Enunciado formalizado, de acuerdo con lo explicado en clase, el día V 17 de octubre

PROYECTO PROGRAMADO (valor: 26%)

Simulación de un procesador MIPS para enteros con ejecución de hilos de un mismo proceso

CAMBIO: CÓDIGO DE OPERACIÓN PARA LL DE 0 A 50 Y PARA EL SC DE 1 A 51 (4 DE NOVIEMBRE 2014)

Fecha entrega: **Miércoles 12 de Noviembre**

I. Descripción del proyecto y del procesador a simular:

Diseñar y programar **un procesador MIPS para enteros, de un núcleo,** en el cual debe implementarse el **paralelismo en el nivel de instrucciones** (**pipeline de 5 etapas visto en clase**). Los saltos se resolverán en **ID** (o sea, se implementará el "pipeline" MIPS "mejorado")

II. Detalles importantes:

1. **Cada etapa del pipeline debe ser implementada con un hilo de ejecución.**
2. Este procesador simulado recibirá para ejecutar **varios hilos MIPS que se supone forman parte de un único proceso**, por lo que entre ellos comparten memoria.
3. La **asignación de la CPU** entre estos hilos, se hará utilizando un algoritmo de planificación por turno rotatorio **(round robin**) con un **quantum de X** ciclos de reloj para cada "hilo". Para efectos de la simulación, el valor del quantum será dado como un dato inicial por el usuario.
4. **La sincronización** entre los hilos MIPS que ejecutará su procesador **se realiza solo mediante semáforos binarios** (mutex o locks) los cuales permiten a un hilo como máximo el ingreso a una sección crítica (exclusión mutua). Estos mutex se implementarán tan solo usando dos instrucciones que **agregaremos** al MIPS: **Load linked (LL) y el Store Conditional (SC)**- se describen más adelante.
5. Los hilos MIPS que ejecutará el procesador ya vendrán **codificados en "lenguaje de máquina"** de acuerdo con los formatos MIPS, en un archivo de texto con una instrucción por línea, pero NO en código binario, sino decimal.
6. Cada uno de estos hilos deberá copiarse en el vector que simula la **memoria para instrucciones** (el mismo vector para instrucciones, utilizado para la TP2, pero con mayor dimensión-ver punto 10 de esta sección**.** No se utilizará una caché de instrucciones, por lo que el "IF" siempre traerá la instrucción de dicho vector por lo que **nunca se tendría un fallo de caché de instrucciones**.
7. Como notará más abajo, cada instrucción está compuesta por 4 números en decimal, por lo que será muy simple simular las direcciones de memoria múltiplo de 4 para las instrucciones ya que ***cada inicio*** de instrucción estaría en un subíndice del vector que es "múltiplo de 4".
8. En cuanto al almacenamiento de datos, se tendrá una **caché de datos** y se mantendrá el **vector de datos** de la TP2 la cual cumplirá el rol de **memoria principal** pero se debe aumentar su dimensión-ver punto 10 de esta sección .
9. **La caché de datos** tendrá capacidad para **8 bloques**, cada uno de **4 palabras**. Su estrategia para asignación de bloques es: **Mapeo directo.** La **estrategia para "hit" de escritura** es **"WRITE BACK"** (escribe solo en caché, y la actualización a memoria se hace **cuando** se vaya a **reemplazar el bloque**. La estrategia para fallo de escritura será **"WRITE ALLOCATE**", es decir se sube el bloque a caché.
10. En cuanto al tamaño de los vectores para la memoria de instrucciones y la de datos:
    - Para instrucciones el vector debe tener espacio para almacenar **192** instrucciones (cada una de 4 bytes, pero que en realidad las estamos implementando con 4 enteros), es decir **768 enteros** ("48 bloques de 4 palabras cada uno")
    - Para datos se necesita que tenga espacio para 832 enteros (**832 entradas**) para un total de 208 bloques, del bloque 48 al 255 , **dirección inicial para datos: 768**, dirección para el último dato: **4092**.
    - Note que se están manejando **dos estructuras** para simular la memoria principal, una para las instrucciones, y otra para la parte de memoria que contiene la memoria compartida entre los hilos MIPS que correrán. Sin embargo, **como la memoria es una sola**, se supondrá que del bloque 0 al 47 se almacenarán instrucciones, y del bloque 48 al 255 los datos. Es por ello que las direcciones deben manejarse acorde con eso. Ningún dato puede guardarse en una dirección menor a la 768, que es el byte en donde comienza el bloque 48.

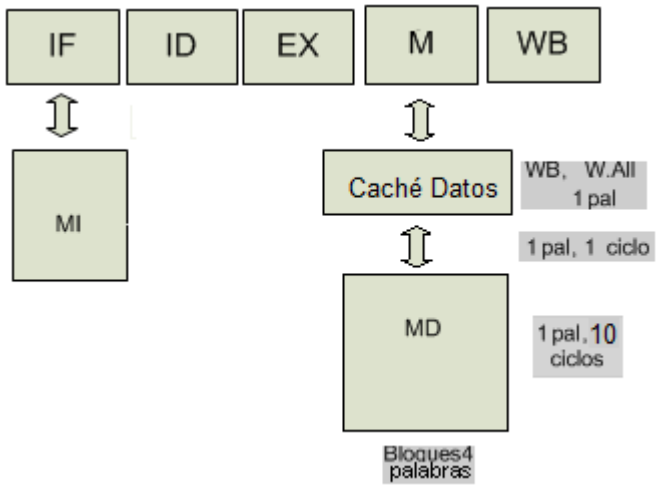
- **LA MEMORIA DE DATOS SE DEBE INICIALIZAR CON EL VALOR 1 EN CADA UNA DE SUS PALABRAS. Esto es para simular que todos los candados fueron puestos por el "hilo principal"**

11. Para contabilizar los **ciclos de retraso** cuando se produce un fallo en la caché de datos, se supondrá lo siguiente (aunque esto no se debe simular, tan solo se explica para comprender el tiempo que se tarda):

Como los bloques son de 4 palabras, y para subir un bloque a memoria (o para escribir un bloque a memoria) se hace palabra por palabra, el total de ciclos de reloj que se tarda es:
4 * (lo que se tarda en leer o escribir un palabra de o en memoria)=
= 4 * (transmisión de dirección de mem. en el bus +latencia de la memoria para buscar la palabra + transmisión en el bus a caché de la palabra solicitada en el bus) =

4 * (1 + 10 +1) = **48 ciclos de reloj**

Lo mismo se tarda para **escribir un bloque de caché a memoria**.



12. En el proyecto se implementarán las siguientes instrucciones MIPS, las sombreadas con gris no fueron implementadas para la TP2, por lo que ahora se deben incluir **:**

INSTRUCCIÓN			CODIFICACIÓN				
Operación	Operandos	Acción	0-5	6-10	11-15	16-20	21-31
			Cód. Operación (DECIMAL)	Rf1	Rf2-Rd	Rd	inmediato
DADDI	RX, RY, #n	Rx ← (Ry) + n	001000 - 8	Y	X		n
DADD	RX, RY, RZ	Rx ← (Ry) + (Rz)	100000 - 32	Y	Z	X	
DSUB	RX, RY, RZ	Rx ← (Ry) - (Rz)	100010 - 34	Y	Z	X	
DMUL	RX, RY, RZ	Rx ← (Ry) * (Rz)	011000 - 12	Y	Z	X	
DDIV	RX, RY, RZ	Rx ← (Ry) / (Rz)	011010 - 14	Y	Z	X	
LW	RX, n(RY)	Rx ← M(n + (Ry))	100011 - 35	Y	X		n
SW	RX, n(RY)	M(n + (Ry)) ← Rx	101011 - 43	Y	X		n
BEQZ	RX, ETIQ	Si Rx = 0 SALTA	000100 - 4	X	0		n
BNEZ	RX, ETIQ	Si Rx ≠ 0 SALTA	000101 - 5	X	0		n
LL	RX, n(RY)	Rx ← M(n + (Ry)) y establece el link register: RL = n + (Ry)	50	Y	X		n
SC	RX, n(RY)	If RL = n + Ry Then M(n + (Ry)) ← Rx y mantiene el 1 en Rx Else pone un 0 en Rx y no escribe en memoria	51	Y	X		n
JAL	n	R31←PC, PC← PC+n	000011 - 3				n
JR	RX	PC ← (Rx)	001000 - 2	X	0		0
"FIN"		Detiene el programa	111111 - 63	0	0		0

13. Los conflictos de datos se resolverán deteniendo a la instrucción en ID hasta que todos sus operandos estén escritos en el registro correspondiente. Es decir, **no hay "forwarding".**
14. Para los conflictos de control (saltos) se detendrá el ingreso de instrucciones hasta que no se resuelva el salto (condicional o no). O sea, **no hay predicción para branches.**
15. Debe simularse **el reloj del procesador** de manera que cada "TIC" de reloj ocurrirá cuando TODAS las etapas del pipeline hayan finalizado lo que debían hacer en ese ciclo de reloj.
16. **Mientras corre la simulación debe verse en pantalla** el **valor del reloj**

17. **Al finalizar la simulación** debe desplegarse en pantalla
    - **El contenido de la memoria** (las 832 direcciones y su contenido)
    -el contenido de la **caché de datos**
    - el **contenido de los 32 registros** del procesador de cada uno de los hilos que corrió
    - la **cantidad de ciclos** que tardó cada hilo

III. Forma de realizar y entregar el proyecto:

1. Debe trabajarse en el mismo grupo de la tarea programada 2.
2. Al igual a como se indicó para la TP2, el programa de simulación se puede realizar en el **lenguaje de alto nivel que el grupo de trabajo escoja**, pero este lenguaje debe permitir manejar muy bien el paralelismo en el nivel de hilos. El programa debe utilizar arquitecturas paralelas, es decir, es **indispensable que se diseñe utilizando hilos que realmente puedan correr en paralelo** y logren no solo aprovechar el paralelismo que permiten los procesadores de al menos doble núcleo sino que también permitan hacer una simulación más "real".
3. Debe hacerse la **correcta sincronización entre hilos**.
4. **INDISPENSABLE** buena documentación interna la cual debe dejar muy claros los algoritmos utilizados para resolver cada problema.
5. **El día** de entrega o antes, se envía por correo electrónico a arqui.g1@gmail.com ó arqui.g2@gmail.com según su grupo lo siguiente:
    a. Descripción a grandes rasgos de la **lógica de la simulación** (use un diagrama de bloques-o de actividades)
    b. Lista de los **hilos** que se usaron para la simulación, describiendo la forma en la que se logró toda la **sincronización requerida** entre ellos y el uso de estructuras de datos (incluyendo el hilo principal y la forma de cambiar de ciclo de reloj) **sincronización** utilizados
    c. El fuente y el **ejecutable** del programa (debe hacerse el ejecutable si aplica)
    d. Un **manual muy claro de instalación**
    e. Lista de **problemas no resueltos** al tiempo de entrega e ideas sobre su solución.
    f. Una **nota** para cada uno de los miembros del grupo de trabajo de acuerdo con el desempeño del integrante en el desarrollo del proyecto (o a 100)

Importante: en esta documentación se deben hacer las descripciones breves pero claras, y en lo posible ayudándose con diagramas

IV. CALIFICACIÓN: La nota del proyecto se calcula así:

Puntaje:	Documentación	10%	
	Diseño y lógica	50%	(a revisar en doc, en código, y en corrida)
	Programación correcta	40%	

La nota para cada estudiante por el proyecto se calcula así:

Nota del proyecto * Nota puesta por el grupo según su desempeño/100