Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

UNIDAD DE CONTROL SEGMENTADA

UNIDAD DE CONTROL SEGMENTADA

UNIDAD DE CONTROL SEGMENTADA

UNIDAD DE CONTROL SEGMENTADA

Índice

UNIDAD DE CONTROL SEGMENTADA

Presentación

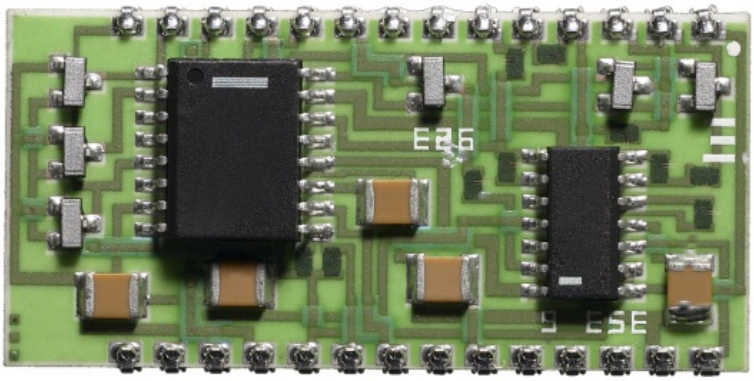
En este tema, veremos cómo aumentar el rendimiento de los procesadores individuales. Es decir, aumentar el rendimiento de un procesador en sí mismo, no duplicando el número de procesadores o unidades funcionales.

Explicaremos en qué consiste el principio de funcionamiento de la segmentación y que problemas conllevará su diseño en la implantación de estos sistemas en procesadores.

Analizaremos los tipos de fallos que se pueden generar, como los fallos de control de datos o los fallos estructurales, y se estudiarán posibles métodos para solucionarlos, así como los puntos fuertes e inconvenientes de cada uno de ellos.

Dividiremos el estudio de todo esto en los siguientes puntos principales:

- Unidad de control monolítica.
- Procesador segmentado.
- Riesgos estructurales, riesgos de datos, riesgo de control.

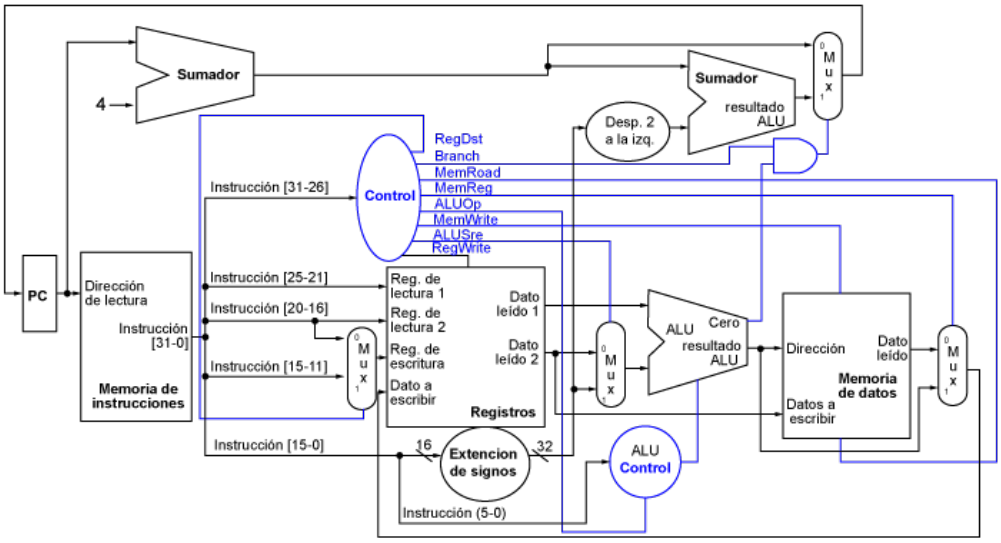


UNIDAD DE CONTROL SEGMENTADA

Unidad de control clásica: monolítica

Hasta ahora, nuestro microprocesador tenía un proceso de trabajo secuencial. Este tipo de unidades de control se conoce como máquinas secuenciales (o monolíticas).

En la imagen adjunta, tenemos un ejemplo de unidad de control monolítica. Todo lo que está en azul es el bus de control y la unidad de control. En este procesador, el controlador va dirigiendo cada una de las etapas por las que tiene que pasar una instrucción para que al final termine ejecutada.



UNIDAD DE CONTROL SEGMENTADA

Proceso de trabajo secuencial

Es decir:

- Trae una instrucción de memoria.
- La decodifica.
- Carga los operandos que intervienen desde memoria.
- Ejecuta la instrucción.
- Guarda los resultados en memoria.

Y así sucesivamente. En el mejor de los casos, se necesitan cinco ciclos de reloj para poder ejecutar una instrucción. Y eso considerando que solo necesita cinco pasos, ya que en instrucciones muy complejas como el producto de números reales donde la fase de ejecución puede durar mucho tiempo, este número de ciclos puede incrementarse.

UNIDAD DE CONTROL SEGMENTADA

La segmentación, descripción general

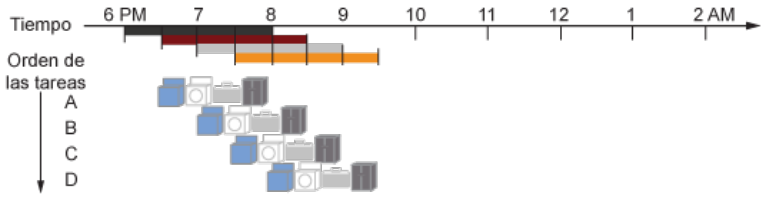
Vamos a empezar la descripción de lo que es una unidad de control segmentada con un ejemplo.

Imaginemos que tenemos que hacer la colada en casa. El proceso completo para hacer la colada es:

- Echar la ropa en la lavadora y esperar que termine (1 hora).
- Echar la ropa a la secadora y esperar que termine (1 hora).
- Planchar cada prenda, aunque para ello tenemos la planchadora automática que prácticamente lo hace sola (1 hora).
- Guardar la colada, aunque lo hará nuestro compañero de piso, pues nosotros le hemos planchado (0.5hora).

En total, 3.5 horas de tarea.

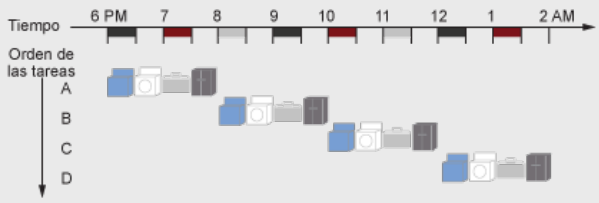
Imaginemos ahora que llegamos de un fantástico viaje, por lo que llegamos con las maletas llenas y un montón de coladas por delante. Si por cada colada completa necesitamos 3.5 horas, difícilmente podremos hacer más de tres coladas al día, pero ya que tenemos ayuda en este proceso, bien de las máquinas o de nuestro compañero de piso, podemos acelerar este proceso solapando tareas.



Este método permite reducir el tiempo total de procesamiento de una secuencia de coladas, pero no reduce el tiempo en hacer una colada, la ganancia en tiempo es proporcional al número de etapas en las que se divida nuestro proceso original, y cuantas de ellas se puedan solapar.

UNIDAD DE CONTROL SEGMENTADA

Horas de tarea



Horas de tarea

Una vez que la primera lavadora ha terminado, el proceso continúa con la secadora y así sucesivamente. Pero ¿que hace la lavadora mientras la secadora esta en su ciclo? Nada, está parada, por lo que podríamos utilizar esta inactividad para que vaya lavando ya la segunda colada. Transcurrida una hora, tanto la secadora como la lavadora habrán terminado, por lo que es momento de ponernos a planchar la ropa que sale de la secadora. Al mismo tiempo, la ropa de la segunda colada la pasamos a la secadora, con lo que podemos poner en la lavadora una tercera colada.

Si continuamos con este proceso, el tiempo en hacer todas las coladas se reduce sustancialmente, pero con una curiosidad, ¿Cuánto es el tiempo que tarda una camisa desde que entra en la lavadora hasta que esta guardada en el armario? Las 3,5 horas, es decir igual que antes.

Número de etapas

Por ejemplo, si en el caso anterior, nuestro compañero de piso no estuviese por la labor de echarnos una mano, las tareas de planchar y colocar no podrían ser solapadas en el tiempo. Por lo que la ganancia no es tan buena.

Para nuestro caso original, el tiempo de hacer una colada es de 3,5horas, mientras que para dos seria de 4,5 horas y para 3 seria de 5,5 horas. En consecuencia, en un día podríamos hacer seis o siete coladas.

UNIDAD DE CONTROL SEGMENTADA

Procesador segmentado I

Consideremos que queremos llevar esta idea a un procesador: el procesador ejecuta una secuencia de instrucción infinita, por lo que la ganancia sería extraordinaria. Además, la ejecución de una instrucción pasa por una secuencia de fases, por lo que se podrían solapar en el tiempo.

Lo primero que habría que estipular es qué frecuencia de reloj utilizar, ya que este va a ser el indicador de que una fase ha terminado y se debe pasar la fase siguiente al mismo tiempo que se deja libre la fase actual para que una nueva instrucción ocupe su lugar.

Para determinar el tiempo del reloj, hagamos un estudio de los tiempos de instrucción que se pueden ejecutar en un procesador muy simple. Consideremos pues que solo tiene instrucciones de movimiento de datos, operaciones aritméticas simples y saltos condicionales. El problema de esta máquina es que no todas las fases necesitan el mismo tiempo.

En la siguiente tabla podemos ver cómo serían los tiempos de cada una de las instrucciones que tiene este procesador. Vemos que 200 ps es el tiempo requerido para hacer la fase más costosa, por lo que haremos coincidir el reloj de nuestro sistema segmentado con este tiempo.

Clase de instrucción	Búsqueda de la instrucción	Lectura de registros	Operación ALU	Acceso al dato	Escritura en registro	Tiempo total
Almacenar palabra (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Cargar palabra(sw)	200 ps	100 ps	200 ps	200 ps		700 ps
Formato R(add, sub, and, or, slt)	200 ps	100 ps	200 ps		100 ps	600 ps
Salto(beq)	200 ps	100 ps	200 ps			500 ps

UNIDAD DE CONTROL SEGMENTADA

No todas las fases necesitan el mismo tiempo

Por ejemplo, para una instrucción de suma sobre registros:

- La carga de la instrucción ocupa 200 ps.
- La selección de operadores 100 ps.
- La operación de suma 200 ps.
- Y la escritura del resultado en un registro, otros 100 ps.

En total 600ps.

No todas las fases necesitan el mismo tiempo

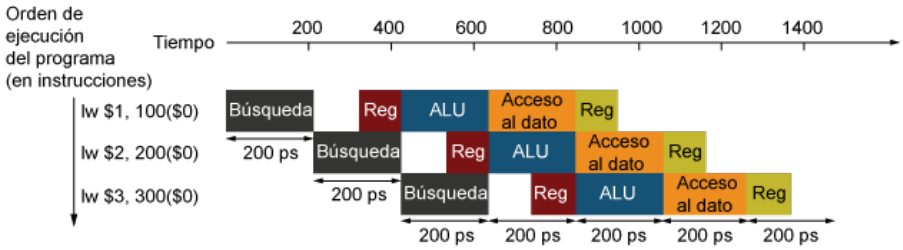
Si queremos hacer una unidad segmentada con este repertorio de instrucciones, ya que en un mismo intervalo de tiempo se están ejecutando todas las fases aunque de instrucciones diferentes, tendremos que ajustar nuestro reloj a la tarea mas larga, la que necesita 200ps.

UNIDAD DE CONTROL SEGMENTADA

Procesador segmentado II

La consecuencia es que hacemos muestro procesador más lento que el original, ya que para la misma instrucción antes solo se requerían 600 ps y ahora, al hacerlos todos de igual duración, 200 ps, el tiempo total es de 800 ps. Pero la peor circunstancia es que no todas las instrucciones necesitan los mismos ciclos de reloj. La carga de datos desde memoria necesita cinco fases, por lo que nuestra máquina también necesitará 5 fases. En definitiva, hay que ajustar nuestro procesador segmentado a la peor de las instrucciones,

Pero la ganancia viene al ejecutar infinitas instrucciones seguidas, ya que el tiempo entre dos instrucciones finalizadas es de 200 ps.



Tal como se aprecia en la imagen, aunque la ejecución de tres instrucciones en una unidad monolítica, consume 800ps*3=2400ps, mientras que nuestra unidad segmentada solo necesita 1400 ciclos, siendo un 71% más rápida.

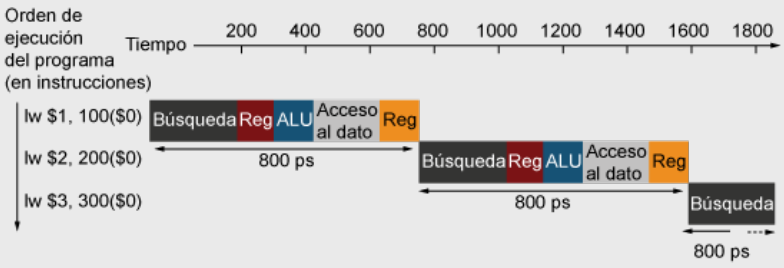
Pongámonos en el peor caso, por ejemplo, la suma (pues los saltos condicionales se tratan de diferente manera). Consideremos el tiempo en hacer cinco sumas:

- La máquina monolítica necesita (200+100+200+100)*5=3000 ps
- La máquina segmentada solo necesita 1000+200+200+200+200=1800ps.

En detalle Resumen

UNIDAD DE CONTROL SEGMENTADA

Tiempo total



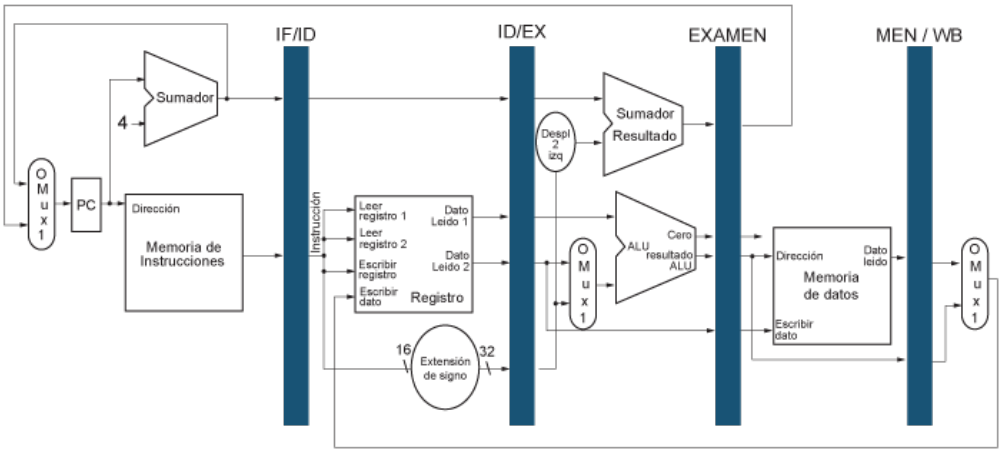
Resumen

La segmentación mejora las prestaciones incrementando la productividad de las instrucciones en lugar de hacerlo disminuyendo el tiempo de ejecución de cada instrucción individual. Esto es justo lo que nos interesa, ya que un programa real tiene que ejecutar miles de millones de instrucción. Y todo esto sin aumentar o replicar la unidad de control (lo que aumentaría costes), solo particionando una unidad de control clásica.

UNIDAD DE CONTROL SEGMENTADA

Construcción de una UC segmentada

La implementación tiene algunos puntos importantes que merecen su estudio, pero analicemos antes cómo puede variar el diseño desde una máquina monolítica a una máquina segmentada.



En la imagen anterior vemos cómo la unidad de control se ha dividido en segmentos, cada uno de los cuales puede hacer una fase de la ejecución completa de una instrucción. Esta unidad se ha dividido concretamente en cinco fases y, dado que ahora la instrucción tiene que avanzar a través de las fases, se necesita un lugar de intercambio.

Así, en la unidad de la imagen se han coloreado de azul los lugares de intercambio, constituidos por buffer, sobre los que cada fase puede dejar los datos para que la fase siguiente los recoja y los procese. Estos buffer también añaden retraso a la ejecución de una instrucción, aunque se considera despreciable dentro del tiempo que se tarda en ejecutar una instrucción completa a través de las fases de la unidad de control.

Lugar de intercambio

Imaginemos nuestro ejemplo de la colada: se necesita una mesa auxiliar de donde ir cogiendo la ropa para ser planchada y otra donde dejar la ropa planchada, para que esta pueda ser cogida y llevada al armario.

UNIDAD DE CONTROL SEGMENTADA

Problemas de las UC segmentadas

La segmentación de una unidad de control, no es un proceso tan sencillo como en el caso de la colada, realmente tiene muchos problemas.

Para empezar, vamos a ver uno que ya hemos enunciado: el problema de que todas las instrucciones tengan que tener el mismo tamaño en ciclos, y el mismo tamaño en número de bits, y que cada ciclo tenga que tener la misma duración.

Estos tres problemas se solucionan en tiempo de diseño del conjunto de instrucciones, por lo que no son un problema en sí mismo, sino una premisa más que hay que cumplir, una premisa que en ocasiones es muy difícil de cumplir.

Pero estas limitaciones no solo son perjudiciales, sino que cumplirlas permite mejorar factores de la máquina, como:

- Que todas las instrucciones se puedan cargar en un solo ciclo de memoria.
- Que todos los operandos de una instrucción deban ser datos en registros.
- Que solo ciertas instrucciones de carga y almacenamiento de datos en memoria permitan el acceso a la misma, de manera que el acceso a datos es simple y se puede efectuar en un ciclo de reloj.

No todos los problemas que origina la unidad segmentada se pueden resolver con un simple rediseño de instrucciones. Algunos de ellos son más críticos y necesitan la incorporación de nuevos algoritmos o hardware especial para solventarlos.

UNIDAD DE CONTROL SEGMENTADA

A este tipo de problemas, le llamaremos *riesgos del pipe* y los más destacados son los siguientes:

- Riesgos estructurales.
- Riesgos de datos.
- Riesgos de control.

Problemas de la segmentación

En lo que queda de tema vamos a analizar los riesgos existentes en la segmentación de una unidad de control y a proponer soluciones para los mismos.

Premisa difícil de cumplir

Por ejemplo, la familia Intel tiene instrucciones cuyo tamaño va desde de 1 byte hasta los 17 bytes.

Intel soluciona este inconveniente haciendo un programa (microprograma, o programa de microinstrucciones) que se ejecuta para dar soporte a esas instrucciones tan largas, haciendo que la instrucción que invoca este microprograma cumpla con las restricciones antes indicadas.

El acceso a datos es simple

Por consiguiente, como solo las instrucciones de carga acceden a memoria, este acceso se hace en la ejecución, por lo que se puede disponer de dos ciclos para el proceso de escritura.

Además, las instrucciones son muy parecidas, por lo que la decodificación es rápida, y a veces se puede solapar con otras acciones como la carga de operandos, reduciendo etapas de la instrucción.

UNIDAD DE CONTROL SEGMENTADA

Riesgos estructurales

El riesgo estructural (*structural hazard*) se aplica para determinar que el hardware no soporta una serie de combinaciones de instrucciones durante el mismo ciclo.

Una manera fácil de verlo, retomando el ejemplo de la colada, es imaginar que tenemos una lavadora-secadora. Está claro que, en tal caso, no podríamos solapar el tiempo de lavado y secado. Estos riesgos normalmente se producen cuando dos etapas tienen necesidad de acceder a memoria y solo tenemos un único banco de memoria, o cuando solo tenemos una ALU y dos ciclos de instrucción necesitan hacer un cálculo.

Estos dos riesgos son reales, por lo que en los sistemas actuales se colocan:

- Dos bancos de memoria distintos, uno con el origen de datos y otro para almacenar los resultados.

- Simplemente una memoria única que tenga más de un puerto de datos y que permita atender dos peticiones simultáneamente.

El caso de las ALU es muy característico. Si analizamos bien el ciclo de carga de instrucción, una de las cosas que hace esta fase es incrementar el contenido del registro PC (Contador de Programa) para hacer que siempre apunte a la siguiente instrucción en ser ejecutada. Para ello necesita que la ALU sume, a este registro, el número de bytes que ocupa una instrucción (normalmente cuatro).

Pero si esta ocupa la ALU del sistema, ¿cómo puede, la parte ejecutiva de la unidad segmentada, hacer la operación que requiere la instrucción en curso?

UNIDAD DE CONTROL SEGMENTADA

La solución es otra vez la replicación, esta vez de una ALU muy específica, que solo suma cuatro, y asignarla permanentemente a la fase de carga de instrucción.

Esquema de la solución

UNIDAD DE CONTROL SEGMENTADA

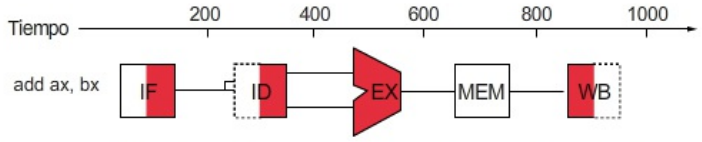
Riesgos de datos

El riesgo de datos (data hazard) ocurre cuando el pipe se tiene que parar porque un paso de ejecución no puede continuar hasta que otro se haya terminado. Suele conocerse como dependencia de datos.

Es decir, supongamos un mini trozo de código, como el siguiente, en el que ax y bx son registros estándar de máquinas Intel.

- add ax, bx => sumar: ax = ax + bx
- sub ax, dx => restar: ax = ax – bx

El problema de este trozo de código es que no se puede calcular la resta de ax – bx, hasta que ax esté totalmente calculado, ya que la instrucción anterior modifica su valor.



La instrucción de suma aún está en el pipe sin terminar, por lo que el valor de ax no está correcto, por lo que la segunda instrucción no puede ser realizada, pues operaría con un valor incorrecto.

Recordemos que para que la instrucción se ejecute, necesita atravesar un pipe de n fases. Supongamos que pueden ser cinco fases y que cada una de ellas hace las tareas de:

- IF: Carga la instrucción, preparándola para la ejecución.
- ID: Decodifica la instrucción, identificando tipo de instrucción y operandos.
- EX: Ejecución de la instrucción o cálculo de la dirección.
- MEM: Acceso a la memoria de datos.
- WB: Escritura del resultado (write back).

1/2

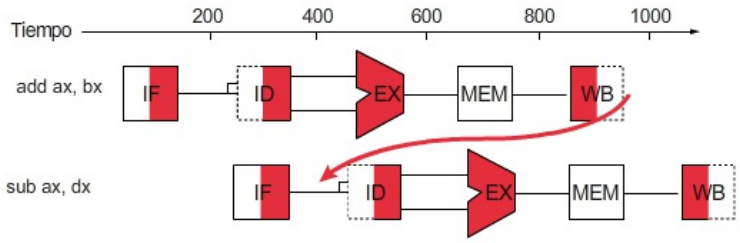
Analicemos este pipe

¿Cuándo está calculado el dato de la instrucción add?

Al final de la fase de ejecución, pero aun así, el dato no puede estar disponible hasta después de la fase de escritura. Por otro lado, la instrucción sub necesita el valor ax.

¿Cuándo se carga este para ser ejecutado?

En la fase de decodificación y carga de operandos. Por tanto, el dato tiene que estar disponible antes de que esta instrucción alcance dicha fase, tal como se puede ver en la imagen, marcada con la línea roja, aunque si hacemos un cronograma de ejecución, veremos que esto no es así.



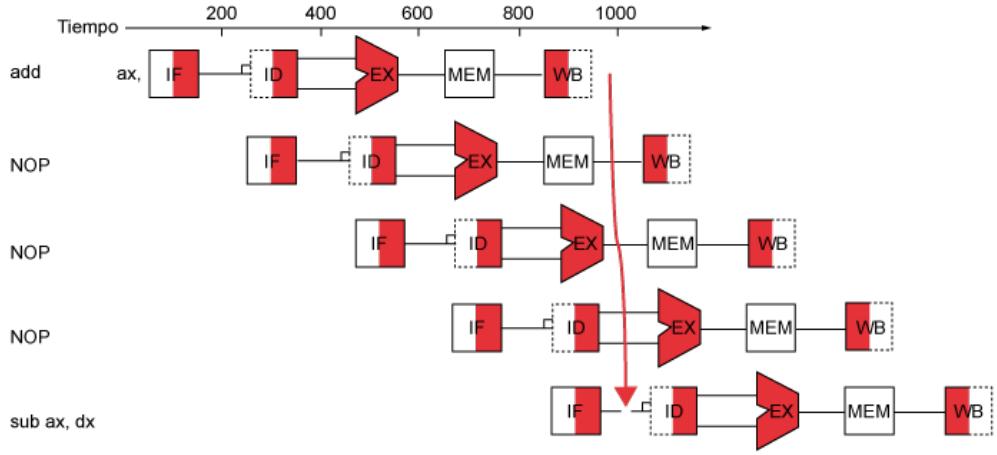
2/2

UNIDAD DE CONTROL SEGMENTADA

Solución a la dependencia de datos I

¿Son estos riesgos solucionables?

La verdad es que hay varios métodos para solucionarlos. El más simple es el que permite solucionarlos por software, es decir, metiendo instrucciones NOP *(Not OPeration)* entre las dos instrucciones hasta que la distancia entre ellas sea la suficiente como para que no haya dependencia de datos. Este es un método poco eficiente pero barato que también se conoce como *bloquear el pipe* o *meter huecos*.



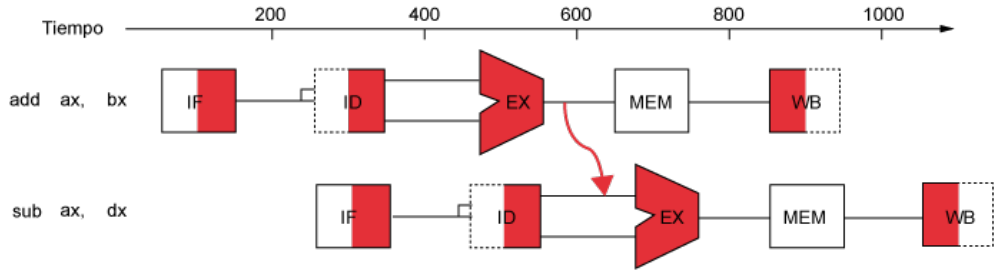
Estas dependencias ocurren mucho más a menudo de lo que desearíamos, por lo que esta solución no es muy eficiente. Por suerte, hay otras mejoras que permiten aumentar la eficiencia de los procesadores segmentados ante la dependencia de datos.

UNIDAD DE CONTROL SEGMENTADA

Solución a la dependencia de datos II

La solución más eficiente para solucionar la dependencia de datos en una unidad de control segmentada es la anticipación de resultados (*forwarding*) o realimentación (*bypassing*).

Esta solución consiste en incorporar caminos de datos que permitan comunicar la salida de la parte ejecutiva de la suma con la salida de la búsqueda de operandos, tal como se puede apreciar, en rojo, en la imagen.



En la figura podemos apreciar cómo sería este adelanto en los datos. Si se hace este puente de datos adelantado, entonces ya no habrá dependencia de datos.

Pero incluso con este adelanto en el bus de dependencias, no todas las dependencias podrían ser solucionadas. Por ejemplo, supongamos que la primera instrucción del ejemplo, en vez de una suma, fuese la carga de un registro de memoria. El registro solo estaría cargado después de que se accediese a la memoria de datos. Por lo tanto, la realimentación tendría que ser después de esta fase, algo imposible, pues la fase de EX sería anterior en el tiempo que la fase de MEM.

Otro de los métodos para corregir la dependencia de datos puede ser la *reordenación de instrucciones*, lo que permite utilizar estas instrucciones nulas para adelantar trabajo.

En definitiva, la solución correcta suele ser una mezcla de ambas soluciones.

UNIDAD DE CONTROL SEGMENTADA

Riesgo de control

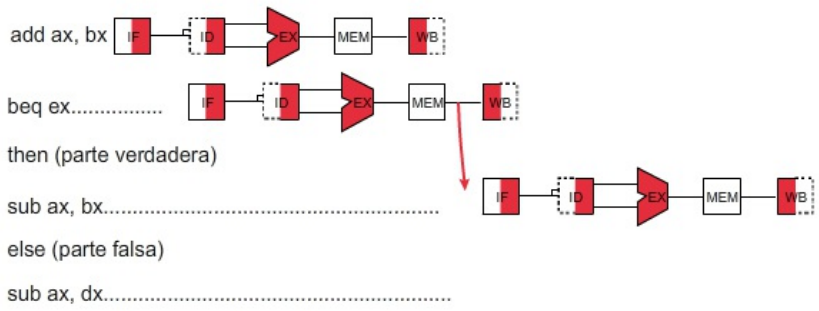
Los riesgos de control (*control hazards*), surgen con las instrucciones de salto condicional.

Imaginémonos una secuencia de código que implementa un "if" de un lenguaje de alto nivel. Este troco de código, dependiendo del resultado de una comparación, hay que hacer un salto para ejecutar la parte "else" o por el contrario, hay que continuar con la ejecución, ya que no hay salto y, por tanto, se ejecuta la parte "then".

El problema de aquí es, ¿en qué momento sabemos el resultado de la comparación del "if"?, lo más pronto posible es después de la fase ejecutiva, pero es dato, una vez más, no está disponible hasta que la fase de escritura haya terminado. Por consiguiente, no sabemos cuál es la instrucción que hay que meter en el buffer de instrucciones hasta que la comparación atraviese todo el pipe.

Una vez más, la solución más simple, es detener el pipe, introduciendo huecos, hasta que estemos en condiciones de determinar cuál es la nueva instrucción a cargar en el pipe.

Tal como se aprecia en la imagen donde la instrucción a ejecutar es la parte "then" de la condición.



1/3

Otras soluciones no tan derrochadoras de tiempo de procesador son la predicción de saltos (branch prediction). Es muy simple, consiste en tomas una decisión (que puede ser aleatoria, o siempre la misma) y actuar en consecuencia. Si se ha tenido éxito con la predicción, no se toman medidas en el pipe pero, si por el contrario, se ha equivocado en la decisión, hay que resetear el pipe completo y empezar con la carga de instrucciones correcta.

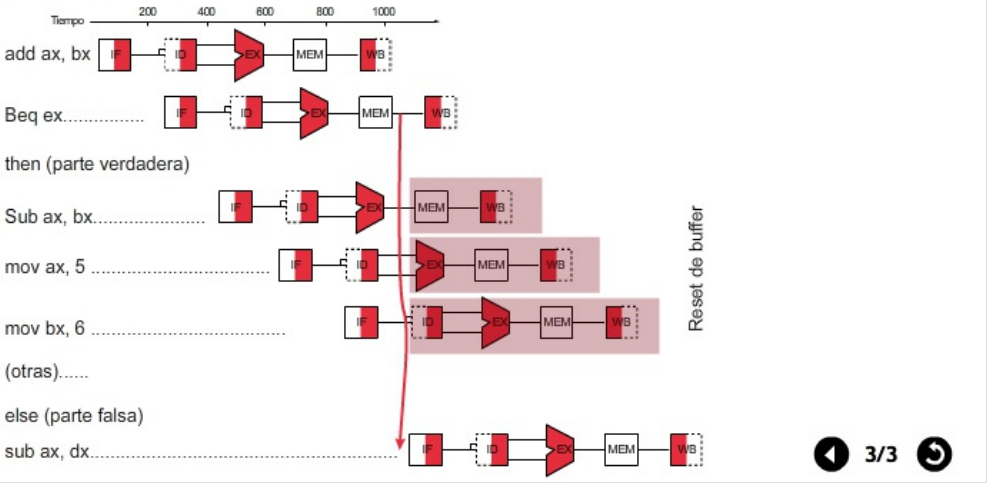
Esta solución, no es perfecta, pero no se pierde nada, y sí se puede ganar si se ha tenido éxito en la decisión. Además, el coste para añadir esta solución es muy simple y barato. Si además echamos mano de la estadística, que dice que en más del 80% de las instrucciones de salto, siempre se realiza el salto, los beneficios pueden ser muy buenos a poco coste.

2/3

UNIDAD DE CONTROL SEGMENTADA

En la imagen, se muestra el funcionamiento de una función con predicción de "nunca salto", como se comete un error, hay que resetear el buffer y comenzar de nuevo con la parte "else".

Una mejora de este algoritmo se basa en los predictores por hardware, que intentan analizar el tiempo de sato y un histórico de las últimas operaciones de salto, para determinar con mayor éxito la predicción, aumentando el índice de éxito al 90%. Al igual que el caso anterior, si la predicción es errónea, se borra el pipe se comienza con la carga de la nueva instrucción.

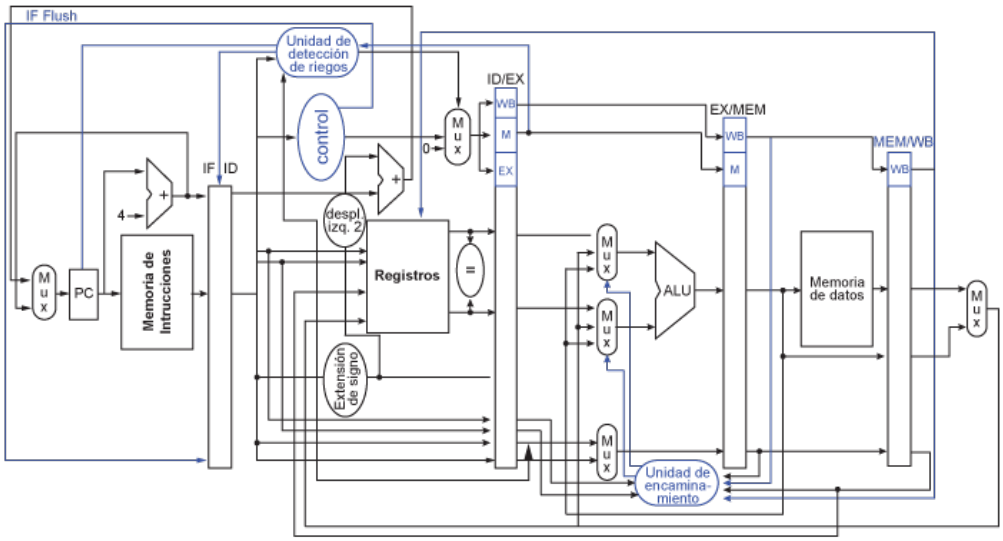


UNIDAD DE CONTROL SEGMENTADA

Resumen de la visión general del pipe

Las segmentaciones, o las unidades de control segmentadas (pipe), a diferencia de otros métodos de paralelismo que consisten en la réplica de unidades funcionales con el aumento de costes, explotan el paralelismo entre las instrucciones de un flujo de instrucciones secuenciales.

En detalle Descripción de la imagen



UNIDAD DE CONTROL SEGMENTADA

Descripción de la imagen

En la imagen puedes apreciar la estructura de una unidad de control segmentada sobre la que se han añadido la mayor parte de las mejoras para evitar riesgos que antes hemos visto.

En ella se ha añadido una unidad de control auxiliar, encargada de detectar los riesgos, que actúa sobre (entre otros) los multiplexadores de datos que hay en la unidad de ejecución a la entrada de la ALU, lo que permite la realimentación adelantada de datos procedentes de las etapas siguientes.

Además, actúa sobre los buffer intermedios a las fases, lo que permite resetearlos para poder empezar con un conjunto de instrucciones nuevo en los saltos condicionales erróneos (riesgos de control). Se marcan en azul todas las nuevas señales que hay que generar para la modificación desde la primera unidad de control segmentada hasta esta.

UNIDAD DE CONTROL SEGMENTADA

Resumen

En este tema, hemos visto cómo mejorar la eficiencia de un procesador sin la necesidad de replicar componentes, que es la técnica más utilizada para conseguir un alto rendimiento.

Hemos comprobado cómo funciona la unidad de control segmentada, detallando sus problemas y virtudes.

Para ello, hemos analizado los tipos de riesgos existentes en la construcción de una unidad de control segmentada.

Hemos ido dando solución a estos riesgos, tanto en el caso de los riesgos estructurales, como en el de los riesgos de datos y el de los riesgos de control.

Es importante quedarnos con la idea de cuáles son los tiempos de ejecución de una instrucción, y porque el beneficio de una unidad de control segmentada ocurre cuando hay un número grande de instrucciones secuenciales a ejecutar por un único procesador.