PRÁCTICA 8: JERARQUÍA DE MEMORIA IV

ESTUDIO DEL RENDIMIENTO DE LA JERARQUÍA DE MEMORIA EN CACHÉS UNIFICADAS Y SEPARADAS

ARQUITECTURA DE COMPUTADORES. 2º CURSO

1. OBJETIVOS

Con esta práctica se pretende que el alumno profundice en el estudio del comportamiento de las memorias caché teniendo en cuenta, además de los accesos a datos, los accesos a instrucciones que se producen durante la ejecución de un programa específico. En este contexto, el alumno estudiará el impacto que tiene sobre el rendimiento de la jerarquía de memoria el empleo de memorias caché unificadas y separadas.

2. PREPARACIÓN

Antes de acudir a la sesión de laboratorio el alumno debe:

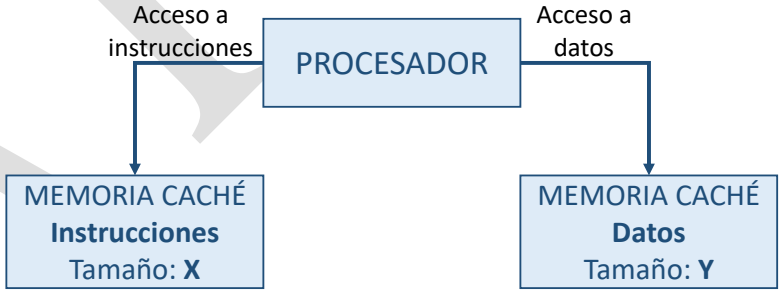
- **Repasar los contenidos de teoría del tema de jerarquía de memoria con especial atención a las ventajas e inconvenientes de las memorias caché unificadas y separadas.**
- **Leer y asimilar los contenidos que se exponen en el siguiente apartado de este boletín.**
- **Realizar el estudio previo de forma manuscrita. Este cuestionario deberá ser entregado al profesor al comienzo de la sesión.**

3. INTRODUCCIÓN TEÓRICA.

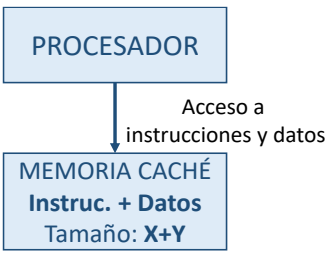
3.1. TIPOS DE MEMORIA CACHÉ.

Como se estudió en el tema de jerarquía de memoria, la memoria caché puede clasificarse en función del tipo de información que contiene: cachés unificadas y separadas. De esta clasificación da lugar a dos formas distintas de configurar el nivel de las memorias caché:

Nivel con Memorias caché separadas: en el nivel se disponen de dos memorias, una dedicada a almacenar datos y otra las instrucciones del programa. Esta configuración se caracteriza por duplicar el ancho de banda del acceso al nivel permitiendo así accesos simultáneos a memoria.



Nivel con Memoria caché unificada: el nivel de la memoria caché dispone de una memoria que va a contener tanto los datos como las instrucciones del programa. En este caso, puesto que no está delimitado el espacio para instrucciones y datos, el aprovechamiento de la memoria es mayor.



3.2. TRAZA DE LOS ACCESOS A MEMORIA

Para estudiar el comportamiento de estas memorias (unificadas y separadas) será necesario previamente generar una traza de accesos a partir de la ejecución de un programa especificando cada tipo de acceso: lectura de una instrucción, o lectura o escritura de un dato.

Para ilustrar la generación de una traza de accesos va a utilizarse un programa simple como el siguiente:

```
.text 0x06543210
     lui $1,0x1122          0x6543210
     ori $1,$1,0x3344       0x6543214
     addi $2,$0,10          0x6543218
otro:lw $3,0($1)            0x654321C
     sw $3,100($1)          0x6543220
     addi $1,$1,4           0x6543224
     addi $2,$2,-1          0x6543228
     bnez $2,otro           0x654322C
```

En primer lugar, deberá determinar la dirección de memoria desde la que se va a leer cada instrucción del programa. Puesto que las instrucciones se almacenan de forma consecutiva en memoria, la primera instrucción se encontrará en la dirección de carga del programa (0x06543210 en el ejemplo) y la dirección de las instrucciones posteriores se obtiene sumándole sucesivamente el tamaño de la instrucción (4 bytes en el caso del MIPS32) a la dirección de carga. A la derecha del programa ejemplo anterior se muestra la dirección donde comienza cada instrucción del código.

Como se estudió en el tema de procesadores, durante la ejecución de la etapa **IF** (búsqueda de la instrucción) el procesador solicitará la lectura en memoria de los 4 bytes que definen la instrucción. Eso se traducirá en un acceso a la caché de instrucciones en el caso de cachés separadas.

Las instrucciones de carga (como son lb, lbu, lh, lhu, lw…) producen una lectura de datos durante la ejecución de su etapa **MEM.** De forma análoga, las instrucciones almacenamiento (sb, sh, sw…) producen una escritura de datos en su etapa **MEM**. Por tanto, los accesos a datos sólo los llevan a cabo las instrucciones de carga y almacenamiento en su etapa MEM, que se traducirá en accesos a la caché de datos en caso de cachés separadas. Téngase en cuenta que la dirección de acceso a datos suele conocerse únicamente en tiempo de ejecución (salvo que utilice el registro $0, etc.) ya que las instrucciones de carga y almacenamiento utilizan el modo de direccionamiento registro base más desplazamiento, por lo que en muchos casos será necesario realizar una traza de ejecución para determinar la traza de accesos a datos. Por ejemplo. la instrucción *lw* del programa anterior lee los datos a partir de la dirección dada por el registro $1, el cual se incrementa en 4 cada iteración, lo que provoca lecturas en las direcciones 0x11223344, 0x11223348,…0x11223368. La instrucción *sw* realiza escrituras con un patrón de accesos similar a *lw* a partir de la dirección $1+100: 0x112233A8, 0x112233AC,… 0x112233CC.

Para determinar el orden en el que se producen los accesos a memoria deberá tenerse en cuenta, además del hecho de que **todas leen la instrucción de memoria en la etapa IF (acceso a instrucciones) y que las de carga y almacenamiento acceden a memoria en su etapa MEM (acceso a datos), si el procesador es secuencial o segmentado**.

3.2.1 Trazas en un procesador secuencial

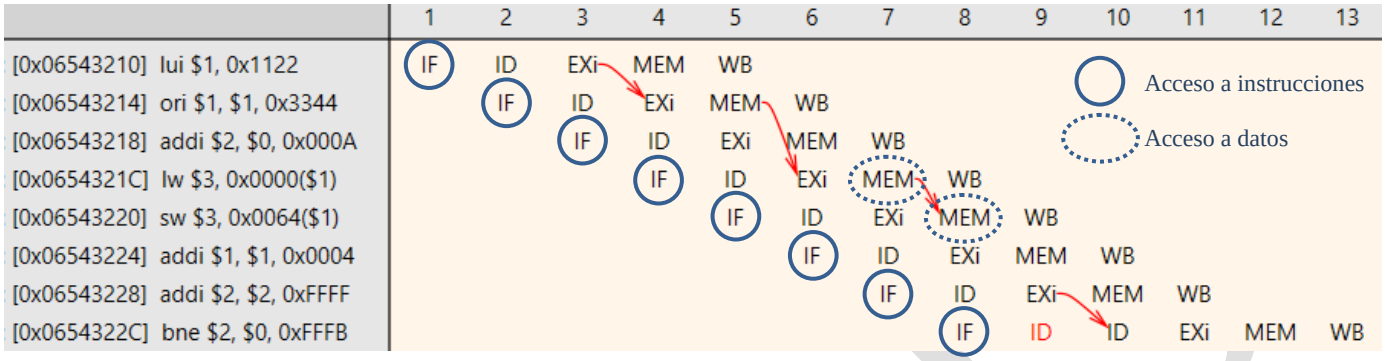
En un procesador secuencial, puesto que sólo hay una instrucción en ejecución, el orden de los accesos es evidente por lo que no requiere realizar cronograma alguno para obtenerlo. Por ejemplo, la ejecución del programa anterior en un MIPS secuencial, provocaría la siguiente secuencia de accesos:

Instrucción	Etapa	Tipo de Acceso	Tipo de Operación	Dirección
`lui $1,0x1122`	IF	Instrucciones	Lectura	0x6543210
`ori $1,$1,0x3344`	IF	Instrucciones	Lectura	0x6543214
`addi $2,$0,10`	IF	Instrucciones	Lectura	0x6543218
`lw $3,0($1)`	IF	Instrucciones	Lectura	0x654321C
`lw $3,0($1)`	**MEM**	**Datos**	**Lectura**	**0x11223344**
`sw $3,100($1)`	IF	Instrucciones	Lectura	0x6543220
`sw $3,100($1)`	**MEM**	**Datos**	**Escritura**	**0x11223344 + 1000 = 0x11223348**
`addi $1,$1,4`	IF	Instrucciones	Lectura	0x6543224

3.2.2 Trazas en un procesador segmentado con memorias caché <u>separadas</u>.

En un procesador segmentado, el orden de los accesos es alterado respecto a la versión secuencial al encontrarse varias instrucciones en ejecución. En estos casos, la forma más adecuada de extraer la traza de accesos es **realizar el cronograma de ejecución teniendo en cuenta que puede realizarse un acceso a instrucciones y a datos simultáneamente.**

A continuación, se muestra el cronograma y la traza de accesos en un procesador **segmentado** con bypass y con memorias **separadas** para instrucciones y datos:



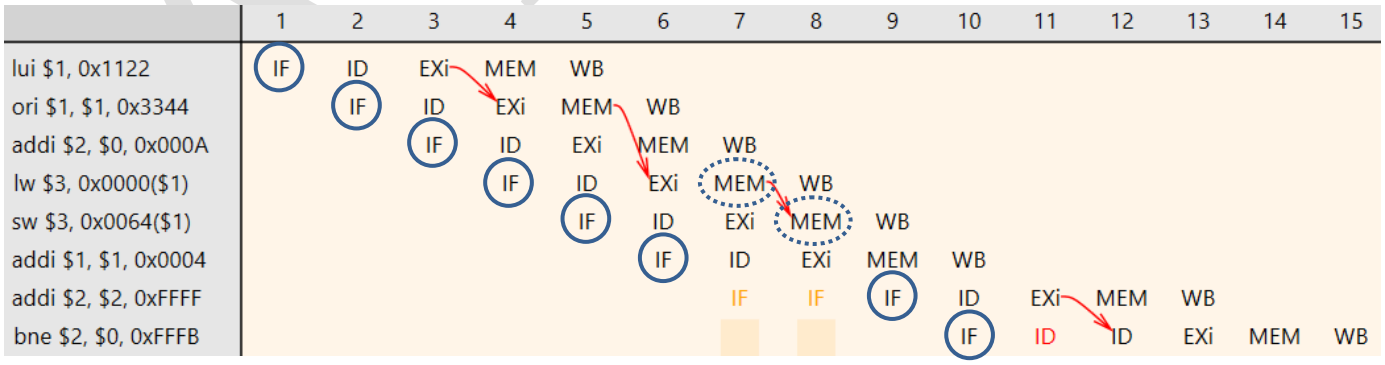
Ciclo	Instrucción	Acceso	Operación	Dirección	Instrucción	Acceso	Operación	Dirección
1	lui	Instrucción	Lectura	0x06543210				
2	ori	Instrucción	Lectura	0x06543214				
3	addi	Instrucción	Lectura	0x06543218				
4	lw	Instrucción	Lectura	0x0654321C				
5	sw	Instrucción	Lectura	0x06543220				
6	addi $1…	Instrucción	Lectura	0x06543224				
7	addi $2…	Instrucción	Lectura	0x06543228	lw	Dato	Lectura	0x11223344
8	bne	Instrucción	Lectura	0x0654322C	sw	Dato	Lectura	0x112233A8

Como se aprecia en los ciclos 7 y 8 del cronograma anterior, se producen accesos simultáneos a instrucciones y datos gracias a la utilización de cachés separadas.

3.2.3 Trazas en un procesador segmentado con memoria caché <u>unificada</u>.

Como se estudió en el tema anterior, cuando se dispone de una memoria común para instrucciones y datos (cachés unificadas), puede producirse bloqueos estructurales puesto que no puede accederse simultáneamente a instrucciones y datos (véase las transparencias sobre ejemplos de bloqueo estructural del tema de procesadores). Lo habitual en estos casos es que una instrucción posterior no pueda realizar la etapa IF si una instrucción de carga o almacenamiento anterior está accediendo a MEM en el mismo ciclo que la instrucción posterior.

Como se muestra en el siguiente el cronograma considerando una memoria caché unificada, en la ejecución se producen bloqueos estructurales en addi debido a que las instrucciones de memoria lw y sw.



3.3. CÁLCULO DE LA FRECUENCIA DE FALLOS.

El cálculo de la frecuencia de fallos en cachés separadas se realiza de forma similar al caso con caché unificadas:

$$FF_{Total} = \frac{\text{Nº Fallos}}{\text{Nº Accesos}} = \frac{\text{Nº Fallos}_{Instruc} + \text{Nº Fallos}_{Datos}}{\text{Nº Accesos}_{Instruc} + \text{Nº Accesos}_{Datos}} = \%Instruc \cdot FF_{Instruc} + \%Datos \cdot FF_{Datos}$$