CC4301 Arquitectura de Computadores – Examen – Otoño 2019 – Prof.: Luis Mateu

Pregunta 1

Parte a.- La figura muestra ciclo por ciclo la ejecución de varias instrucciones en un procesador con un pipeline de 4 etapas (i) ¿En qué instantes se realiza register bypassing y por qué? (ii) Complete el diagrama (iii) Rehaga el diagrama para una arquitectura superescalar de grado 2 con 5 etapas (F, D, A, E y S).

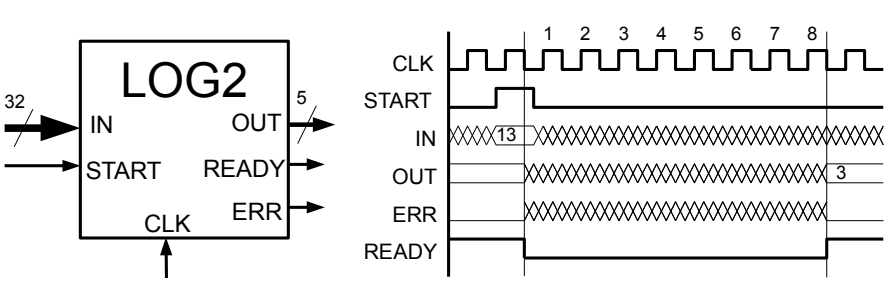
		1	2	3	4	5	6	7	8	9	10	...
a.-	ADD R5, R6, R7	F	D	E	S							
b.-	SUB R8, 4, R9		F	D	E	S						
c.-	ADD R9, 4, R10			F	D	E	S					
d.-	LOAD [R10+8], R11				F	D	E	M	M	M	S	
e.-	ADD R5, 20, R12					.	.	.				
f.-	SUB R11, 4, R13											
g.-	ADD R1, R2, R3											

Parte b.- La figura muestra un extracto del contenido de un *cache* de 4 KB (2^{12} bytes) de 1 grado de asociativad con 256 líneas de 16 bytes. El computador posee un bus de direcciones de 16 bits. Por ejemplo en la línea 0f (en hexadecimal) se almacena la línea de memoria que tiene como etiqueta 20f (es decir, la línea que va de la dirección 20f0 a la dirección 20ff).

línea cache	*etiqueta*	*contenido*
c9	6c9	
0f	20f	
8b	78b	

Un programa accede a las siguientes direcciones de memoria: 20f8, 48b4, 90f0, 20f0, 6c90, 90f8, 30f0. Indique qué accesos a la memoria son aciertos en el cache, cuales son desaciertos y rehaga la figura mostrando el contenido final del cache.

Pregunta 2



I. El circuito LOG2 de la figura recibe como entrada un número IN de 32 bits sin signo y calcula eficientemente la posición del bit en 1 más significativo de IN. LOG2 inicia el cálculo cuando START se pone en 1. La línea READY se va a 0 mientras se realiza el cálculo. El término del cálculo se señala llevando la línea READY a 1 y entregando los 5 bits del resultado en OUT. La línea ERR es 0 a menos que IN no posea ningún bit en 1, en cuyo caso ERR es 1. Las salidas permanecen constantes hasta que se solicite un nuevo cálculo llevando START a 1.

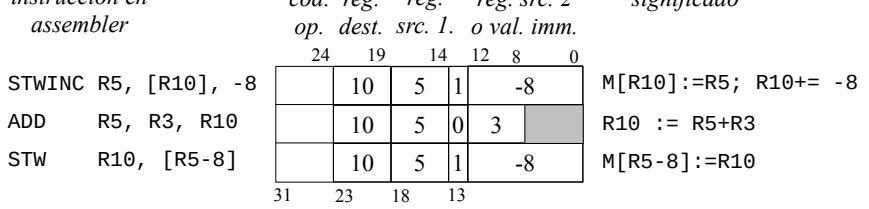
Considere el circuito LOG2 como un dispositivo de E/S. Implemente una interfaz de LOG2 mapeada en memoria para M32. Su interfaz debe ser tal que al escribir un valor *x* en el puerto de E/S con dirección 0xff000000 el circuito LOG2 inicie el cálculo con IN=*x*. Cuando se lee de ese mismo puerto Ud. debe entregar en el bus de datos un valor *z* tal que OUT corresponde a los 5 bits menos significativos de *z* (D4-D0), READY es el sexto bit (D5) y ERR el séptimo (D6). El resto de los bits deben quedar en 0 (D31-D7).

II. Escriba en C la función `int log2(int x, int *res)` que utiliza la interfaz de la parte *I* para calcular eficientemente la posición del bit más significativo de *x*. Si hay algún bit en 1 en *x* la función log2 debe retornar 0 y entregar la posición del bit más significativo en ***res*. De lo contrario debe retornar 1. No olvide realizar un ciclo de *busy-waiting* para esperar a que READY se coloque en 1.

III. Se desea implementar la instrucción *store word and increment* que realiza lo siguiente:

```
STWINC R5, [R10], 8  # M[R10]:= R5; R10 += 8
```

La instrucción recibe 3 parámetros. El primero es el registro que contiene el valor que se almacerá en memoria (R_{s1}). El segundo es el registro que contiene la dirección de memoria que se escribirá (R_d). Y el tercero es un registro o valor inmediato que se sumará al registro que contiene la dirección. La figura muestra la codificación de STWINC en comparación con STW y ADD.



Implemente esta instrucción en M32 indicando ciclo a ciclo las transferencias entre registros y las señales de control necesarias para realizar esas transferencias.