan los eventos en el bus, pueden ser:

Temporización síncrona: la presencia de un evento en el bus está determinada por un reloj.

Temporización asíncrona: la presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo.

La temporización **síncrona** es más fácil de implementar y comprobar pero es menos flexible que la **asíncrona**, ya que todos los dispositivos deben utilizar la misma frecuencia de reloj.

Diferencias entre PCI y SCSI

Bus PCI (Peripheral Component Interconnect - Interconexión de Componente Periférico)
Es un bus estándar de computadoras para conectar dispositivos periféricos directamente en la placa base. de ancho de banda elevado, independiente del procesador.

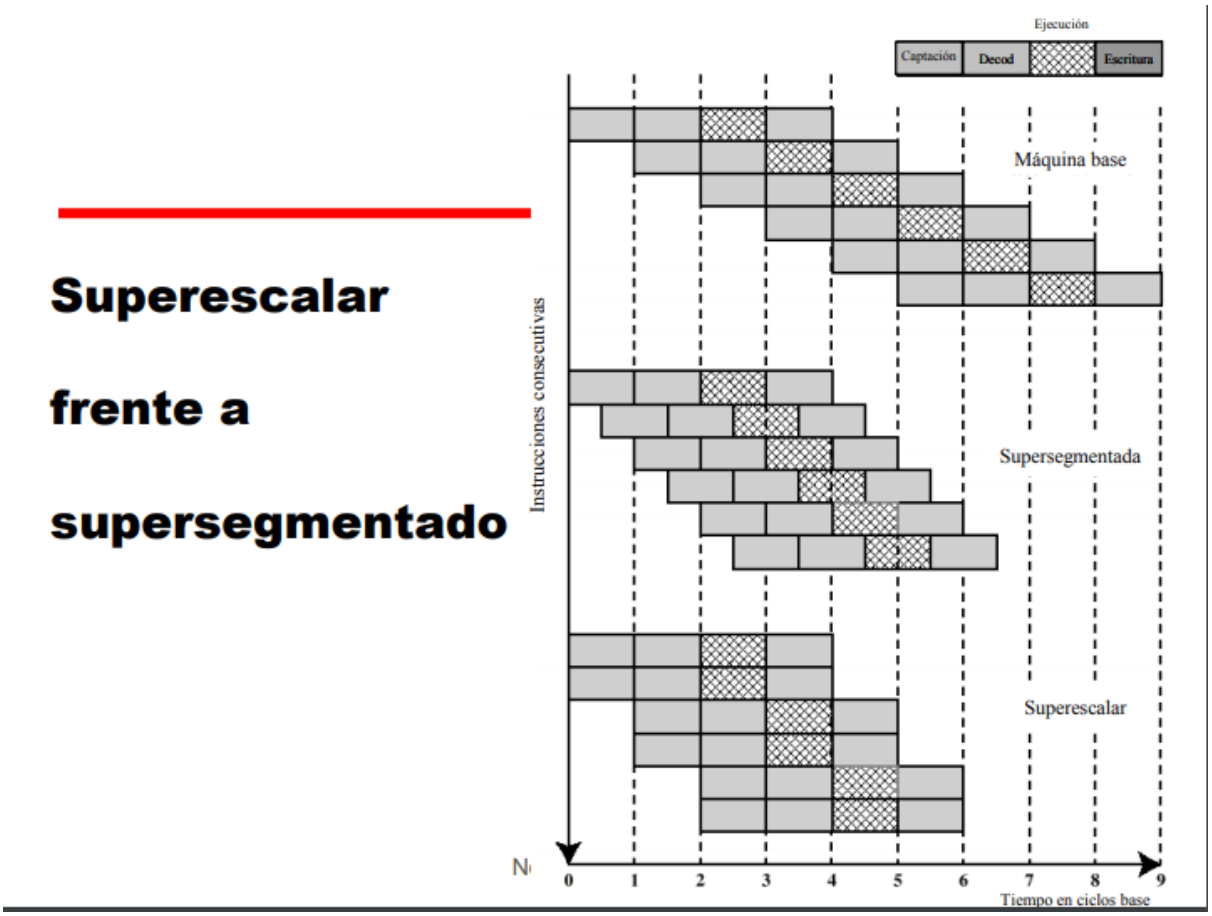
Bus SCSI (Small Computer System Interface - Pequeña interfaz del sistema de cómputo)
Es una interfaz estándar para la transferencia de datos entre distintos dispositivos del bus de la computadora. Se utiliza para comunicar dispositivos rápidos, como discos CD-ROM, dispositivos de audio y dispositivos de almacenamiento externo de datos. Requiere un controlador de interfaz.

7
Procesadores superescalares

Un procesador superescalar es aquel que usa múltiples cauces de instrucciones independientes, cada cause consta de múltiples etapas, de modo que puede tratar varias instrucciones a la vez.
El enfoque superescalar conlleva a la duplicación de algunas o todas las partes de la CPU/ALU. Capta varias instrucciones a la vez y a continuación intenta encontrar instrucciones cercanas que sean independientes entre sí y puedan ejecutarse en paralelo. Puede eliminar algunas dependencias innecesarias mediante el uso de registros adicionales y el renombramiento de las referencias a registros en el código original.

El concepto de diseño superescalar se asocia generalmente a RISC, pero también puede aplicarse en un CISC.

(imagen diagrama ignorar el supersegmentada)**



Paralelismo: (este tema no lo entendi bien pero saque la siguiente conclucion)**
Se refiere al grado en el que las instrucciones de un programa se pueden ejecutar en paralelo. Al ejecutarse más instrucciones en paralelo el grado de paralelismo y, por lo tanto, la aceleración de la máquina aumenta

Un procesador superescalar saca provecho de lo que se conoce como “paralelismo a nivel de instrucciones” que existe cuando instrucciones de una secuencia son independientes y entonces pueden ejecutarse en paralelo solapándose.

Paralelismo de instrucciones, (depende de las siguientes? preguntar) Limitaciones:

-Dependencia de datos verdadera:

Una instrucción necesita un dato producido por otra instrucción previa.

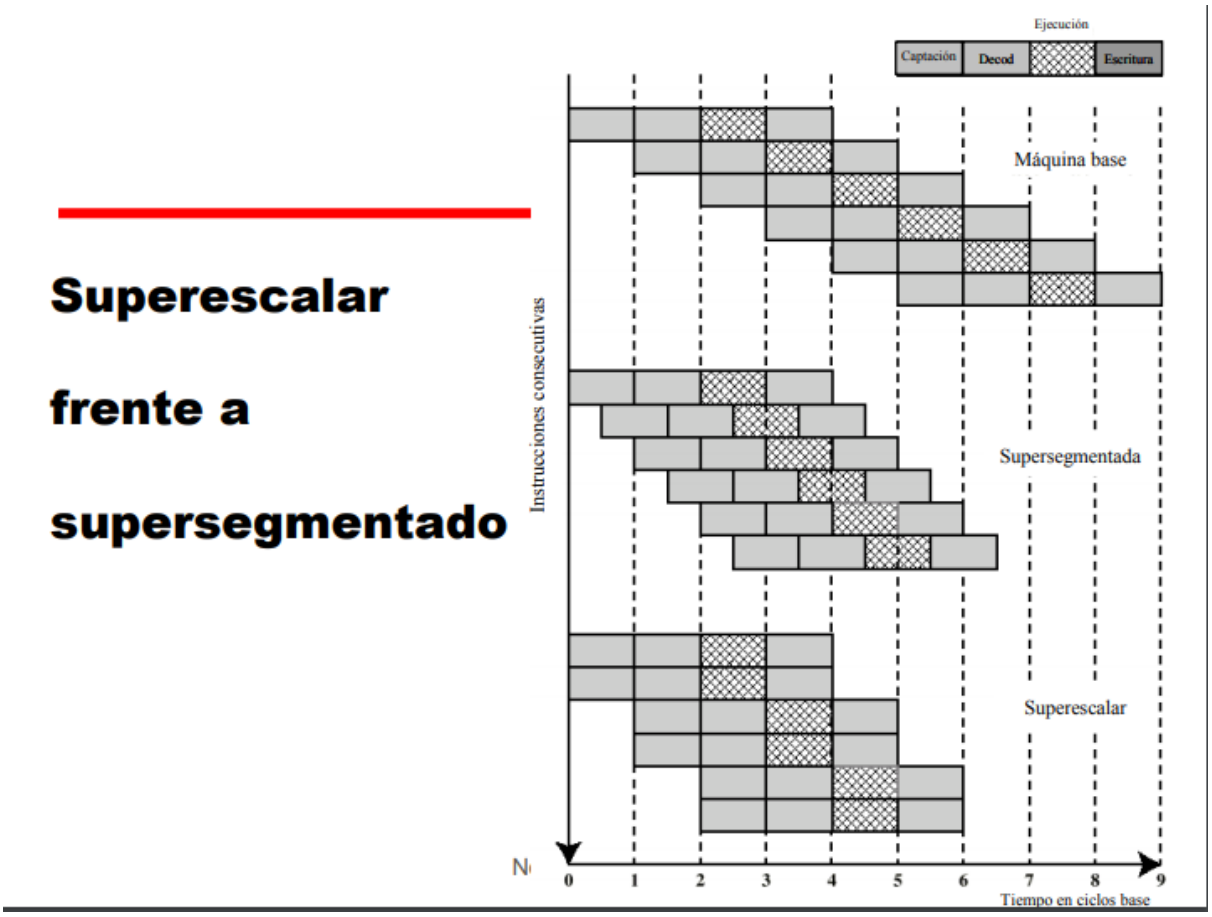
-Dependencia relativa al procedimiento: Las instrucciones que siguen a una bifurcación tienen una dependencia relativa al procedimiento en esa bifurcación y no pueden ejecutarse hasta que esta lo haga.

-Conflictos en los recursos: Es una pugna entre dos o más instrucciones por el mismo recurso al mismo tiempo. Presenta el mismo comportamiento que una dependencia de datos, sin embargo, los conflictos en los recursos pueden superarse duplicando estos.

-Dependencia de salida (WAW) y Antidependencia (WAR): El paralelismo a nivel de instrucciones depende de la frecuencia de dependencias de datos verdadera y dependencias relativas al procedimiento que haya en el código. (no encontre mejor explicacion)**

Superescalar frente a supersegmentado (no creo que lo tomen)**
La supersegmentación aprovecha el hecho de que muchas etapas del cauce realizan tareas que requieren menos de la mitad de un ciclo de reloj. De este modo, se dobla la velocidad de reloj interna, lo que permite la realización de dos tareas en un ciclo de reloj externo.

El procesador supersegmentado se queda atrás con respecto al procesador superescalar al comienzo del programa y en cada destino de un salto.



Políticas de emisión de instrucciones. (no creo que lo tomen, yo solo me acordaria los nombres de las emisiones y fue)**

Protocolos usados para emitir instrucciones para optimizar la utilización de los diversos elementos del cauce de los procesadores superescalares.

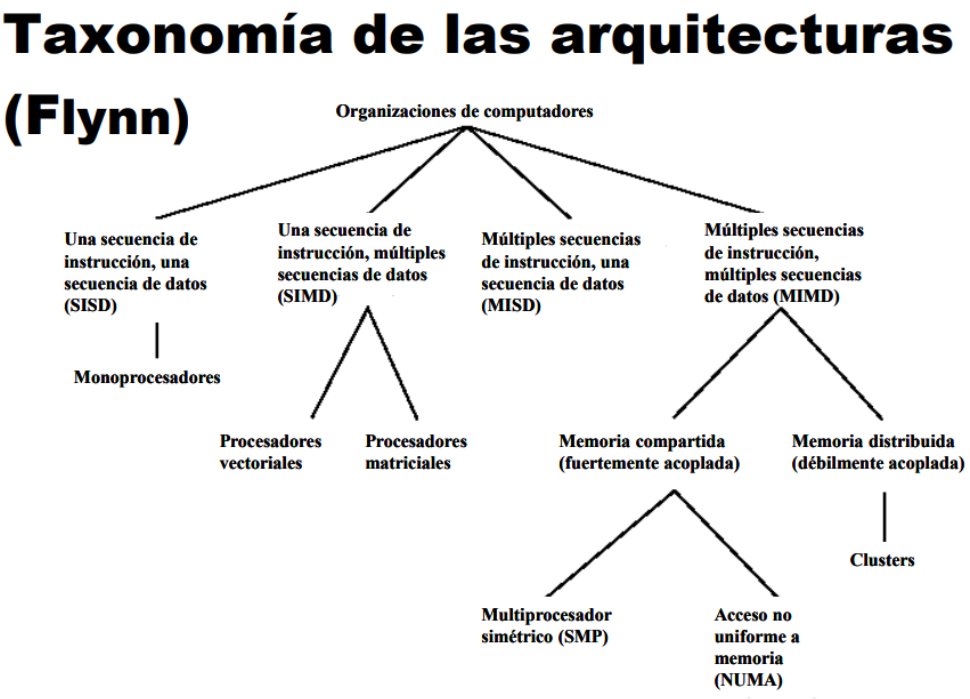
categorías:

- **Emisión y finalización en orden:** Las políticas de emisión de instrucciones más sencilla es emitir instrucciones en el orden exacto en que lo haría una ejecución secuencial y escribir los resultados en el mismo orden
- **Emisión en orden y finalización desordenada:** Se usa para mejorar la velocidad de las instrucciones que necesitan muchos ciclos. Puede haber cualquier número de instrucciones en la etapa de ejecución en un momento dado, hasta alcanzar el máximo grado de paralelismo de la máquina, ocupando todas las unidades funcionales.
  La finalización desordenada necesita una lógica de emisión de instrucciones más compleja que la finalización en orden. Además, es más difícil ocuparse de las interrupciones y excepciones.

- **Emisión y finalización desordenada**: El resultado de esta organización permite identificar instrucciones independientes que pueden introducirse en la etapa de ejecución. Para permitirla es necesario desacoplar las etapas del cauce de decodificación y ejecución.

Taxonomía de Flynn

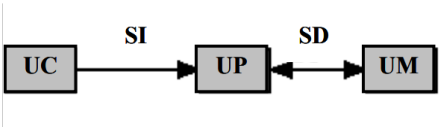
La taxonomía de Flynn clasifica a los sistemas de varios procesadores según sus capacidades de procesamiento paralelo.



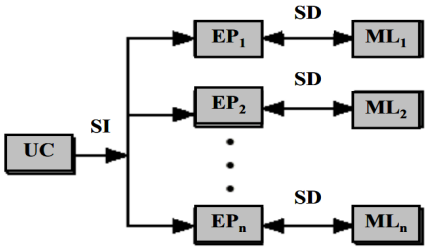
yo me acordaria solo ese diagrama y no las siguientes definiciones:**

SISD: una secuencia de instrucciones y una secuencia de datos. Un único procesador interpreta una única secuencia de instrucciones (SI) para operar con los datos almacenados en una única memoria(UM). Computadoras monoprocesador caen en esta categoría.

Organización SISD



SIMD: una secuencia de instrucciones y múltiples secuencias de datos. Una única instrucción máquina controla paso a paso la ejecución simultánea de un cierto número de elementos de proceso (EP). Cada elemento de proceso tiene una memoria dedicada (ML). Procesadores vectoriales y matriciales.

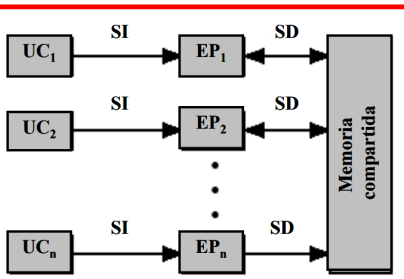


MISD: múltiples secuencias de instrucciones y una secuencia de datos. Se transmite una secuencia de datos a un conjunto de procesadores. Cada procesador ejecuta una secuencia de instrucciones diferente. Esta estructura nunca ha sido implementada.

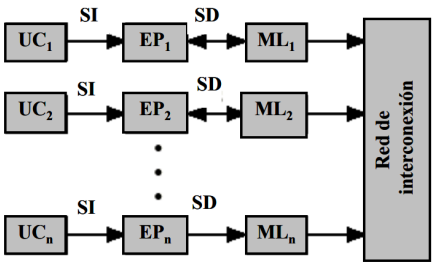
MIMD: múltiples secuencias de instrucciones y múltiples secuencias de datos. Significa que un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes, Los SMP, los “clusters” y los sistemas NUMA (sistemas paralelos) son ejemplos de esta categoría.

Se pueden dividir según la forma de comunicarse

Organización MIMD de memoria compartida



Organización MIMD de memoria distribuida



esto se pregunta en un examen:**

La taxonomía de Flynn clasifica a los sistemas de varios procesadores según sus capacidades de procesamiento paralelo. Una de las categorías es MIMD: Múltiples secuencias de instrucción, múltiples secuencias de datos.

Significa que un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes, Los SMP, los “clusters” y los sistemas NUMA son ejemplos de esta categoría.

En la organización MIMD, los procesadores son de uso general: cada uno es capaz de procesar todas las instrucciones necesarias para realizar las transformaciones apropiadas de los datos.

Los computadores MIMD se pueden dividir según la forma que tienen los procesadores para comunicarse:

- Memoria compartida (fuertemente acoplada): Los procesadores comparten una memoria común, Todos los procesadores tienen acceso a todas las partes de la memoria y los procesadores se comunican unos con otros a través de esa memoria.
    - Multiprocesador simétrico (SMP): **Varios procesadores comparten una única memoria mediante un bus compartido u otro tipo de mecanismo de interconexión.** El tiempo de acceso es igual a todas las regiones de memoria, así como el tiempo de acceso a memoria es el mismo para todos los diferentes procesadores.
    - Acceso no uniforme a memoria (NUMA): El tiempo de acceso a zonas de memoria diferentes puede diferir.
- Memoria distribuida (débilmente acoplada):
    - Clusters: Computadoras completas interconectadas mediante algún tipo de red que trabajan conjuntamente como un único recurso de cómputo, creando la ilusión de que se trata de una única máquina.

Clúster vs SMP

Ambos: Dan soporte a aplicaciones de alta demanda de recursos y están disponibles comercialmente (SMP es más antiguo).

SMP :Más fácil de administrar y configurar, más cercano a los sistemas de un solo procesador

CLUSTER: Superior escalabilidad incremental y superior disponibilidad

Definiciones que dejo como informacion por si acaso:

Multiprocesador simetrico (SMP)

Computadora autónoma con las siguientes características:

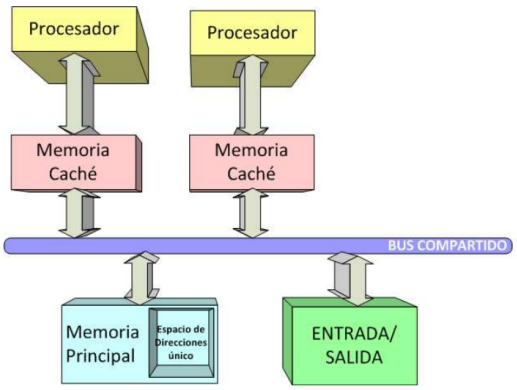
• Dos o más procesadores similares interconectados mediante un bus u otro tipo de sistema de interconexión que pueden desempeñar las mismas funciones.

• Comparten la memoria principal y las E/S. El tiempo de acceso a memoria es similar para todos los procesadores (UMA).

Un SMP tiene las siguientes ventajas frente a un monoprocesador:

- 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 y es escalado: En función de la cantidad de procesadores
- Cuidado: Bus compartido

Arquitectura de un SMP



Clusters

Computadoras completas interconectadas que trabajan conjuntamente como un único recurso de cómputo, creando la ilusión de que se trata de una única máquina.

Con clusters tenemos 4 beneficios:

- Escabilidad absoluta: Es posible configurar clusters grandes, que pueden superar las prestaciones de las computadoras independientes más potentes.
- Escabilidad incremental: Un cluster se configura de forma que sea posible añadir nuevos sistemas a este en ampliaciones sucesivas.
- Alta disponibilidad: El fallo de uno de los nodos no significa la pérdida del servicio. El software proporciona la tolerancia a fallos.
- Mejor relación precio/prestaciones: Es posible configurar un cluster con mucha potencia a menos costo que un computador independiente.

UMA

Acceso uniforme a memoria: Todos los procesadores pueden acceder a la memoria principal utilizando instrucciones de carga y almacenamiento. El tiempo de acceso es igual a todas las regiones de memoria, así como el tiempo de acceso a memoria es el mismo para todos los diferentes procesadores.

NUMA

Acceso no uniforme a memoria: Todos los procesadores tienen acceso a todas las partes de la memoria principal utilizando instrucciones de carga y almacenamiento. El tiempo de acceso a memoria principal de un procesador depende de la región a la que se acceda. Diferentes procesadores acceden a diferentes regiones de memoria principal a diferentes velocidades.

CC-NUMA

NUMA con coherencia de cache: Un computador NUMA en el que la coherencia de cache se mantiene en todas las caches de los distintos procesadores.

Pila (stack)

Una pila es una estructura de daros en la que el modo de acceso a sus elementos es de tipo LIFO (último en entrar, primero en salir) que permite almacenar y recuperar datos. Para el manejo de los datos se cuenta con dos operaciones básicas: apilar (push), que coloca un objeto en la pila, y su operación inversa, desapilar (pop), que retira el último elemento apilado. En cada momento sólo se tiene acceso a la parte superior de la pila, es decir, al último objeto apilado (top). La operación desapilar permite la obtención de este elemento, que es retirado de la pila permitiendo el acceso al siguiente (apilado con anterioridad), que pasa a ser el nuevo top.

En memoria principal se reserva un bloque de posiciones contiguas para la pila. Para su correcto funcionamiento se necesitan 3 direcciones, normalmente memorizadas en registros de la CPU:
Puntero de pila (SP): Dirección de la cabecera de la pila. Si se añade o elimina un elemento de la pila, el mismo incrementa o decrementa para actualizarse.
Base de la pila (BP): Dirección base del bloque reservado para la pila. Si se hace un POP con la pila vacía, se informa un error.
Límite de la pila: Dirección del otro extremo del bloque reservado. Si se hace un PUSH con la pila llena, se informa un error.

Principales funciones de la pila
- Pasaje de parámetros entre el programa principal y una, o varias, subrutinas:
o VALOR: apilando registros que contienen los datos.
o REFERENCIA: apilando las direcciones efectivas de los datos.
- Guardar el contexto del procesador:
o Guardar el PC: que contiene la dirección de la próxima instrucción a ejecutar.
o Guardar el estado del procesador en ese momento (flags).

1- Repaso pila y subrutinas - YouTube