LD F0, 0(R1)	-	-	-	-	-
-	LD F0, -8(R1)	-	-	-	-
ADDD F4, F0, F2	-	LD F0, -16(R1)	-	-	-
-	ADDD F4, F0, F2	-	LD F0, -24(R1)	-	-
-	-	ADDD F4, F0, F2	-	LD F0, -32(R1)	-
SD 0(R1), F4	-	-	**ADDD F4, F0, F2**	-	**LD F0, -40(R1)**
-	SD -8(R1), F4	-	-	ADDD F4, F0, F2	-
-	-	SD -16(R1), F4	-	-	ADDD F4, F0, F2
-	-	-	SD -24(R1), F4	-	-
-	-	-	-	SD -32(R1), F4	-
-	-	-	-	-	SD -40(R1), F4
-		-	-	-	-

El principal problema que surge es que el formato de instrucción VLIW dispone de únicamente un campo para operaciones de acceso a memoria y el cuerpo del bucle consta de una carga y un almacenamiento. Ello obliga a generar varias instrucciones VLIW para el cuerpo del bucle segmentado con el riesgo de pérdidas de datos. Una posible solución es intentar generar un patrón que responda al formato de instrucción VLIW, esto, es, una operación de acceso a memoria, una de coma flotante y una entera. La siguiente tabla presenta una solución.

LD F0, 0(R1)	-	-	-
-	-	-	-
ADDD F4, F0, F2	LD F0, -8(R1)	-	-
-	-	-	-
-	ADDD F4, F0, F2	LD F0, -16(R1)	-
SD 0(R1), F4	-	-	-
-	-	**ADDD F4, F0, F2**	**LD F0, -24(R1)**
-	SD -8(R1), F4	-	-
-	-	-	ADDD F4, F0, F2
-	-	SD -16(R1), F4	-
-	-	-	-
-	-	-	SD -24(R1), F4

Una posible transformación de la secuencia previa en instrucciones VLIW se presenta a continuación. Preste atención al juego que se tiene que hacer con los valores del desplazamientos para aprovechar el hueco de retardo del salto.

	Carga/almacenamiento	Coma flotante	Entera
	LD F0, 0(R1)		
	LD F0, -8(R1)	ADDD F4, F0, F2	
	LD F0, -16(R1)	ADDD F4, F0, F2	SUBI R1, R1, #32
Inicio:	**SD 32(R1), F4**	**-**	**BNEZ R1, inicio**
	LD F0, 8(R1)	**ADDD F4, F0, F2**	**SUBI R1, R1, #8**
	SD 32(R1), F4		
		ADDD F4, F0, F2	
	SD 24(R1), F4		
	SD 16(R1), F4		