# Códigos complicados que se pueden bloquear con crono planificación

1. Sea Superescalar de grado grande y bucle donde todos sus elementos o iteraciones se pueden ejecutar en paralelo; p ej SAXPY (tiene 8 o 9 instrucciones , luego el superescalar debe emitir 8 o 9 instrucciones por ciclo). Cada iteración debe ejecutarse en un ciclo, pero las dependencias de los registros bases de LW/SW son como un contador (instrucciones tipo ADDI Rb,Rb,4 necesarias para ir recorriendo los vectores) y tienen dependencia entre iteraciones ; luego acabarían saturando las HS.
2. Pocas UF de algún tipo. P ej solo una unidad funcional de suma entera. Este código debe acabar ocupando todos los bits del BRT, aunque no haya apenas dependencias reales RAW:

BUCLE:

ADDI R1,R1,1;

ADDI R2,R1,1;

ADDI R3,R1,1;

ADDI R4,R1,1;

ADDI R5,R1,1;

ADDI R6,R1,1;

ADDI R7,R1,1;

ADDI R8,R1,1;

J BUCLE

1. El del viernes pasado de muchas RAWs seguidas, más fácil para que solo se cuente de uno en uno:

BUCLE:

ADDI R1,R1,1;

ADDI R1,R1,1;

ADDI R1,R1,1;

ADDI R1,R1,1;

J BUCLE

1. Accesos a memoria. NOTA: Si vas a detectar coincidencias de direcciones ten presente que los accesos de varios bytes (WORD=4 B, double = 8 B, etc) debería detectar coincidencia de direccion con que alguno de los bytes coincida . EJ hay una SS con la dirección 1000 de un acceso de 8B . Entonces si viene la direccion 1006 de cualquier tamaño, o la dirección 998 de más de 2 B , entonces COINCIDENCIA.

* Un código sin coincidencias de direcciones, pero con muchos accesos a memoria, podría bloquear por la misma causa que el caso 2.
* Códigos con muchas coincidencias de direcciones sería similar al caso 3. P ej:

ADDI R1,R0,1000;

BUCLE:

SW (R1)0, R2;

LW R3, (R1)0;

ADDI R2,R2,R3;

J BUCLE