Subrutinas – pasaje de parámetros – pila

Preguntas.

¿Qué es una pila? Describir el comportamiento con anidamiento de múltiples procedimientos/funciones utilizando pila.

Describa el funcionamiento y uso de la pila.

La pila es una zona de memoria (RAM) destinada para un uso especifico.

Cuando se corre un programa el s.o le otorga un sector de memoria a la pila.

Permite almacenar y recuperar datos, siendo el modo de acceso a sus elementos de tipo LIFO (último en entrar, primero en salir).

Operaciones de la pila:

Push: apilo datos en la pila

Pop: desapilo datos de la pila

La pila es usada por el sistema (salva la dirección de retorno cuando hay un call a una subrutina o se produce una interrupción por hardware que salva todo el contexto del programa) o por el usuario (push y pop).

Existe el registro Sp (stack pointer) que tiene la dirección de la cabecera de la pila y Bp (base pointer) que es utilizado por el usuario para direccionar la pila.

Como sp no es un registro base o índice, no está permitido [sp].

En el caso del simulador MSX88, no existe bp por lo tanto se usa bx.

El sp siempre es inicializado apuntando en la pila en 8000H por el s.o al comienzo de un programa.

Anidamiento de múltiples procedimientos/ funciones:

Antes de llamar a la subrutina se apilan los parámetros y cuando se hace el “call subrutinaX” automáticamente el s.o salva la dirección de retorno.

Luego estando en la subrutina se hacen estos pasos:

1: Salvar el viejo bp (push bp)

2: Salvar el estado de sp (bp = sp)

3: Opcional: reservar espacio para variables locales

4: Opcional: salvar valores de otros registros (si es que se van modificando)

5: Acceder a los parámetros (bx + desplazamiento)

6: Escribir sentencias a ejecutar (se puede llamar a otra subrutina . Si se vuelve a hacer una llamada se debe apilar los parámetros a pasar a la subrutina y la dirección de retorno. La subrutina llamada debe repetir los pasos anteriores más los que siguen)

7: Opcional: retornar parámetros

8: Regresar correctamente del procedimiento (sp tiene que estar apuntando a la dirección de retorno, desapilar todo lo que apile, y restaurar el valor de bp a su antiguo valor)

Explique los métodos de pasaje de argumentos a procedimientos o funciones.

Explique los métodos de pasaje de argumentos a procedimientos o funciones.

¿Qué métodos para pasaje de parámetro podemos usar en una computadora?

Las subrutinas en general necesitan datos de entrada y proveen datos de salida.

Estos parámetros pueden ser pasados de 3 formas diferentes:

Vía registros: su principal limitación es la poca cantidad de registros que hay.

El código tiene que estar documentado (por si otro programador también trabaja en el mismo programa)

Vía memoria: son variables globales (en memoria RAM), difícil de estandarizar, por si otro programador también trabaja en el mismo programa, se le tendría que decir el nombre de las variables o dirección de memoria.

Vía pila: método mas usado, es el verdadero “pasaje de parámetros”. Es muy utilizado en los lenguajes de alto nivel.

Es independiente de los registros y la memoria.

La pila es una zona de memoria RAM destinada a este uso, se tiene que saber usar ya que es utilizada por el s.o y por el usuario.

Interrupciones - PIC

Preguntas.

Finalidad de las interrupciones.

¿Qué es una interrupción? Describa cómo funcionan.

Las interrupciones surgieron ya que los distintos dispositivos operan a su ritmo, se requiere una sincronización entre los dispositivos y la cpu.

Una solución es llamada polling en la cual la cpu en loop pregunta si hay algún dato disponible, cuando el dispositivo este listo para pasarle cierto dato la cpu lo procesa. (y luego verifica si necesita mas datos o no siguiendo en loop)

Esta solución tiene una gran desventaja ya que la cpu permanece ociosa. Al estar preguntando todo el tiempo si hay algún dato disponible.

Además tendrá que esperar a los dispositivos más lentos.

Una solución mejor fueron las interrupciones (funcionalidad implementada en hardware para ayudar al procesador) que cada dispositivo interrumpe el flujo normal del procesamiento cuando necesita de la cpu.

Cuando el dispositivo externo está listo para aceptar más datos del procesador, el módulo de E/S de este dispositivo externo envía una señal de 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.

Hay interrupciones por hardware:

Es una llamada a una subrutina, se produce por una señal externa.

Son las verderas interrupciones.

Son producidas por dispositivos de entrada salida.

No están relacionados con el proceso en ejecución de ese momento.

Son conocidas como interrupt request.

La cpu verifica si hay una interrupción de hardware al final del ciclo de instrucción.

E interrupciones por software:

Es una llamada a una subrutina, es una instrucción especial (INT N)

Son funcionalidades del s.o.

Son llamadas interrupciones porque el mecanismo para encontrar la subrutina a ejecutar, es el mismo que la de interrupciones por hardware.

La cpu maneja esta interrupción con el ciclo de instrucción ya que es una instrucción.

¿Cómo se utiliza un controlador de interrupción?

Para que se utiliza un controlador de interrupciones.

Describa las características y el funcionamiento de un PIC.

Describa las características y esquematice el controlador de interrupciones PIC.

El procesador tiene una única entrada de interrupciones, al tener varios dispositivos externos que pueden interrumpir, la solución es un dispositivo controlador de interrupciones programable (PIC).

Es un chip que tiene varios registros, que administra los dispositivos que interrumpen a la cpu.

Adapta un montón de entradas y tiene una única salida al procesador llamada int.

PIC (controlador de interrupciones programable):

En el simulador MSX88 tiene 8 entradas: 4 utilizadas y las otras 4 no son utilizadas

Tiene los siguientes registros internos (de 8 bits cada uno)

IMR: registro de mascara de interrupción, habilita o deshabilita la interrupción

ISR: registro de interrupción en servicio

IRR: registro de petición de interrupción

Numero de vector (interrupciones de hardware asignadas):

INT0: se conecta a la tecla f10

INT1: se conecta a la línea OUT del timer

INT2: se conecta a la línea INT del handshake

INT3: se conecta al CDMA

INT4..7: no son utilizadas

EOI: end of interrupt

Funcionamiento (interrupciones vectorizadas):

1: le llega un pedido de interrupción al PIC (el PIC ya sabe de qué dispositivo se trata, ya que lo tiene identificado por que esta conectado al registro INTX). El PIC prioriza este con otros pedidos de interrupción (el numero mas bajo tiene mayor prioridad)

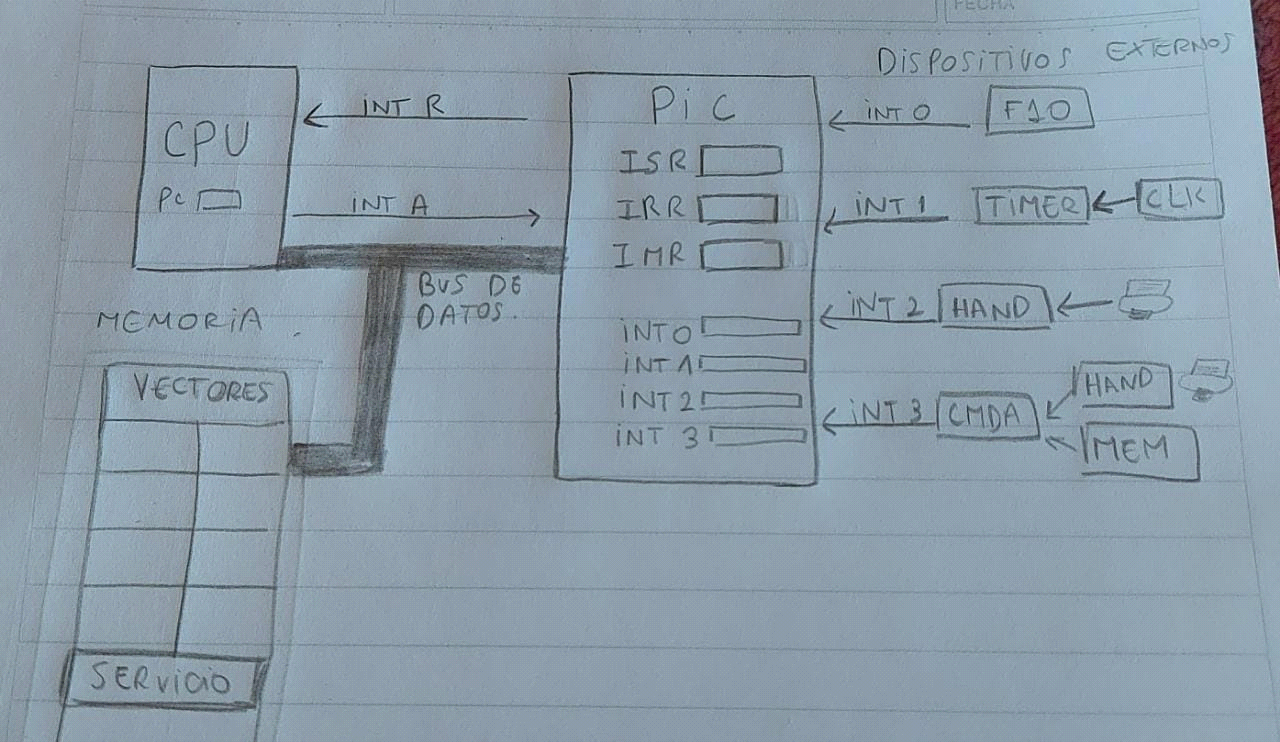
2: por la línea INTR le avisa al procesador que hay un pedido de interrupción.

3: la cpu acepta la interrupción y pide identificación del dispositivo por la línea INT A

4: el procesador recibe el puntero y lo usa para acceder a la tabla de vectores (el numero de vector lo multiplica por 4, ya que cada vector ocupa 4 bytes)

5: accede a la tabla de vectores y encuentra la dirección

6: salva en la pila la dirección actual, los flags y salta a la dirección de servicio.



¿A que método de atención lo llamamos “interrupciones vectorizadas”?

Explique el mecanismo de interrupción.

Interrupciones por hardware: al momento de identificar una interrupción se puede hacer de tres formas:

Interrupciones vectorizadas.

Interrupciones múltiples.

Interrupciones únicas.

Vectores de interrupción:

En el simulador MSX88 tenemos 256 entradas a la tabla, de los cuales c/u ocupa 4 bytes.

Hay algunos vectores ya preasignados (interrupciones de software)

INT 0: finalización

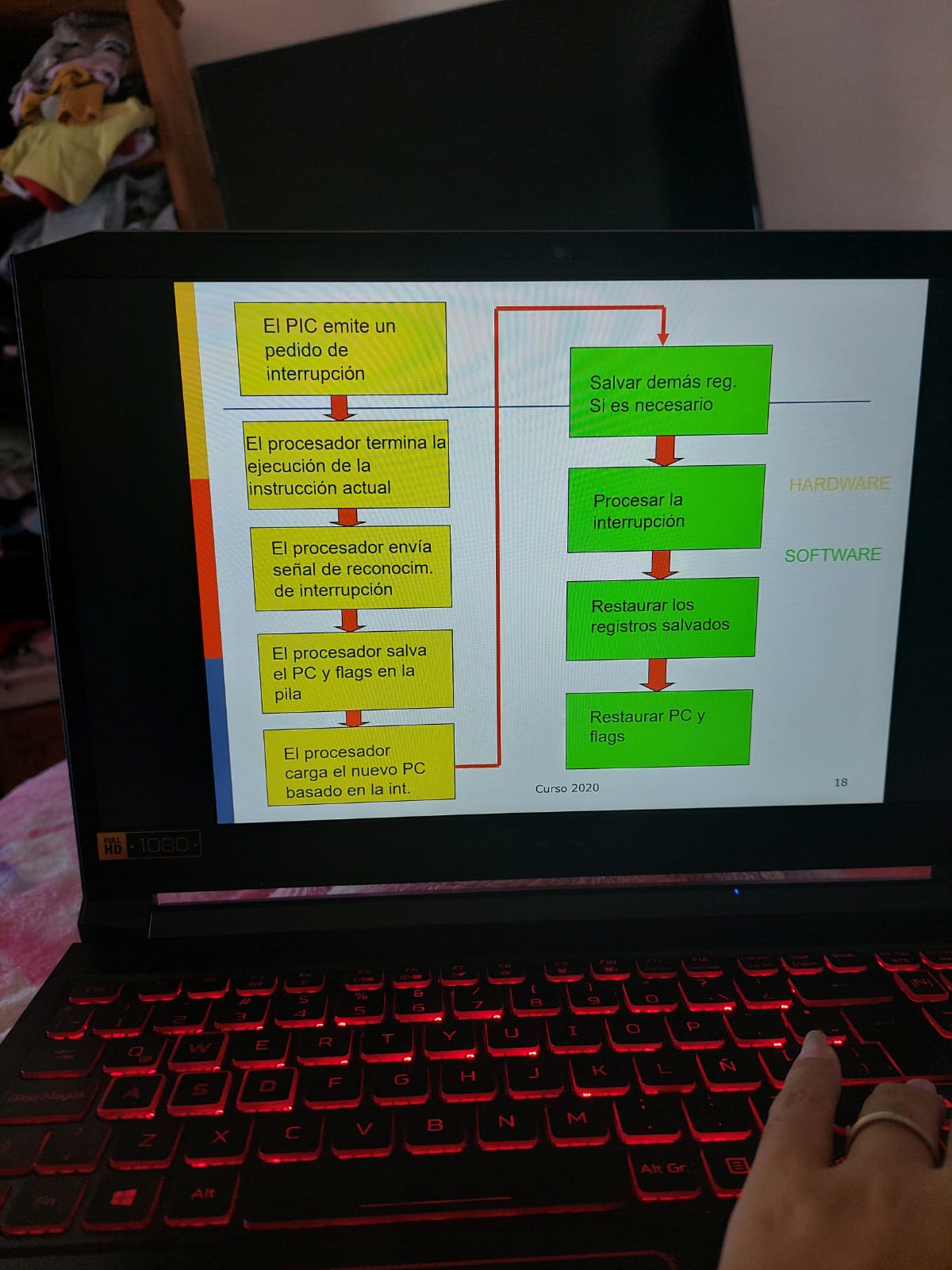
INT 3: puntos de parada

INT 6: leer

INT 7: escribir

Los otros restantes están libres para ser usados.

Mecanismo de interrupción:



¿Cuáles son las diferencias en la invocación de una subrutina y el gestor de interrupciones?

Subrutina:

Antes de llamar a una subrutina primero se salvan los registros luego cuando se llama a esta subrutina (call subrutinaX) el s.o automáticamente salva la dirección de retorno, luego ya en el procedimiento se salva el valor del viejo bp, se salva el valor de sp, opcionalmente se guarda lugar para las variables locales, opcionalmente se salva valores de otros registros, luego se acceden a los parámetros y se ejecutan las sentencias.

Ya por ultimo opcionalmente se retornan los parámetros y se regresa del procedimiento con la instrucción RET que saca de la pila 2 bytes (dirección de retorno) y se restaura el valor de bp.

Gestor de interrupciones:

Cuando ocurre una interrupción implica transferir el control a otro programa, el gestor de interrupciones, que salva el estado del procesador, responde a la causa de la interrupción, restaura el estado original del procesador y retorna a la ejecución normal del programa interrumpido.

La instrucción IRET que además de recuperar de la pila la dirección de retorno, saca de la pila 4 bytes (2 para la dir de retorno y 2 para el registro de las banderas).

¿Como y cuando utiliza las denominadas interrupciones por software?

Las interrupciones por software son instrucciones, entonces se van a ejecutar con el ciclo de instrucción.

Como se dan:

Son funcionalidades del s.o

En el simulador MSX88 hay 4 ya preasignadas:

Tipo 0: finalización del programa

Tipo 3: parada de espera del programa

Tipo 6: espera la entrada de un carácter

Tipo 7: imprimir en pantalla

Módulos de E/S - DMA

Preguntas.

¿Cómo funciona un módulo de E/S? . Esquematice

Describir la estructura de un módulo de E/S.

¿Cómo es la estructura interna de un módulo de E/S?

¿Como es la estructura de un módulo de E/S? Describa las posibles técnicas que puede utilizar una CPU para realizar operaciones de E/S.

La problemática de E/S surgio ya que los distintos dispositivos externos que se conectan a la cpu para ayudarla en sus tareas, manejan otros métodos de operación (diferentes cantidades, otras velocidades, etc.)

En general estos dispositivos son mucho mas lentos en comparación con la cpu y la memoria.

Se necesita de un modulo de E/S con cierta inteligencia para que haga:

Control y temporizacion de los dispositivos externos

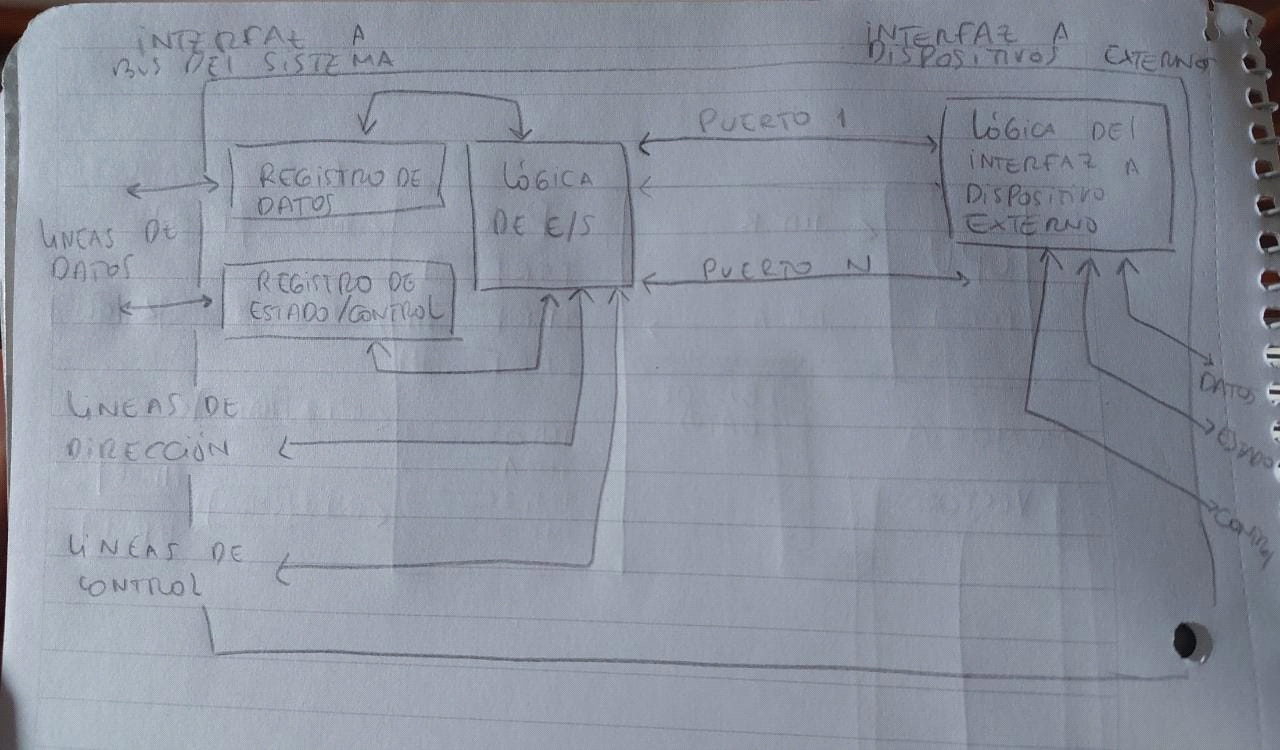
Comunicación con la cpu y memoria

Transferencia de datos

Almacenamiento temporal

Informar a la cpu el estado del periferico

El módulo de E/S debe ocultar los detalles de temporización, formatos y electromecánicos de los dispositivos externos para que el procesador pueda funcionar únicamente en términos de órdenes de lectura y escritura.



Del lado derecho tiene datos, estado y control que dependen del periférico (se puede conectar mas de un periférico por los diferentes puertos).

Y del lado izquierdo esta el bus de datos, el bus de direcciones y el bus de control (se conecta con la cpu).

En un modulo de e/s se puede encontrar lógica de e/s, hay lógica secuencial, lógica combinacional, hay registros de datos (Los datos que se transfieren a y desde el módulo se almacenan temporalmente en uno o más registros de datos) y registro de estado/control (proporcionan información del estado presente. Un registro de estado también puede funcionar como un registro de control, para recibir información de control del procesador).

Hay diferentes técnicas de gestión:

E/S con DMA

E/S programada con espera de respuesta

E/S con interrupciones

Describa las características fundamentales de un DMA.

¿Qué es DMA y cómo funciona?. Esquematice

Describa las características funcionales del acceso directo a memoria - DMA

Describa las características fundamentales de un DMA

Describa el funcionamiento de un controlador DMA(las etapas de transferencia)

Cuando hay grandes volúmenes de transferencia de datos surge un problema ya que la transferencia implica ir de memoria al procesador y luego del procesador al periférico.

El procesador solo sirve como intermediario, esto lo que genera es una pérdida de tiempo ya que hay que hacer un conjunto de instrucciones, como salvar en la pila, incrementar el puntero, etc. Sobrecargo la cpu.

la mejor solución es CDMA (controlador de acceso directo a memoria).

De la memoria se transfiere al CDMA y luego al periférico.

Se hace una transferencia de datos sin pasar por el procesador, el que controla esta operación es el CDMA.

El CDMA no tiene que salvar nada en la pila, porque no hay otro programa corriendo mas que la transferencia de datos.

(esa es la ventaja)

3 etapas en la transferencia:

inicialización: interviene la cpu.

Configuración de dispositivos, direcciones, a que dispositivo transferir, etc.

Transferencia de datos: no interviene la cpu.

Finalización: interviene la cpu.

El CDMA interrumpe a la cpu para avisar que termino.

Nota: No es una interrupción, el procesador no guarda el contexto.

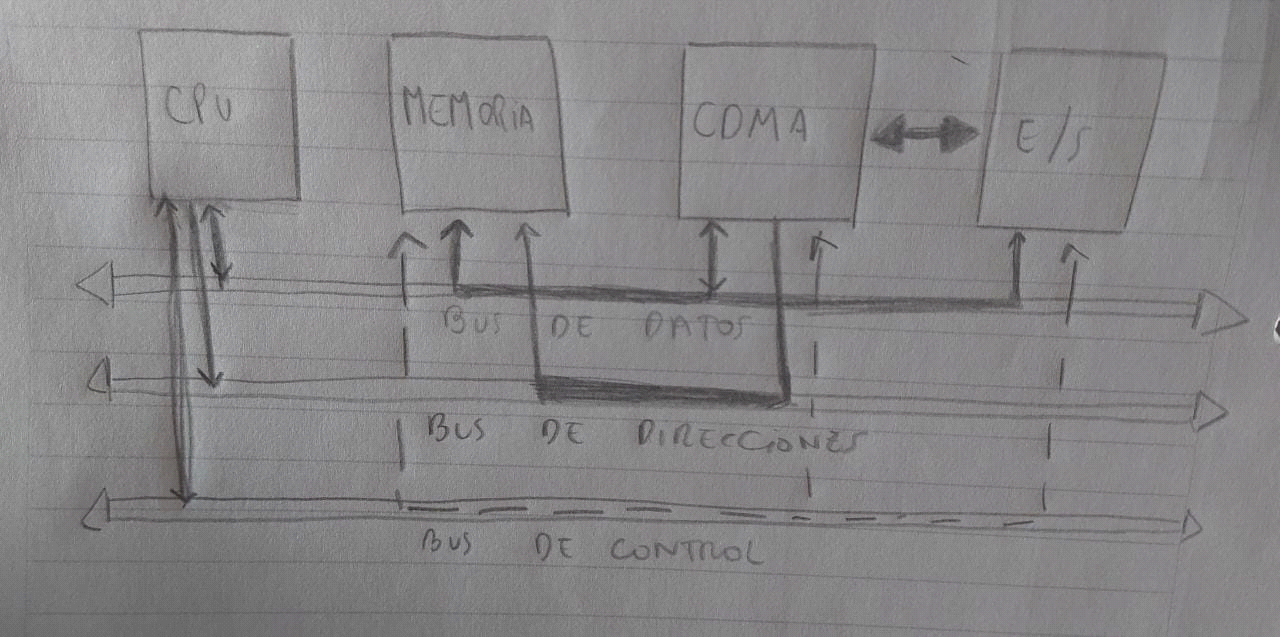
Modos de transferencia:

Modo ráfaga: cuando el CDMA toma el control del bus, transfiere los datos y no libera el bus hasta terminar.

Modo robo de ciclo: el CDMA toma el control del bus, transfiere el dato y luego devuelve el control del bus.

Luego la cpu ejecuta una instrucción y asi se van alternando entre la cpu y el CDMA

Modo transparente: el CDMA transfiere los datos en los casos que la cpu no utiliza los buses.



Describa como funciona de E/S programada con espera de respuesta.

Describa como funciona de E/S con interrupciones

e/s programada con espera de respuesta:

La cpu espera que el modulo e/s termine la operación.

Por lo tanto la cpu permanece ociosa durante un periodo de tiempo (no deseable).

Donde se usa: por ejemplo computadoras destinadas a ser instrumento de medida.

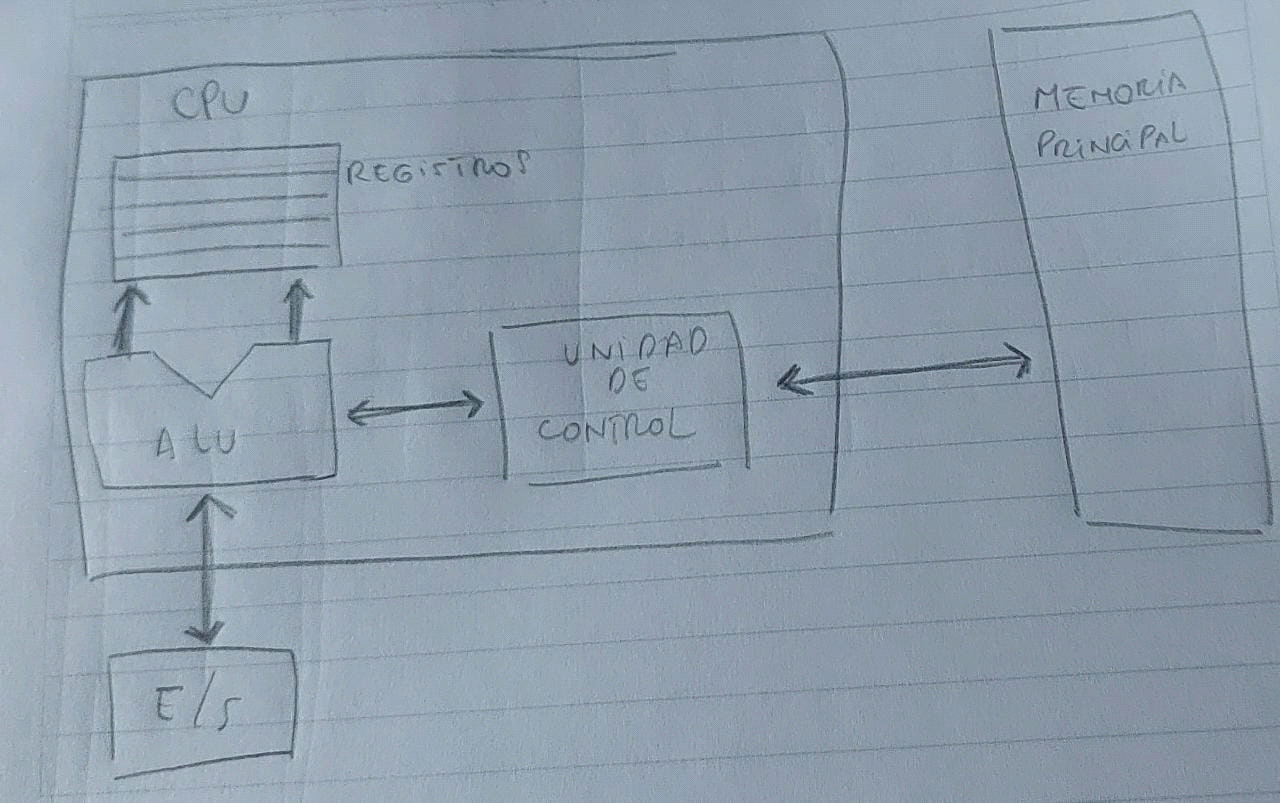
e/s con interrupciones:

la cpu no tiene que esperar la finalización de la tarea de e/s, puede seguir procesando.

No se repite la comprobación de los estados de los módulos.

El modulo envía un pedido de interrupción a la cpu cuando está listo nuevamente.

Modos de ubicación de e/s con una arquitectura Von Neumann.



/// e/s aislada , e/s mapeada en memoria?

/// incompleto.

Segmentación de cauce.

Preguntas.

¿Qué es la segmentación de cauce?

¿Qué entiende por segmentación de cauce? ¿Qué ventajas proporciona su implementación?

La segmentación de cauce (pipelining) es una forma particularmente efectiva 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 fases o etapas que permitan una ejecución simultánea.

Se hacen tareas al mismo tiempo, pero de distintas instrucciones y distintas etapas. (por que hay una sola alu)

La ventaja es que acelera la ejecución de las instrucciones. Ósea incrementa la productividad pero no por eso es que la ejecución de cierta instrucción se hace más rápido.

La instrucción entonces estará dividida en 5 etapas:

Búsqueda (if)

Obtención de la instrucción

Incrementa el pc (pc + 4 fijo)

Decodificación (id)

Decodificación de la instrucción

Leo registros

Verifico si salto (si es que salto actualizo pc)

Ejecución (ex)

Cálculos en la alu

Cálculos de dirección de operandos en memoria

Acceso a memoria (mem)

Acceso a memoria load/store

Escritura de los registros (wd)

Segmentación de cauce. Describir atascos por dependencia de datos y su solución.

Describa los métodos y técnicas para disminuir o evitar las paradas (stalls) que afectan el funcionamiento de los cauces

Describa tres (3) diferentes causas o motivos que pueden retardar un cauce de instrucciones segmentado

Describa las dependencias de los datos que puedan afectar un cauce segmentado y las posibles soluciones para evitarlos.

Explique los atascos producidos por saltos.

En un cauce segmentado, con secuencia de instrucciones independientes ¿Qué consecuencias trae el paso de una instrucción de salto? Analice los casos de salto incondicional y condicional. Mencione que posibles soluciones se pueden aplicar para evitar o disminuir las consecuencias.

Describir técnicas para el tratamiento de saltos condicionales

Describa las diferentes formas que pueden mejorar el funcionamiento de un cauce cuando ejecuta instrucciones de dependencias de control.

Aparecen problemas por empezar instrucciones antes de que terminen las anteriores, los atascos son situaciones que impiden a la siguiente instrucción que se ejecute en el ciclo que le corresponde.

Se dividen en 3 tipos:

Estructurales:

Cuando dos o más instrucciones intentan usar el mismo recurso de hardware a la vez.

(el hardware no está duplicado).

Solución: duplicar hardware (sumadores y restadores además de la alu)

Problema 1: Se accede a la memoria a la vez (etapa if y etapa mem)

Solución: Memoria separada (Harvard)

Problema 2: Se escriben y leen los registros a la vez (etapa id y etapa wb)

Solución: Escritura en el 1º subciclo y lectura en el 2º

Dependencia de datos:

dependencia verdadera.

Cuando una instrucción depende de los resultados de otra instrucción, de manera que ambas no pueden ejecutarse de forma solapada.

3 tipos:

**Lectura después de Escritura (RAW, dependencia verdadera)**

una instrucción genera un dato que lee otra posterior

Solución de software

Inserción de instrucciones NOP (se alarga el tiempo de ejecución y da tiempo de que la 1er instrucción termine)

Reordenación de código (más eficiente que el caso anterior, ya que no pierde tiempo)

Solución de hardware

Detención del cauce (estiro la instrucción [IF] [ID] [RAW] [RAW] [EX] etc ..)

Forwarding (adelantamiento) es utilizar el dato ni bien este disponible, uso el dato que está en el registro temporal (final de la operación EX), en vez de esperar a la etapa WB.

Se agrega hardware.

En el simulador se puede activar o desactivar.

Fácil de implementar si se identifican todos los adelantamientos y se comunican a los registros de segmentación correspondientes.

// los siguientes se producen cuando instrucciones

difieren en el número de ciclos para completarlas ó

hay ejecución desordenada//

**Escritura después de Escritura (WAW, dependencia en salida)**

una instrucción escribe un dato después que otra posterior

una solución es con ayuda del hardware en la cual hay un renombre de registros (ej r1a r1b) y que guarde los resultados y luego los escriba en orden.

**Escritura después de Lectura (WAR, antidependencia)**

**una instrucción modifica un valor antes de que otra anterior que lo tiene que leer, lo lea**

Dependencia de control:

Riesgos de control.

Problema de los saltos: se comienza a ejecutar la instrucción de abajo del salto pero hay que tirar todo el pipeline porque no era la instrucción correcta (ya que salto o lo que sea)

Solución de software

Instrucciones NOP

Reordenar el código con salto retardado

salto retardado: siempre se ejecuta la instrucción debajo del salto y luego toma el lugar el salto. Aumenta el tiempo de ejecución.

Solución de hardware

Detención del cauce

Predicción del salto (siempre alguna penalización hay) no se usa en mips

Mejor estrategia saber si en la etapa amarilla saber si saltar o no. Esto se usa en mips

BTB: Branch target buffer (mejora)

En la etapa ID se comprueba si el salto se lleva a cabo o no y donde saltar.

No se puede comprobar antes porque primero hay que decodificar la instrucción.

Si en la etapa IF se pudiera saber si hay que saltar o no, el pipe tomaría el camino correcto.

• Se “agrega” en la etapa IF una memoria “muy rápida” (BTB) donde se almacenan las direcciones (IP) de memoria donde están los saltos solamente. Si saltó o no la última vez y

la dirección donde saltó.

• Cada vez que se busca una instrucción se comprueba si esta dirección está en la tabla (BTB). Si está es porque es un salto.

• Así sabemos que es un salto, sin decodificar, y si saltó y donde la última vez.

• En la tabla se anotan la dirección del salto (PC), donde saltar y la “predicción” del salto.

• La tabla es analizada en la etapa IF.

• Es como una memoria caché de saltos.

RISC VS CISC

Preguntas.

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

Un poco de historia… (esto lo explico runco en unas clases de 2021) antiguamente se programaba con números en hexadecimal, luego con el lenguaje ensamblador aparecieron instrucciones (mov, add, sub, etc.) que el procesador era quien se encargaba del código de instrucción. Entonces claramente esto era más sencillo para el programador.

Hubo un problema, que los programadores debían conocer la arquitectura, es decir conocer el nombre de los registros, etc.

Por eso se llego a los lenguajes de alto nivel. Acá habría una brecha entre los lenguajes de alto nivel y las instrucciones de máquina.

Por cada instrucción de alto nivel, eran varias instrucciones de máquina, esto hacia a los programas más lentos en ejecutarse.

Los diseñadores para tratar de cerrar esta brecha hicieron arquitecturas CISC en las cuales había grandes conjuntos de instrucciones y muchos modos de direccionamiento.

Pero luego los compiladores ya no eran tan eficientes con las instrucciones de máquinas complejas, entonces ya se buscaba hacer programas mas pequeños y de menos acceso a memoria.

Se hicieron estudios

Se dio que los calls y los lazos eran los menos usados pero los que más instrucciones de maquina hacían.

(los procedimientos eran los que mas acceso a memoria generaban)

Aparece una nueva arquitectura RISC en la cual se obtuvo una conclusión de optimizar los accesos a memoria y aumento en el número de registros.

Tres elementos claves:

Conjunto reducido de instrucciones: más sencillo

Aumentos en el número de los registros (las variables son guardadas acá por lo tanto no hay mucho acceso a memoria, hablando de los procedimientos)

Segmentación de la instrucción (por la alta demanda de calls y de procedimientos).

Resumen

|  |  |
| --- | --- |
| CISC | RISC |
| Muchos modos de direccionamiento | Pocos modos de direccionamiento |
| Repertorio complejo de instrucciones | Conjunto reducido de instrucciones |
| Formato de instrucción variable | Formato de instrucción fijo |

Cache – jerarquía de memoria

Preguntas.

¿Por qué funciona una jerarquía de memoria?

Los programadores desean acceder a cantidades ilimitadas de memoria rápida.

Jerarquía de memoria: dentro del sistema de cómputo encontramos distintos tipos de memoria (distintas cantidades y distintos tipos de acceso), fabricados con tecnologías diferentes que todas trabajan en una manera organizada como si se comportaran como una memoria única.

La jerarquía de memoria funciona por las propiedades de los programas:

Localidad temporal los elementos de memoria referenciados recientemente en un corto lapso de tiempo volverán a serlo próximamente.

Ejemplo en los programas: procesos, funciones y estructuras de control (for y while)

Localidad espacial si accedo a una posición de memoria en poco tiempo accedo a la vecina

Ejemplo en los programas: vectores

Justifique el uso de dos niveles de caché

La memoria caché es un tipo de memoria específica que está preparada para servir de apoyo al procesador, y que es capaz de trabajar a velocidades muy elevadas. Esta memoria se divide en diferentes niveles, pero su función base es la misma: apoyar a la CPU para mejorar su rendimiento.

Para ello actúa como un sistema de almacenamiento de instrucciones y de datos capaz de comunicarse con el procesador a gran velocidad. Normalmente en la memoria caché se almacenan datos e instrucciones que el procesador va a necesitar para completar diferentes tareas a corto plazo. Con ello se evitan dos cosas muy importantes que podrían reducir el rendimiento:

Que el procesador tenga que volver a completar ciclos de trabajo para conseguir los datos o instrucciones que necesita.

Que tenga que acceder a la memoria RAM (más lenta que la caché) para obtener dichos datos o instrucciones.

Se observo que más del 90% de las veces que el procesador le solicita a la cache datos acertaba y menos del 10% no, por eso se pensó en tener otro nivel de la cache.

Se hizo la cache de la cache (l1 trabaja a la velocidad de los registros y siempre está dentro del procesador) y l2 la cache.

Caché: mencione algoritmos de reemplazo

Tengo la memoria cache llena

En correspondencia directa: el que ocupa el lugar del nuevo. (no tengo otra alternativa, solo hay una posible línea para cada bloque, se necesita una sustitución de esa línea sí o sí)

En correspondencia asociativa:

Hay un controlador de cache que lleva estas estadísticas.

Los algoritmos deben implementarse en hardware (para conseguir velocidad).

Estas son las estrategias:

LRU (menos recientemente usado)

Requiere controladores de tiempo

FIFO (más antiguo) primero en entrar – primero en salir

Requiere controles de acceso.

Se sustituye aquella línea que ha estado más tiempo en la cache.

LFU (menos frecuentemente usado)

requiere controles de uso.

Se sustituye aquella línea que ha experimentado menos referencias.

Forma aleatoria

Se sustituye una línea al azar.

políticas de escritura.

Analice las políticas de escritura desde el punto de vista de la coherencia de datos.

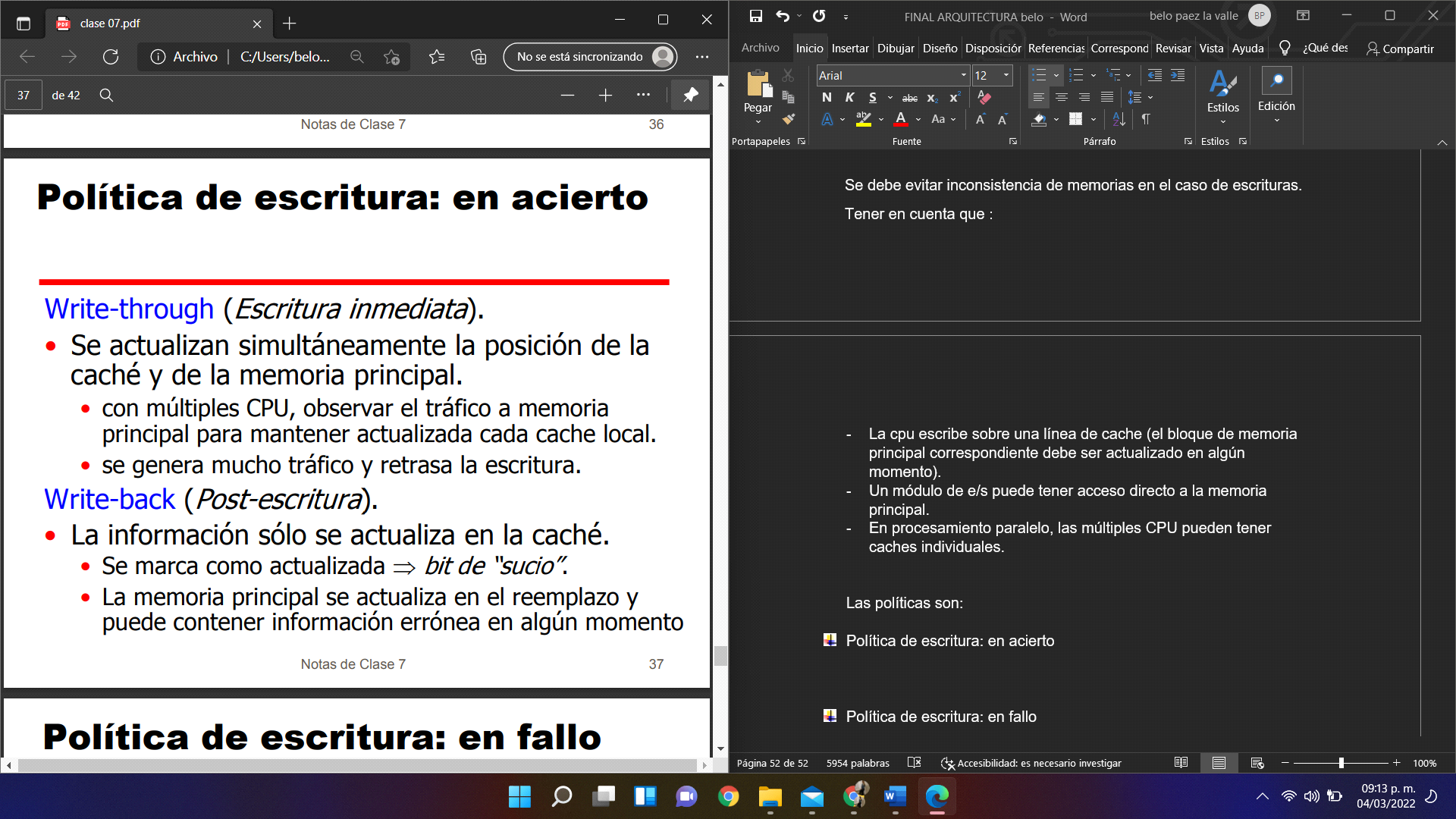
Políticas de escritura según correspondencia de datos.

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

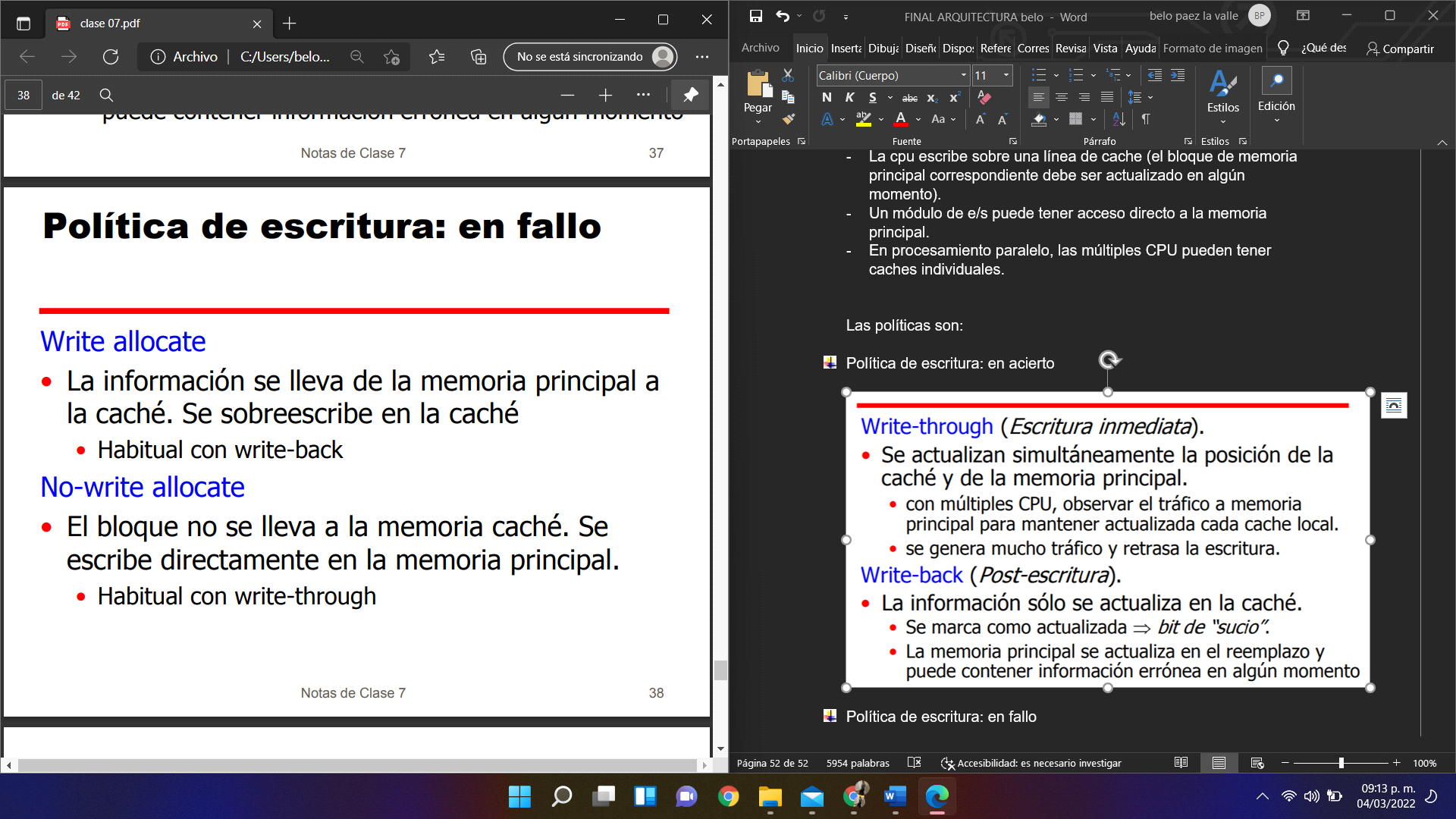
Analice las políticas de escritura de datos desde el punto de vista de la coherencia de los mismos en la jerarquía.

Las políticas son:

Política de escritura: en acierto (el dato que sobrescribo esta tanto en la cache como en memoria principal)



Política de escritura: en fallo (el dato que voy a sobrescribir no está en la cache. Estoy escribiendo en memoria principal, los principios de localidad dicen que si lo voy a usar conviene dejarlo en la cache).



Describa las políticas de ubicación y de reemplazo de bloques en memoria caché.

Describa las técnicas de ubicación de bloques

Describa las funciones que se utilizan en la política de ubicación de bloques en memoria caché.

Describa las funciones de correspondencia entre memoria principal y caché.

Funciones de correspondencia cache-memoria

Mencionar los tipos de correspondencia de la memoria caché.

Correspondencia directa: un bloque solo puede estar almacenado en un lugar de la cache.

N° LINEA CACHE = N° BLOQUE REF. **MOD** N° LINEAS CACHE.

\*ESTO LO HACE EL HARDWARE\*

Ventaja: es simple, poco costosa, es fácil buscarla, porque un bloque le corresponde a un único lugar por la cuenta.

Desventaja: hay pocas opciones de donde guardarla.

Ej. Hay una posición concreta para cada bloque

dado:

sí un programa accede a dos bloques que se

corresponden a la misma línea (diferentes bloques de

memoria principal) de forma repetida, las pérdidas de

cache (desaciertos) serán muy grandes.

Correspondencia totalmente asociativa: un bloque puede almacenarse en cualquier lugar de la cache.

(hay un controlador de cache que se fija cual está libre y lo guarda ahí)

Ventaja: muchas opciones de guardar la información. O sea un bloque de memoria principal puede colocarse en cualquier línea de la cache.

Desventaja: búsqueda Todas las etiquetas de las líneas se examinan para buscar una coincidencia = Búsqueda costosa (en tiempo principalmente)

Lo mejor es lo mejor hacer una mezcla de ambas técnicas:

Correspondencia asociativa por conjuntos: un bloque puede almacenarse en un conjunto restringido de lugares en la cache.

Un conjunto es un grupo de líneas de la cache

N° CONJUNTO = N° BLOQUE REF. **MOD** N° CONJUNTOS CACHE.

Que hay que tener en cuenta a la hora de diseñar una cache.

Diseño de una cache:

Organización y tamaño: distintos niveles de cache

Políticas de ubicación (funciones de correspondencia): el dato que tengo en la memoria principal en que línea de la cache lo guardo?

Políticas de reemplazo (algoritmos): tengo la cache llena, que línea reemplazo?

Políticas de escritura: cuando escribo algo, en que nivel lo hago? Coherencia de datos.

¿La coherencia de datos de un sistema jerárquico de memoria se ve afectado por el uso del DMA?

La DMA puede llevar a problemas de coherencia de caché. Si tenemos una CPU equipada con una memoria caché y una memoria externa que se pueda acceder directamente por los dispositivos que utilizan DMA. Cuando la CPU accede a X lugar en la memoria, el valor actual se almacena en la caché. Si se realizan operaciones posteriores en X, se actualizará la copia en caché de X, pero no la versión de memoria externa de X. Si la caché no se vacía en la memoria antes de que otro dispositivo intente acceder a X, el dispositivo recibirá un valor caducado de X.

Del mismo modo, si la copia en caché de X no es invalidada cuando un dispositivo escribe un nuevo valor en la memoria, entonces la CPU funcionará con un valor caducado de X.

Buses

Preguntas.

¿Qué es un Bus? Describa los diferentes tipos, métodos de arbitraje y técnicas de sincronización. Mencione las principales diferencias entre PCI y SCSI.

Que es un Bus, tipos de buses, temporización y métodos de arbitraje

¿Qué elementos definen un bus?

Es un camino de comunicación entre dos o más dispositivos.

Suele agruparse: Varios caminos de comunicación o líneas con

función común

Bus de direcciones: identifica la fuente o destino de un dato.

Bus de datos: viajan los datos

Bus de control: transmite señales de control y temporización

Como son

Es un conjunto de conductores eléctricos paralelos.

Líneas de metal.

Poseen conectores para colocar ‘tarjetas’

Jerarquía

Bus local y bus del sistema:

Buses rápidos y cortos.

Buses propietarios (no estándares)

Optimizados para la arquitectura

N° fijo de dispositivos de prestaciones conocidas

Bus de expansión:

Buses más largos y lentos

Bus abierto (estándar)

Accesible por el usuario

N° indeterminado de dispositivos de distintas prestaciones

Tipos de buses

Las líneas del bus se pueden dividir en dos tipos genéricos:

Dedicadas: está permanentemente asignada a una función o a un subconjunto físico de componentes del computador. Ej: el uso de líneas separadas para direcciones y para datos.

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.

Métodos de arbitraje

Más de un módulo puede necesitar el control de un bus y como solo una unidad puede transmitir a través del bus en un momento dado, se requiere algún método de arbitraje:

Centralizados: Un único dispositivo hardware (llamado controlador del bus o árbitro) es responsable de asignar tiempos en el bus. El dispositivo puede estar en un módulo separado o ser parte del procesador.

Distribuidos: No existe un controlador central, sino que cada módulo dispone de lógica para controlar el acceso y los módulos actúan conjuntamente para compartir el bus.

Técnicas de sincronización/temporización

La temporización es la forma en la que se coordinan los eventos en el bus. Puede ser:

Temporización síncrona:

La presencia de un evento en el bus está determinada por un reloj.

El bus incluye una línea de reloj a través de la cual se transmite una secuencia en la que se alternan intervalos regulares de igual duración.

Un intervalo (de un 1 seguido de un 0) se conoce como ciclo de reloj o ciclo de bus y define un intervalo de tiempo unidad.

Todos los dispositivos del bus pueden leer la línea de reloj y todos los eventos empiezan al principio del ciclo de reloj (suelen sincronizar en el flanco de subida)

La mayoría de los eventos se prolongan durante un único ciclo de reloj

Temporización asíncrona:

La presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo

La síncrona es más fácil de implementar y comprobar, pero es menos flexible que la temporización asíncrona, ya que todos los dispositivos de un bus síncrono deben utilizar la misma frecuencia de reloj y el sistema no puede aprovechar mejoras en las prestaciones de los dispositivos. Con la asíncrona, pueden compartir el bus una mezcla de dispositivos lentos y rápidos.

Diferencias entre PCI y SCPI

El bus PCI es un bus de ancho de banda elevado, independiente del procesador, que se puede utilizar como bus de periféricos. Comparado con otras especificaciones comunes de bus, proporciona mejores prestaciones para los subsistemas de E/S de alta velocidad (ej: los adaptadores de pantalla gráfica, los controladores de interfaz de red, los controladores de disco, etc). El PCI ha sido diseñado específicamente para ajustarse económicamente a los requisitos de E/S de los sistemas actuales; se implementa con muy pocos circuitos integrados, y permite que otros buses se conecten al bus PCI.

El PCI está diseñado para permitir una cierta variedad de configuraciones basadas en microprocesadores, incluyendo sistemas tanto de uno como de varios procesadores.. Utiliza la temporización síncrona y un esquema de arbitraje centralizado.

SCSI sólo se utiliza para dispositivos de almacenamiento y debe tener un controlador de interfaz

Superescalares

Preguntas.

¿Qué características definen un procesador como superescalar?

¿Qué características definen un procesador como superescalar?

Avanzando la tecnología aparecieron los diseños de causes segmentados, pero luego surgieron dos técnicas de ejecución con mayores prestaciones.

Procesadores superescalares y supersegmentados.

Procesadores supersegmentados:

Muchas operaciones no necesitan de todo un ciclo de reloj, por lo cual al ciclo se lo sub-divide. Es decir empieza la otra etapa de una instrucción cuando la anterior etapa de otra instrucción esta por la mitad.

Dando así mayor productividad, mayor grado de paralelismo y mayor rapidez.

Los procesadores superescalares llevan a cabo mas de una instrucción simultáneamente (instrucciones independientes).

Esto conlleva a la duplicación de todas o algunas partes de la cpu.

El grado de paralelismo es mayor y también la aceleración de la maquina.

Implementación superescalar

Estrategias de captación simultánea de múltiples

instrucciones.

• Lógica para determinar dependencias verdaderas

entre valores de registros y mecanismos para

comunicar esos valores.

• Mecanismos para iniciar o emitir múltiples

instrucciones en paralelo.

• Recursos para la ejecución en paralelo de

múltiples instrucciones.

• Mecanismos para entregar el estado del

procesador en un orden correcto

Describa las políticas de emisión de instrucciones en un cauce segmentado

• Emisión y finalización en orden

(no hace falta ventana de instrucciones)

La más sencilla. Emitir instrucciones en el orden exacto en que lo haría una ejecución secuencial y escribir los resultados en ese mismo orden.

• Emisión en orden y finalización desordenada

(no hace falta ventana de instrucciones)

La finalización desordenada se usan en los procesadores RISC escalares para mejorar la velocidad de las instrucciones que necesitan muchos ciclos. Con la finalización desordenada 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 emisión de instrucciones se para cuando hay una pugna por un recurso, una dependencia de datos o una dependencia relativa al procedimiento.

Aparece la dependencia de salida o dependencia escritura-escritura.

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.

(registro temporal en donde se guarda el resultado del registro, luego de que la instrucción anterior termine puedo escribir definitivamente en el registro)

• Emisión y finalización desordenada

(necesito ventana de instrucciones) es la que tiene mejor performance.

Con la emisión en orden, el procesador solo decodificara instrucciones hasta que haya conflicto o dependencia, no puede ir más allá del punto del conflicto buscando instrucciones que podrían ser independientes (hasta que el mismo no se resuelva) y que podrían introducirse provechosamente en el cauce

Para permitir la emisión desordenada, es necesario desacoplar las etapas del cauce de decodificación y ejecución, mediante un buffer llamado ventana de instrucciones. Cuando un procesador termina de decodificar una instrucción, coloca ésta en la ventana de instrucciones. Mientras el buffer no se llene, puede continuar captando y decodificando nuevas instrucciones. Cuando una unidad funcional de la etapa de ejecución queda disponible, se puede emitir una instrucción desde la ventana de instrucciones a la etapa de ejecución. Cualquier instrucción puede emitirse, siempre que: necesite la unidad funcional particular que esté disponible y que ningún conflicto ni dependencia la bloqueen.

Así el procesador tiene capacidad de anticipación, permitiendo identificar las instrucciones independientes que pueden introducirse en la etapa de ejecución. Las instrucciones se emiten desde la ventana de instrucciones sin que se tenga en cuenta su orden original en el programa.

¿De qué depende el paralelismo de una maquina superescalar?

“es una medida de la capacidad del procesador para

sacar partido del paralelismo de instrucciones”

Depende de:

• Número de instrucciones captadas por ciclo

• Número de unidades funcionales

• Mecanismos para localizar instrucciones independientes

• Identificar paralelismo y organizar F, D y E en paralelo.

• Renombre de registros (dep. de salida o antidep.)

• Ventana de instrucciones (emisión desordenada)

¿Cuál es el objetivo de la técnica de renombre?

• Las dependencias de salida y antidependencias surgen

porque los valores de los registros no pueden reflejar la

secuencia de valores dictada por el flujo del programa.

• Conflictos de almacenamiento

• Esto puede detener alguna etapa del cauce.

• Cuando la ejecución de una instrucción guarda un resultado

en Registro, se almacena en un registro nuevo

Renombramiento de Registros

• Los registros se asignan dinámicamente

• o sea, las referencias posteriores son a los registros nuevos.

Causas de retraso

El paralelismo de una instrucción es que estas sean independientes y puedan ejecutarse en paralelo.

Esto no siempre es asi por lo cual aparecen los atascos (los mismos que cuando se vio cauce segmentado)

Estas limitaciones pueden ser:

Dependencia de datos verdadera (RAW)

Antidependencia (WAR)

Dependencia de salida (WAW)

Dependencia relativa al procedimiento

Conflictos en los recursos

Taxonomía de Flynn

Preguntas.

¿ Que es la taxonomía de Flynn?

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

¿Cuáles son las arquitecturas que pueden encontrarse en la configuración MIMD de la taxonomía de Flynn?

Una de las categorías de la taxonomía de Flynn 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.

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, entonces cada procesador accede a los programas y datos almacenados en la memoria compartida 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 a memoria principal es aproximadamente el mismo para cualquier procesador.

Acceso no uniforme a memoria (NUMA): El tiempo de acceso a zonas de memoria diferentes puede diferir.

Memoria distribuida (débilmente acoplada):

Clusters: Conjunto de computadores monoprocesadores independientes, o de SMP, que se interconectan. La comunicación entre los computadores es mediante conexiones fijas o mediante algún tipo de red.

¿Qué características posee un multiprocesador simétrico (SMP)?

Características de un computador  SMP:

Hay dos o más procesador similares de capacidades comparables

Estos procesadores comparten la memoria principal y las E/S, y están interconectados mediante un bus u otro tipo de sistema de interconexión, de forma que el tiempo de acceso a memoria es aproximadamente el mismo para todos los procesadores (UMA).

Todos los procesadores comparten los dispositivos de E/S, bien a través de los mismos canales, o bien mediante canales distintos que proporcionan caminos de acceso al mismo dispositivo.

Todos los procesadores pueden desempeñar las mismas funciones (de ahí el término simétrico)

El sistema está controlado por un sistema operativo integrado, que proporciona la interacción entre los procesadores y sus programas en los niveles de trabajo, tarea, fichero, y datos

Ventajas potenciales de un SMP con respecto a una arquitectura monoprocesador:

Mayores prestaciones: Si el trabajo a realizar puede organizarse en paralelo, con varios procesadores será mejor que con uno solo.

Buena disponibilidad: Un fallo en un procesador no detendrá la computadora, ya que todos los procesadores pueden realizar las mismas funciones

Crecimiento incremental: Se pueden añadir más procesadores

Escalado: En función de la cantidad de procesadores de un sistema, hay una gama diferente de productos con prestaciones y precios diferentes

La existencia de varios procesadores es transparente al usuario, el SO es quien sincroniza los procesadores y planifica los hilos o procesos, asignándolos a los distintos procesadores.