

## E.P.S.- UAM.- Arquitectura e Ingeniería de Computadores, Septiembre 2003

**C1.-** A continuación se muestra un posible código para un contador de 4 bits con reset asíncrono y carga asíncrona con su correspondiente testbench.

<pre>library IEEE; use IEEE.std_logic_1164.all; use IEEE.std_logic_unsigned.all;  entity contador is   port( clk: in STD_LOGIC;         load: in STD_LOGIC;         rst: in STD_LOGIC;         d: in STD_LOGIC_VECTOR (3 downto 0);         q: out STD_LOGIC_VECTOR (3 downto 0)   ); end contador;  architecture examen of contador is   signal cuenta :     std_logic_vector(3 downto 0);  begin    -- cuenta en cada flanco de subida   process(clk)   begin     if load='1' then       cuenta &lt;= d;     elsif rising_edge(clk) then       cuenta &lt;= cuenta+'1';       q &lt;= cuenta;     end if;   end process;    -- reset asíncrono   process(rst)   begin     if rst='1' then q &lt;= "0000"; end if;   end process;  end examen;</pre>	<pre>library IEEE; use IEEE.std_logic_1164.all; use IEEE.std_logic_unsigned.all;  entity contador_tb is end contador_tb;  architecture examen of contador_tb is    component contador     port( clk : in std_logic;           rst : in std_logic;           q : inout std_logic_vector(3 downto 0));   end component;    signal clk : std_logic;   signal rst : std_logic;   signal q : std_logic_vector(3 downto 0);  begin    UUT : contador port map     (clk =&gt; clk, rst =&gt; rst, q =&gt; q);    process   begin     -- Inicialización de señales     d &lt;= "0010"; load &lt;= '0'; clk &lt;= '0';     rst = '0'; wait for 10 ns;     -- Testbench     load &lt;= '1'; wait for 10 ns;     rst &lt;= '1'; clk &lt;= '1'; wait for 10 ns;     rst &lt;= '0'; clk &lt;= '0'; wait for 10 ns;     clk &lt;= '1'; wait for 10 ns;     wait;   end process;  end examen;</pre>
---	--

a). ¿Es correcta la lista de sensibilidad de cada proceso? ¿Por qué?

b) ¿Cuánto valdrá la salida q del contador tras ejecutar el testbench?.

b1) "0000"      b2) "0001"      b3) "0010"      **b4) "000X"**

c) Al simular el código se comprueba que el contador no funciona bien. ¿Por qué?. ¿Cuál sería una posible solución para arreglarlo de una manera sencilla?

**C2.-** En un Sistema RISC con arquitectura Harvard y formato de instrucción de 32 bits, se elige un tamaño para la cache de instrucciones de 32 Bytes y un tamaño de bloque de 8 Bytes. El sistema ejecuta como programa el siguiente bucle infinito.

0000 0000	Instrucción1	
0000 0004	Instrucción2	
0000 0008	Instrucción3	
0000 000C	JMP 20	; Instrucción 4 es un salto a la dirección de memoria 20
0000 0010	Instrucción5	
0000 0014	Instrucción6	
0000 0018	Instrucción7	
0000 001C	Instrucción8	
0000 0020	Instrucción9	
0000 0024	JMP 0	; Instrucción10 es un salto a la dirección de memoria 0

En estas condiciones, se pretende seleccionar la organización de la cache de instrucciones, entre todas las que sean posibles, que resulte mas eficiente (menor número de fallos) y en caso de igualdad la de menor coste en recursos. Represente en todos los casos mediante un esquema el contenido del directorio cache y la información guardada en la cache después de ejecutar la primera vez la instrucción JMP 0. Indique también el numero de fallos y los recursos necesarios para su implementación.

**NOTA:** Para indicar como queda guardada una instrucción en la cache utilice la siguiente notación:

[I1] representa la codificación en binario de la instrucción 1.

**C3.-** Se tiene un sistema de memoria virtual con las siguientes características: Sistema paginado con un TLB completamente asociativo de 4 entradas y una cache real para datos de 8 Kbytes asociativa con 2 vías y guardando 8 bytes por bloque.

En un determinado momento, la CPU solicita leer un byte cuya dirección es  $D35F072B_{16}$ . Se produce un acierto en el TLB pero un fallo en la cache. El sistema accede a la dirección de memoria  $45702B_{16}$  y se actualiza el bloque en la cache desde la cual la CPU puede leer el byte solicitado.

Con esta información se pide contestar, justificando brevemente la respuesta, las siguientes preguntas:

a) Tamaño en bits de las direcciones virtual y real:

b) Tamaño máximo de página:

c) Sin tener en cuenta ningún bit de control, el contenido de los campos de la entrada del TLB en donde se ha producido el acierto.

d) Sin tener en cuenta ningún bit de control, el contenido de todos los campos en la entrada de la cache donde se ha guardado el nuevo bloque. Se sabe que el contenido de las direcciones de la memoria real contiguas a la solicitada es:

Dirección ( $4570..$ ) <sub>16</sub>	..30	..2F	..2E	..2D	..2C	<b>..2B</b>	..2A	..29	..28	..27	..26	..25	..24
Byte guardado:	19	AA	23	45	76	<b>89</b>	F5	3D	66	3D	FA	2C	76

**C4.-** Se tiene un procesador segmentado con 5 etapas, **S1:** Captura, **S2:** Decodificación y detección de riesgos, **S3:** Captura de operandos, **S4:** Ejecución y **S5:** Escritura de resultados. Sólo tienen importancia los riesgos de control. La estadística de saltos es del 10% para saltos incondicionales y del 20% para los condicionales, de los cuales la mitad son efectivos. En la etapa S3 del procesador se conoce la dirección de salto y en la S4 se comprueba la condición. Inicialmente, el sistema se para hasta eliminar el riesgo. Se quiere añadir una tabla de predicción de saltos ideal (BTB sin fallos) aplicable sólo para saltos condicionales. Aplicando necesariamente la ley de Amdahl, calcular la mejora que se produce al incorporar al sistema el BTB.

**NOTA:** El CPI sin riesgos es 1. Se aconseja el uso de un cronograma para conocer los retardos asociados a las instrucciones que pueden presentar riesgos.

**C5.-** Complete la tabla adjunta correspondiente a la ejecución de la secuencia de instrucciones para un sistema con planificación dinámica de instrucciones que utiliza la técnica del bus común de datos del algoritmo de Tomasulo. El sistema dispone de 2 ALU's para sumar/restar que tardan 2 ciclos, una unidad para multiplicar que consume 6 ciclos, una unidad para dividir que utiliza 10 y una unidad LD/ST que accede a memoria en 4 ciclos. **NOTA:** la sintaxis de los operandos es en todos los casos  $R_{DESTINO}, R_{FUENTE}, R_{FUENTE}$ .

#### Tomasulo

Instrucción	Emisión	Ejecución	Escritura
I1:LD R2, 100+A1	1	5	6
I2:ADD R5,R4,R1	2		
I3:MUL R3,R2,R4			
I4:ST 200+A2,R5			
I5:DIV R1,R3,R4			
I6:ADD R6,R1,R0			
I7:SUB R1,R5,R3			

Explique brevemente como se soluciona el riesgo WAW entre I7 e I5 y el RAW entre I6 e I5, indicando en que ciclo se actualiza el registro R1 y que instrucción lo realiza.

**P2.-** Suponga que la secuencia de código se ejecuta en un sistema con arquitectura Von Neuman, con una unidad de tratamiento que segmenta la ejecución de cada una de las instrucciones en las siguientes etapas: IF (captura), ID (decodificación, captura de operandos y detección de riesgos ), E1 (opera en la ALU en operaciones aritméticas y calcula PC + desplazamiento en las instrucciones de salto), E2 (utiliza la ALU para calcular la condición en saltos), M (acceso a memoria) y W (escritura de registro).

Inicialmente el contador de programa contiene la dirección de la instrucción I1. El formato de las instrucciones es de 32bit, el procesador emite una instrucción por ciclo con ejecución en orden, y todas las instrucciones pasan por todas las etapas. Se Pide :

b) En el cronograma adjunto muestre la evolución temporal de la primera iteración de la secuencia de instrucciones suponiendo que r5 es distinto de cero (i.e.  $R5 = 10$ ), y el salto de la instrucción 6 no es efectivo porque r4 y r6 son distintos. En este apartado no considere ningún tipo de adelantamiento salvo que las etapas **ID** y **W** pueden acceder en el mismo ciclo de reloj al banco de registros (la etapa **W** accede en la primera mitad y la etapa **ID** en la segunda mitad).

[illegible]

**b)** Repita el apartado (b) pero considerando el adelantamiento de datos entre etapas mas eficiente posible. Marque con una flecha en el cronograma entre que etapas se produce adelantamiento de datos

**c)**

Instrucciones	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
I1 : SUB R1,R2,R3	IF*	ID	E1	E2	M	W																
I2 : LOAD R4, 8(R1)																						
I3 : AND R2, R1, R4																						
I4 : SUB R5,R5,1																						
I5 : BEQ R5,R0,200																						
I6 : BEQ R4 ,R6,-24																						
I7 :STORE 8(R7), R6																						
I8: BNE R4,R6, -32																						

**d)** Para cada uno de los dos apartados anteriores indique la variación que se produce en el cronograma si se modifica el sistema con una Arquitectura Harvard.