**Práctica 2. Pipeline segmentado en MIPS**

**Marcos de Diego y Alfredo del Val**

**Optimización de código:**

El código inicial era bastante ineficiente y aunque funcionaba, pero no estaba del todo optimizado por lo que existían múltiples riesgos.

Primero, en la función suma dispersa cambiamos la forma de comparar si el vector de índices es mayor que el vector de datos con el registro $t7 y un slt.

Luego, en el loop de suma dispersa principalmente hemos cambiado el orden, sin modificar la funcionalidad del código original. Esto han sido principalmente movimientos de contadores (addi). El fin principal de esta modificación es aumentar el tiempo que tienen otras funciones como los lw y así obtener un menos número de riesgos.

En MaxMin\_loop también hemos cambiado ligeramente la estructura del bgt, sustituyéndolo por un slt un beq y un addi que resulta en el mismo funcionamiento pero con un menor número de riesgos.

Para al etiqueta check\_min hemos hecho algo muy parecido al anterior partado, sustituir el blt por un slt un addi y un beq.

Hemos implementado dos “funciones” (mover1 y mover2) para copiar en $v0 $t1 y en $t1 $v1 respectivamente. Estas dos etiquetas nos ayudaran a agilizar estos movimientos de copia.

Para incr\_punt hemos movido ambos addi y comprobamos primero si el programa debe acabar con un bne, de lo contrario volvería a saltar a loop.

(\*) En en loop de suma dispersa y en MaxMin\_loop hemos conseguido reducir significativamente los riesgos de datos con los “nops” pero al no saber si son permitidos o no hemos decidido entrgar finalmente el trabajo sin ellos.

En el segundo procesador, la anticipación completa prevé las bifurcaciones y tomar decisiones en una etapa más temprana, en este caso en la etapa de instrucción y puede corregir predicciones de bifurcaciones antes de que lleguen a la etapa de ejecución. Al prever y corregir anticipadamente, se reduce la penalización por predicciones incorrectas, disminuyendo así los riesgos asociados con las bifurcaciones.