

Problema

Dado el siguiente fragmento de código:

```
for (i=0; i<100; i++) {
    if (A[i] > 10) then {
        X[i]:=X[i]+b;
    } else {
        if (A[i] > 20) then {
            X[i]:=X[i]+c;
        } else {
            X[i]:=X[i]+d;
        }
    }
}
```

a) Genere el código intermedio equivalente aplicando operaciones con predicado. Considere que los vectores A y X y las constantes b, c y d se encuentran almacenadas en las posiciones de memoria contenidas en los registros Ra, Rx, Rb, Rc y Rd, respectivamente.

b) A partir del código que ha obtenido, genere el código VLIW correspondiente. Considere que una instrucción VLIW admite una operación de carga/almacenamiento (2 ciclos de latencia), una operación en coma flotante (3 ciclos de latencia) y una operación entera/salto (1 ciclo de latencia). Las instrucciones de manipulación de predicados se consideran operaciones enteras.

Solución

a) Un posible código intermedio con operaciones predicadas es el siguiente:

	LD	Fb, 0 (Rb)		// Carga de la constante b
	LD	Fc, 0 (Rc)		// Carga de la constante c
	LD	Fd, 0 (Rd)		// Carga de la constante d
inicio:	LD	Fa, 0 (Ra)		// Carga del vector A
	LD	Fx, 0 (Rx)		// Carga del vector X
	PRED_LT	p1, p2, Fa, #10		// If (R1<=10) {p1:=T; p2:=F}
	ADDD	Fx, Fx, Fb	(p2)	// X[i]:=X[i]+b
	JMP	fin	(p2)	
else:	PRED_LT	p3, p4, Fa, #20		// If (R2<=20) {p3:=T; p4:=F}
	ADDD	Fx, Fx, Fc	(p4)	// X[i]:=X[i]+c
	ADDD	Fx, Fx, Fd	(p3)	// X[i]:=X[i]+d
	SD	0 (Rx), Fx		
fin:	SUBI	Ra, Ra, #4		
	SUBI	Rx, Rx, #4		
	BNEZ	Ra, inicio		

b) El código VLIW generado a partir del código intermedio del apartado anterior y teniendo en cuentas las restricciones del enunciado en cuanto a número de operaciones y latencias se muestra en la tabla situada a continuación. Observe que no se ha incorporado ningún tipo de optimización.

	Carga/almacenamiento	Operaciones FP	Enteras/saltos
1	LD Fb, 0 (Rb)		
2	LD Fc, 0 (Rc)		
3	LD Fd, 0 (Rd)		
4	Inicio: LD Fa, 0 (Ra)		
5	LD Fx, 0 (Rx)		
6			PRED_LT p1,p2,Fa,#10
7		ADDD Fx,Fx,Fb (p2)	
8			
9			JMP fin (p2)
10			PRED_LT p3,p4,Fa,#20
11		ADDD Fx,Fx,Fc (p4)	
12		ADDD Fx,Fx,Fd (p3)	
13			
14			
15			SD 0 (Rx) , Fx
16			fin: SUBI Ra,Ra,#4
17			SUBI Rx,Rx,#4
18			BNEZ Ra,inicio

Una alternativa al código anterior, un poco más optimizada, se presenta a continuación. La existencia de la instrucción 12 responde a la necesidad de dejar los ciclos necesarios para la finalización de las operaciones de suma en coma flotante con el fin de poder realizar el almacenamiento del resultado.

	Carga/almacenamiento	Operaciones FP	Enteras/saltos
1	LD Fb, 0 (Rb)		
2	LD Fc, 0 (Rc)		
3	LD Fd, 0 (Rd)		
4	Inicio: LD Fa, 0 (Ra)		SUBI Ra,Ra,#4
5	LD Fx, 0 (Rx)		SUBI Rx,Rx,#4
6			PRED_LT p1,p2,Fa,#10
7		ADDD Fx,Fx,Fb (p2)	PRED_LT p3,p4,Fa,#20
8			JMP fin (p2)
9		ADDD Fx,Fx,Fc (p4)	
10		ADDD Fx,Fx,Fd (p3)	
11			
12			fin:
13			BNEZ Ra,inicio
14			SD 0 (Rx) , Fx