TIPO DE EXAMEN: 1ª SEMANA - NACIONAL – FEBRERO 2012

Apellidos: ..... Nombre:.....DNI:.....

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja, y ENTRÉGUELA junto con el resto del examen. Lea atentamente todos los enunciados.

Problema 1 (3 puntos)

Utilizando el algoritmo de Tomasulo para realizar la ejecución del siguiente fragmento de código:

```
i1:  MULTD   F2, F2, F6
i2:  MULTD   F4, F2, F6
i3:  ADDD    F2, F4, F6
i4:  ADDD    F6, F2, F6
```

muestre la evolución de los registros en coma flotante (FR) y de las estaciones de reserva (RS) para todos los ciclos que sean necesarios. Considere las siguientes hipótesis de partida:

- Para reducir el número de ciclos máquina se permite que la FLOS distribuya hasta dos instrucciones en cada ciclo según el orden del programa.
- Una instrucción puede comenzar su ejecución en el mismo ciclo en que se distribuye a una estación de reserva.
- La operación suma tiene una latencia de dos ciclos y la de multiplicación de tres ciclos.
- Se permite que una instrucción reenvíe su resultado a instrucciones dependientes durante su último ciclo de ejecución. De esta forma una instrucción a la espera de un resultado puede comenzar su ejecución en el siguiente ciclo si detecta una coincidencia.
- Los valores de etiqueta 01, 02 y 03 se utilizan para identificar las tres estaciones de reserva de la unidad funcional de suma, mientras que 04 y 05 se utilizan para identificar las dos estaciones de reserva de la unidad funcional de multiplicación/división. Estos valores de etiqueta son los ID de las estaciones de reserva.
- Inicialmente, el valor de los registros es F0=2.0, F2=2.5, F4=4.0 y F6=3.0.

Problema 2 (4 puntos)

Dispone del siguiente fragmento de código intermedio:

```
Loop: LD    F0,0(R1)
      ADDD  F4,F0,F2
      SD    0(R1),F4
      SUBI  R1,R1,#8
      BNEZ  R1,Loop
```

y de un procesador VLIW con un formato de instrucción de 5 slots (4 bytes por slot) que admite dos operaciones de carga/almacenamiento (2 ciclos de latencia), dos operaciones en coma flotante (3 ciclos de latencia) y una operación entera/salto (1 ciclo de latencia). Sin considerar la existencia del hueco de retardo de salto en la planificación, se pide que:

a) Transforme el código intermedio en código VLIW para el procesador indicado.
b) A partir del código anterior y mediante el desenrollamiento del bucle original, complete los slots libres del código VLIW del apartado anterior.
c) Realice el desenrollamiento software del bucle original. Considere que un slot de operación en coma flotante puede ejecutar restas enteras.
d) Calcule para los dos apartados anteriores el número de operaciones por ciclo reloj, el número de ciclos consumidos para un vector de 800 elementos, el tamaño del código en memoria y el porcentaje de espacio desaprovechado.

Problema 3 (3 puntos)

Dada una red con topología de hipercubo con dimensión $d = 5$, se pide que:

a) Dibuje los hipercubos de dimension d-1 que forman dicha red.
b) Calcule la distancia de Hamming para los procesadores 00000 y 11111. Dibuje y explique razonadamente un esquema del camino mas corto para comunicar ambos procesadores.
c) Describa y calcule la conectividad de arco de dicha red.

Solución al problema 1

a)

Ciclo 1: Se distribuyen i1 e i2 en orden.

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01				
02				
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04 (i1)	00	2.5	00	3
05 (i2)	04	--	00	3
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2	1	04	2.5
F4	1	05	4
F6			3

Ciclo 2: Se distribuyen i3 e i4 en orden.

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01 (i3)	05	--	00	3
02 (i4)	01	--	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04 (i1)	00	2.5	00	3
05 (i2)	04	--	00	3
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2	1	01	2.5
F4	1	05	4
F6	1	02	3

Ciclo 3: Al final del ciclo 3, la instrucción i1 finaliza su ejecución y emite su ID (04). En ese momento, todos los campos etiquetados que contienen el valor 04 insertan el resultado emitido.

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01 (i3)	05	--	00	3
02 (i4)	01	--	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04 (i1)	00	2.5	00	3
05 (i2)	04	--	00	3
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2	1	01	2.5
F4	1	05	4
F6	1	02	3

Ciclo 4:

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01 (i3)	05	--	00	3
02 (i4)	01	--	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04				
05 (i2)	00	7.5	00	3
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2	1	01	2.5
F4	1	05	4
F6	1	02	3

Ciclo 5:

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01 (i3)	05	--	00	3
02 (i4)	01	--	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04				
05	00	7.5	00	3
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2	1	01	2.5
F4	1	05	4
F6	1	02	3

Ciclo 6: Al final del ciclo 6 finaliza la ejecución de i2

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01 (i3)	05	--	00	3
02 (i4)	01	--	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04				
05	00	7.5	00	3
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2	1	01	2.5
F4	1	05	4
F6	1	02	3

Ciclo 7:

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01 (i3)	00	22.5	00	3
02 (i4)	01	--	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04				
05				
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2	1	01	2.5
F4			22.5
F6	1	02	3

Ciclo 8: Al final del ciclo 8 finaliza la ejecución de i3.

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01 (i3)	00	22.5	00	3
02 (i4)	01	--	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04				
05				
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2	1	01	2.5
F4			22.5
F6	1	02	3

Ciclo 9:

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01				
02 (i4)	00	25.5	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04				
05				
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2			25.5
F4			22.5
F6	1	02	3

Ciclo 10: Al final del ciclo 10, i4 finaliza su ejecución.

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01				
02 (i4)	00	25.5	00	3
03				
	Suma/Resta			

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04				
05				
	Mult/Div			

FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2			25.5
F4			22.5
F6	1	02	3

Ciclo 11

RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
01				
02				
03				
	Suma/Resta			

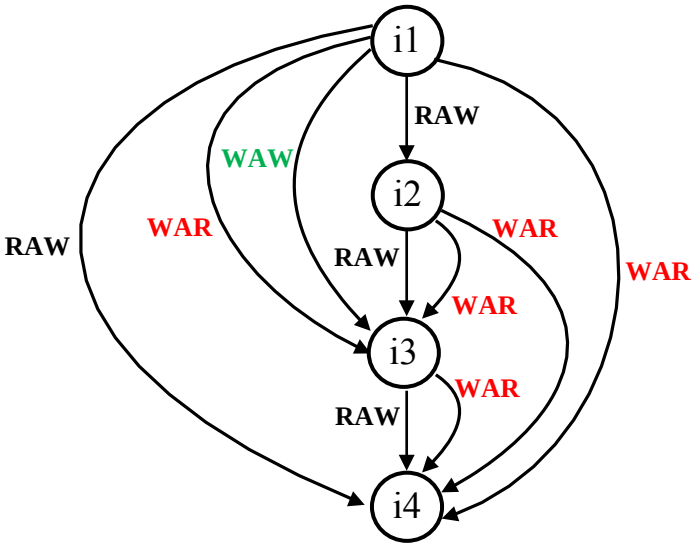
RS

ID	Eti_1	Ope_1	Eti_2	Ope_2
04				
05				
	Mult/Div			

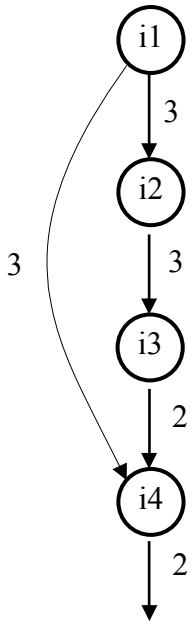
FR

ID	Bit Ocu	Etiq.	Dato
F0			2
F2			25.5
F4			22.5
F6			28.5

b) El siguiente diagrama ilustra todas las dependencias de datos existentes entre las instrucciones que componen la secuencia.



c) A partir del gráfico con las dependencias verdaderas se puede establecer de forma sencilla el límite máximo de flujo de datos. Se puede apreciar que la ruta crítica consta de 10 ciclos que es el límite que debe alcanzar la ejecución aplicando el algoritmo de Tomasulo.



Solución al problema 2

a) Una solución válida aunque no óptima es la siguiente.

	Carga/almacenamiento	Carga/almacenamiento	Operaciones FP	Operaciones FP	Enteras/saltos
1	`LD F0,0(R1)`				
2					
3			`ADDD F4,F0,F2`		
4					
5					
6	`SD 0(R1),F4`				
7					
8					`SUBI R1,R1,#8`
9					`BNEZ R1,Loop`

Número de ciclos consumidos: 9
Número de operaciones realizadas: 5
Operaciones por ciclo: 0,555
Tamaño en memoria: 9 instrucciones * 20 bytes = 180 bytes
Espacio utilizado: 5 operaciones * 4 bytes = 20 bytes
% espacio desaprovechado: 88,89
Ciclos ejecutados para 800 elementos: 9 ciclos * 800 iteraciones : 7200 ciclos
Instrucciones procesadas: 9 * 800 iteraciones: 7200 instrucciones

Otra solución válida que mejora a la anterior es la que se muestra a continuación.

	Carga/almacenamiento	Carga/almacenamiento	Operaciones FP	Operaciones FP	Enteras/saltos
1	`LD F0,0(R1)`				
2					
3			`ADDD F4,F0,F2`		
4					
5					
6	`SD 0(R1),F4`				`SUBI R1,R1,#8`
7					`BNEZ R1,Loop`

Número de ciclos consumidos: 7
Número de operaciones realizadas: 5
Operaciones por ciclo: 0,71
Tamaño en memoria: 7 instrucciones * 20 bytes = 140 bytes
Espacio utilizado: 5 operaciones * 4 bytes = 20 bytes
% espacio desaprovechado: 85%
Ciclos ejecutados para 800 elementos: 7 ciclos * 800 iteraciones : 6300 ciclos
Instrucciones procesadas: 7 * 800 iteraciones: 6300 instrucciones

b) En base a la solución no óptima del apartado (a) se tendría:

	Carga/almacenamiento	Carga/almacenamiento	Operaciones FP	Operaciones FP	Enteras/saltos
1	LD F0,0(R1)	LD F6,-8(R1)			
2	LD F10,-16(R1)	LD F14,-24(R1)			
3	LD F18,-32(R1)	LD F22,-40(R1)	ADDD F4,F0,F2	ADDD F8,F6,F2	
4			ADDD F12,F10,F2	ADDD F16,F14,F2	
5			ADDD F20,F18,F2	ADDD F24,F22,F2	
6	SD 0(R1),F4	SD -8(R1),F8			
7	SD -16(R1),F12	SD -24(R1),F16			
8	SD -32(R1),F20	SD -40(R1),F24			SUBI R1,R1,#48
9					BNEZ R1,Loop

Número de ciclos consumidos: 9
Número de operaciones realizadas: 20
Operaciones por ciclo: 20/9=2,222
Tamaño en memoria: 9 instrucciones * 20 bytes = 180 bytes
Espacio utilizado: 20 operaciones * 4 bytes = 80 bytes
% espacio desaprovechado: 55 %
Ciclos ejecutados para 800 elementos: 9 ciclos * 134 iteraciones : 1206 ciclos
Instrucciones procesadas: 9 * 134 iteraciones: 1206 instrucciones

En base a la solución óptima del apartado (a) se tendría:

	Carga/almacenamiento	Carga/almacenamiento	Operaciones FP	Operaciones FP	Enteras/saltos
1	LD F0,0(R1)	LD F6,-8(R1)			
2					
3			ADDD F4,F0,F2	ADDD F8,F6,F2	
4					
5					
6	SD 0(R1),F4	SD -8(R1),F8			SUBI R1,R1,#16
7					BNEZ R1,Loop

Número de ciclos consumidos: 7
Número de operaciones realizadas: 8
Operaciones por ciclo: 8/7=1,14
Tamaño en memoria: 7 instrucciones * 20 bytes = 140 bytes
Espacio utilizado: 8 operaciones * 4 bytes = 32 bytes
% espacio desaprovechado: 77%
Ciclos ejecutados para 800 elementos: 7 ciclos * 400 iteraciones : 2800 ciclos
Instrucciones procesadas: 7 * 400 iteraciones: 2800 instrucciones

c) Lo primero que hay que realizar es obtener el patrón de ejecución con el objeto de visualizar el prólogo, el patrón que se repite y el epílogo.

Iteracción 1	Iteracción 2	Iteracción 3	Iteracción 4	Iteracción 5	Iteracción 6
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

Tras visualizar el esquema, hay que trasladarlo a las instrucciones VLIW del procesador disponible.

Carga/almacenamiento	Carga/almacenamiento	Operaciones FP	Operaciones FP	Enteras/saltos
LD F0,0(R1)				
LD F0,-8(R1)				
LD F0,-16(R1)		ADDD F4,F0,F2		
LD F0,-24(R1)		ADDD F4,F0,F2		
LD F0,-32(R1)		ADDD F4,F0,F2		
LD F0,-40(R1)	SD 0(R1),F4	ADDD F4,F0,F2	SUBI R1,R1,#8	BNEZ R1,Loop
	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			

Dado que la instrucción de comparación realiza la comparación con 0, es necesario reajustar los desplazamientos de las instrucciones de carga/almacenamiento y el contenido del registro R1 con el objeto de que el último elemento almacenado lo sea en la posición de memoria M[8] tal y como sucede en el bucle original (observe en el bucle escalar original que se almacena en M[0+R1] y tras decrementar se comprueba que R1 sea cero, en caso afirmativo el bucle concluye).

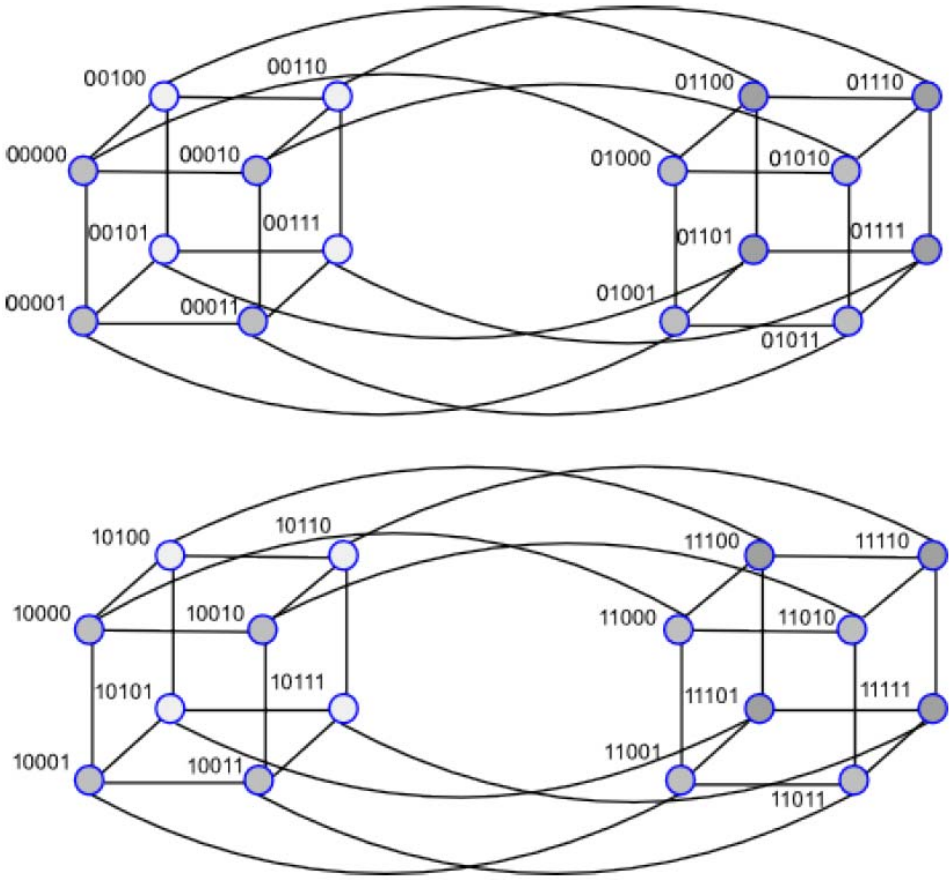
En este caso, el valor inicial de R1 debe ser R1 = R1-48 se procede al proceder al ajuste de los desplazamientos de las instrucciones de carga/almacenamiento. Se tiene así:

Carga/almacenamiento	Carga/almacenamiento	Operaciones FP	Operaciones FP	Enteras/saltos
LD F0,48(R1)				
LD F0,40(R1)				
LD F0,32(R1)		ADDD F4,F0,F2		
LD F0,24(R1)		ADDD F4,F0,F2		
LD F0,16(R1)		ADDD F4,F0,F2		
LD F0,8(R1)	SD 48(R1),F4	ADDD F4,F0,F2	SUBI R1,R1,#8	BNEZ R1,Loop
	SD 48(R1),F4	ADDD F4,F0,F2		
	SD 40(R1),F4	ADDD F4,F0,F2		
	SD 32(R1),F4			
	SD 24(R1),F4			
	SD 16(R1),F4			

Número de ciclos consumidos: 1 ciclo
Número de operaciones realizadas: 5 operaciones
Operaciones por ciclo: 5 operaciones/ciclo
Tamaño en memoria: 11 instrucciones * 20 bytes = 220 bytes
Espacio utilizado: 20 operaciones * 4 bytes = 80 bytes
Espacio desaprovechado: 63 %
Ciclos ejecutados para 800 elementos: 5 del prólogo + 5 del epílogo + 95 iteraciones de 1 ciclo : 105 ciclos
Instrucciones procesadas: 105 instrucciones

Solución al problema 3

a)



b) La distancia de Hamming para los procesadores 00000 y 11111 se calcula utilizando la operación XOR. Así 00000 $\oplus$ 11111 = 11111, siendo la distancia el número de bits a 1 en el resultado de dicha operación, es decir, 5. El esquema del camino más corto será

$$00000 \rightarrow 00001 \rightarrow 00011 \rightarrow 00111 \rightarrow 01111 \rightarrow 11111$$

Dado que todos los bits de la distancia de Hamming entre ambos procesadores tiene valor 1, el camino más corto se realizará cambiando todos los bits del procesador inicial de 0 a 1, desde el menos significativo al más significativo.

c) La conectividad de arco de una red hipercubo se describe como “el menor número de arcos que deben eliminarse para obtener dos redes disjuntas”. Esta conectividad se puede calcular como el log2(p). Dado que un hipercubo de dimensión 5 tiene 32 procesadores, log2(32) = 5.

TIPO DE EXAMEN: 2ª SEMANA - NACIONAL Y UNIÓN EUROPEA - FEBRERO 2012

Apellidos:.....Nombre:.....DNI:.....

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja, y ENTRÉGUELA junto con el resto del examen. Lea atentamente todos los enunciados.

Problema 1 (2 puntos)

Un procesador sin segmentación necesita 150 nseg. para procesar una instrucción. Con respecto a este procesador, calcule la aceleración que se obtiene en los dos siguientes casos:

a) Un procesador A dotado de una segmentación de 7 etapas, consumiendo cada etapa el mismo tiempo. Cada etapa ocasiona una sobrecarga de 6 nseg., no existiendo ningún tipo de detención en la segmentación.
b) Un procesador B con una segmentación de 7 etapas, consumiendo cada una de ellas 30 nseg., 30 nseg., 40 nseg., 50 nseg. y 50 nseg. respectivamente, y siendo la sobrecarga por cada etapa de 6 nseg. Un 33% de todas las instrucciones de la segmentación son detenidas durante un ciclo de reloj y un 8% durante dos ciclos.

Problema 2 (3 puntos)

En un procesador vectorial con las siguientes características:

- Registros con una longitud vectorial máxima de 64 elementos.
- Una unidad de suma vectorial con tiempo de arranque de 6 ciclos.
- Una unidad de multiplicación con tiempo de arranque de 7 ciclos.
- Una unidad de carga/almacenamiento con tiempo de arranque de 12 ciclos.
- La frecuencia de trabajo del procesador es 500 MHz.
- Tbase de 10 ciclos y Tbucle de 15 ciclos.

se pretende ejecutar el siguiente bucle:

```
for (i=1; i<=n; i++)
      A(i) := x*A(i) + y*A(i);
end for;
```

Escriba el código vectorial que realizaría las operaciones ubicadas en el interior del bucle y calcule T_n, T_{1000}, R_{1000} y R_∞ en los siguientes casos:

a) Sin considerar encadenamiento de resultados.
b) Permitiendo encadenamientos.
c) Considerando encadenamientos y dos unidades de multiplicación.

Problema 3 (3 puntos)

Se dispone de un sistema biprocesador (CPUs A y B) de memoria compartida que utiliza un protocolo snoopy de coherencia de caché. Sabiendo que la CPU B tiene cargada en caché la variable X y que la CPU A realiza una lectura sobre la variable X (`read(X)`), seguida de una escritura sobre la misma variable (`write(X)`). Se pide que:

a) Describa la secuencia de acciones de coherencia y las etiquetas de las cachés de cada procesador para la variable X durante las instrucciones ejecutadas.
b) ¿Qué problemas pueden ocurrir en caso de necesitar que las operaciones realizadas por la CPU A se realicen de manera atómica, es decir, que su resultado sea independiente de las posibles acciones realizadas por la CPU B mientras la CPU A está ejecutando sus acciones?
c) Describa la secuencia de acciones de coherencia y las etiquetas de las cachés de cada procesador para la variable X en cada uno de los posibles problemas.

Problema 1 Febrero 2012 2ª (2 puntos) – Igual que actividad 1.3 del libro de texto.

Un procesador sin segmentación necesita 150 nseg. para procesar una instrucción. Con respecto a este procesador, calcular la aceleración que se obtiene en los dos casos siguientes:

a) Un procesador A dotado de una segmentación de 7 etapas, consumiendo cada etapa el mismo tiempo. Cada etapa ocasiona una sobrecarga de 6 nseg. no existiendo ningún tipo de detención en la segmentación.

De acuerdo con el enunciado el tiempo medio de ejecución de una instrucción en el procesador sin segmentar es de 150 nseg. La segmentación de 7 etapas de este apartado se caracteriza por acortar el tiempo medio de ejecución de una instrucción a 27,43 nseg.:

$$\frac{150\ nseg}{7\ etapas} + 6\ nseg = 27,43\ nseg$$

Por lo tanto, la aceleración obtenida por la máquina A con respecto a la máquina sin segmentar es 5,47:

$$\frac{150\ nseg}{27,43\ nseg} = 5,47\ veces\ más\ rápido$$

b) Un procesador B con una segmentación de 7 etapas, consumiendo cada una de ellas 30 nseg., 30 nseg., 40 nseg., 50 nseg. y 50 nseg. respectivamente, y siendo la sobrecarga por cada etapa de 6 nseg. Un 33% de todas las instrucciones de la segmentación son detenidas durante un ciclo de reloj y un 8% durante dos ciclos.

La etapa más lenta es la que dicta la velocidad de las restantes etapas, por lo que cada etapa consumirá 56 nseg. (50 nseg. más los 6 nseg. de retardo).

El 8% ocasiona una detención de dos ciclos, por lo que consumen 168 nseg. $\left(3\ ciclos \cdot 56\ nseg\right)$.

El 33% ocasiona una detención de un ciclo, consumiendo 112 nseg. $\left(2\ ciclos \cdot 56\ nseg\right)$

El 59%, no provocan detenciones, empleando sólo un ciclo de reloj (56 nseg.).

De acuerdo con esto, el tiempo medio consumido por una instrucción es:

$$0,33 \cdot 56 \cdot 2\ nseg + 0,08 \cdot 56 \cdot 3\ nseg + 0,59 \cdot 56 \cdot 1\ nseg = 83,44\ nseg\ .$$

Por lo tanto, la aceleración obtenida por la máquina B con respecto a la máquina sin segmentar es de 1,8:

$$\frac{150\ nseg}{83,44\ nseg} = 1,8\ veces\ más\ rápido$$

Problema 2 Febrero 2ª Semana 2012 (3 puntos) - Igual que actividad 3.12 del libro de texto.

En un procesador vectorial con las siguientes características:

- Registros con una longitud vectorial máxima 64 elementos.
- Una unidad de ***suma vectorial*** con tiempo de arranque de 6 ciclos.
- Una unidad de ***multiplicación*** con tiempo de arranque de 7 ciclos.
- Una unidad de ***carga/almacenamiento vectorial*** con tiempo de arranque de 12 ciclos.
- La frecuencia del trabajo del procesador es 500 MHz.
- T_{base} de 10 ciclos y T_{bucle} de 15 ciclos.

Se pretende ejecutar el siguiente bucle:

```
for (i=1; i<n; i++)
   A(i):= x*A[i]+ y*A[i]);
end for;
```

Escriba el código vectorial que realizaría las operaciones ubicadas en el interior del bucle y calcule T_n, T_{1000}, R_{1000} y R_∞ en los siguientes casos:

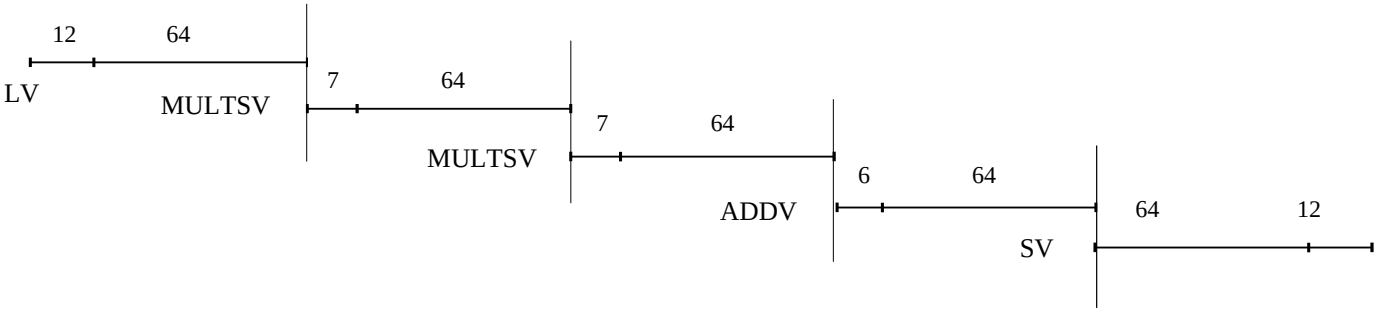
a) Sin considerar encadenamiento de resultados.
b) Permitiendo encadenamientos.
c) Considerando encadenamientos y dos unidades de multiplicación

a) Analizando los riesgos estructurales se obtiene el siguiente código vectorial:

```
Convoy 1:  LV       V1, R1       // Carga de A en V1
Convoy 2:  MULTSV   V2, F0, V1   // B := x * A
Convoy 3:  MULTSV   V3, F2, V1   // C := Y * A
Convoy 4:  ADDV     V4, V3, V2   // A := B+C
Convoy 5:  SV       R1, V4       // Almacenamiento de A
```

La secuencia de ejecución de los cinco convoyes si se considera que VLR es 64 es la que se muestra en la siguiente figura.

$T_{elemento}$ = 5 ciclos



Dado que hay cinco convoyes, $T_{elemento}$ es 5 ciclos y el $T_{arranque}$ total es igual a la suma de los tiempos de arranque visibles de los cinco convoyes. Esto es

$T_{arranque} = T_{arranque}$ LV + 2*$T_{arranque}$ MULTSV + $T_{arranque}$ ADDV + $T_{arranque}$ SV

$T_{arranque}$ = (12 + 2*7 + 6 + 12) ciclos = 44 ciclos

Sustituyendo los valores conocidos de $T_{arranque}$ y $T_{elemento}$ en la expresión que determina el tiempo de ejecución de un bucle vectorizado para vectores de longitud *n* se tiene

$$T_n = 10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15+44) + n \cdot 5$$

que para el caso particular de n=1000 es

$$T_{1000} = 10 + \left\lceil \frac{1000}{64} \right\rceil \cdot (15+44) + 5 \cdot 1000 = 10 + 16 \cdot (15+44) + 5 \cdot 1000 = 5954 \ \ ciclos$$

$$R_{1000} = \frac{3 \cdot 1000}{T_{1000}} = \frac{3000}{5954} = 0,5039 \ {}^{FLOP}\!/_{ciclo}$$

El rendimiento expresado en FLOP/ciclo es

$$R_\infty = \lim_{n\to\infty} \left(\frac{3n}{T_n} \right) = \lim_{n\to\infty} \left(\frac{3n}{10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15+44) + 5n} \right)$$

Para simplificar los cálculos, la expresión $\lceil n/64 \rceil$ se puede reemplazar por una cota superior dada por $(n/64 + 1)$. Sustituyendo esta cota en R_∞ y teniendo en cuenta que el número de operaciones vectoriales que se realizan en el bucle DAXPY son dos, una multiplicación y una suma, se tiene

$$R_\infty = \lim_{n\to\infty} \left(\frac{3n}{10 + \left\lceil \frac{n}{64} + 1 \right\rceil \cdot (15+44) + 5n} \right) = \lim_{n\to\infty} \left(\frac{3n}{69 + 5,9219n} \right) = 0,5066 \ \ FLOP/ciclo$$

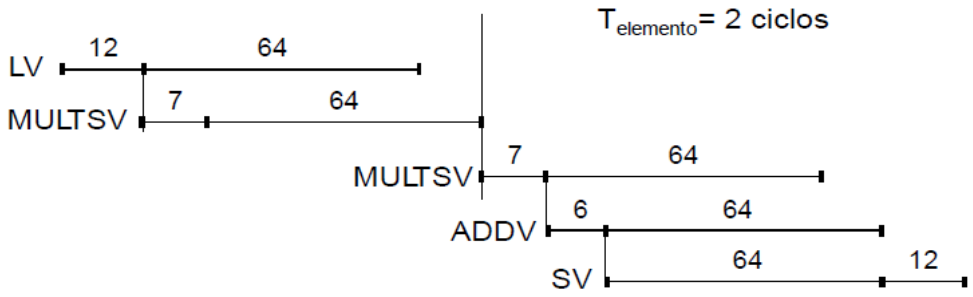
Para expresar R_∞ en FLOPS, habría que multiplicar el valor en FLOP/ciclo por la frecuencia del procesador. Se tendría así

$$R_\infty = 0,409 \text{ FLOP/ciclo} \cdot (500 \cdot 10^6) \text{ Hz}$$

$$R_\infty = 204,5 \text{ MFLOPS}$$

b) Dado que ahora es posible encadenar los resultados de las unidades, la organización del código vectorial en convoyes quedaría de la siguiente forma:

```
Convoy 1:   LV        V1, R1       // Carga de A en V1
            MULTSV    V2, F0, V1   // B := x * A
Convoy 2:   MULTSV    V3, F2, V1   // C := y * A
            ADDV      V4, V3, V2   // A := B + C
            SV        R1, V4       //Almacenamiento de A
```



El $T_{elemento}$ ha pasado a ser de 2 ciclos dado que ahora se tienen dos convoyes. El $T_{arranque}$ total se obtiene de sumar los tiempos de arranque visibles de las unidades funcionales. Si se analiza la figura se tiene

$T_{arranque} = T_{arranque}$ LV + 2* $T_{arranque}$ MULTV + $T_{arranque}$ ADDV + $T_{arranque}$ SV

$T_{arranque}$ = (12 + 2*7 + 6 + 12) ciclos = 44 ciclos

Con estos valores la expresión del tiempo total de ejecución queda

$$T_n = 10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15 + 44) + n \cdot 2$$

que para el caso particular de n=1000 es

$$T_{1000} = 10 + \left\lceil \frac{1000}{64} \right\rceil \cdot (15 + 44) + 2 \cdot 1000 = 10 + 16 \cdot (15 + 44) + 2 \cdot 1000 = 2954 \ \ ciclos$$

$$R_{1000} = \frac{3 \cdot 1000}{T_{1000}} = \frac{3000}{2954} = 1,0156 \ {}^{FLOP}\!/_{ciclo}$$

En lo que respecta al rendimiento expresado en FLOP por ciclo

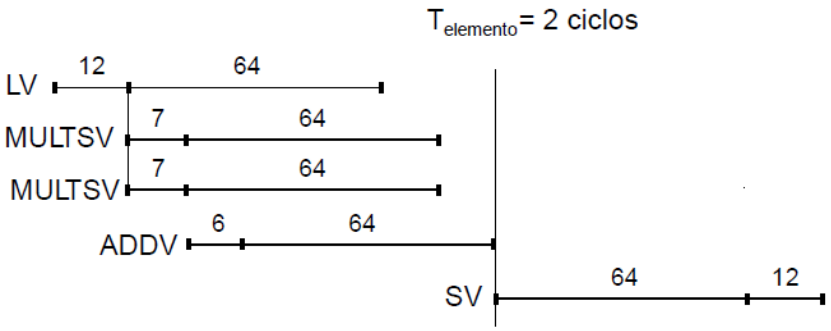
$$R_\infty = \lim_{n \to \infty} \left(\frac{3n}{T_n} \right) = \lim_{n \to \infty} \left(\frac{3n}{10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15 + 44) + 2n} \right) = \lim_{n \to \infty} \left(\frac{3n}{10 + \left\lceil \frac{n}{64} + 1 \right\rceil \cdot (15 + 44) + 2n} \right) =$$

$$\lim_{n \to \infty} \left(\frac{3n}{69 + 2,9219n} \right) = 1,0267 \ \ FLOP / ciclo$$

Claramente se aprecia la mejora en el rendimiento del procesador gracias al encadenamiento de los resultados entre las unidades funcionales.

c) Ahora es posible encadenar los resultados de las unidades y se dispone de dos unidades de multiplicación:

Convoy 1:	LV	V1, R1	// Carga de A en V1
	MULTSV	V2, F0, V1	// B := x * A
	MULTSV	V3, F2, V1	// C := y * A
	ADDV	V4, V3, V2	// A := B + C
Convoy 2:	SV	R1, V4	//Almacenamiento de A



El T*elemento* ha pasado a ser de 2 ciclos dado que ahora se tienen dos convoyes. El $T_{arranque}$ total se obtiene de sumar los tiempos de arranque visibles de las unidades funcionales. Si se analiza la figura se tiene

$T_{arranque} = T_{arranque}$ LV + $T_{arranque}$ MULTV + $T_{arranque}$ ADDV + $T_{arranque}$ SV

$T_{arranque}$ = (12 + 7 + 6 + 12) ciclos = 37 ciclos

Con estos valores la expresión del tiempo total de ejecución queda

$$T_n = 10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15+37) + n \cdot 2$$

que para el caso particular de n=1000 es

$$T_{1000} = 10 + \left\lceil \frac{1000}{64} \right\rceil \cdot (15+37) + 2 \cdot 1000 = 10 + 16 \cdot (15+37) + 2 \cdot 1000 = 2842 \ \ ciclos$$

$$R_{1000} = \frac{3 \cdot 1000}{T_{1000}} = \frac{3000}{2842} = 1,0555 \ {}^{FLOP}\!/_{ciclo}$$

En lo que respecta al rendimiento expresado en FLOP por ciclo

$$R_\infty = \lim_{n\to\infty}\left(\frac{3n}{T_n}\right) = \lim_{n\to\infty}\left(\frac{3n}{10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15+37) + 2n}\right) = \lim_{n\to\infty}\left(\frac{3n}{10 + \left\lceil \frac{n}{64} + 1 \right\rceil \cdot (15+37) + 2n}\right) =$$

$$\lim_{n\to\infty}\left(\frac{3n}{69 + 2,8125n}\right) = 1,315 \ \ FLOP/ciclo$$

TIPO DE EXAMEN: ORIGINAL - NACIONAL - SEPTIEMBRE 2012

Apellidos:.....**Nombre:**.....**DNI:**.....

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja, y ENTRÉGUELA junto con el resto del examen. Lea atentamente todos los enunciados. **SE PERMITE UN LIBRO Y CALCULADORA NO PROGRAMABLE.**

Problema 1 (3 puntos)

Utilizando el algoritmo de Tomasulo, muestre la evolución de los registros en coma flotante (FR) y las estaciones de reserva (RS) para todos los ciclos que sean necesarios en la ejecución del siguiente fragmento de código:

```
i1:  ADDD    F6,F4,F2
i2:  MULTD   F0,F4,F6
i3:  MULTD   F6,F6,F2
i4:  ADDD    F2,F6,F0
```

Considere las siguientes hipótesis de partida:

- Para reducir el número de ciclos máquina se permite que la FLOS distribuya hasta dos instrucciones en cada ciclo según el orden del programa.
- Una instrucción puede comenzar su ejecución en el mismo ciclo en que se distribuye a una estación de reserva.
- La operación suma tiene una latencia de un ciclo y la de multiplicación de dos ciclos.
- Se permite que una instrucción reenvíe su resultado a instrucciones dependientes durante su último ciclo de ejecución. De esta forma, una instrucción a la espera de un resultado puede comenzar su ejecución en el siguiente ciclo si detecta una coincidencia.
- Los valores de etiqueta 01, 02 y 03 se utilizan para identificar las tres estaciones de reserva de la unidad funcional de suma, mientras que 04 y 05 se utilizan para identificar las dos estaciones de reserva de la unidad funcional de multiplicación/división. Estos valores de etiqueta son los ID de las estaciones de reserva.
- Inicialmente, el valor de los registros es `F0`=8.0, `F2`=3.5, `F4`=2.0 y `F6`=3.0

Problema 2 (4 puntos)

Suponga que un salto tiene la siguiente secuencia de resultados efectivos (E) y no efectivos (N):

E, E, E, N, N, E, E, E, N, N, E, E, E, N, N

- Muestre mediante una tabla la secuencia de predicciones utilizando un contador de saturación de 1 bit (predictor de Smith) para la secuencia dada. Suponga que el estado inicial del contador es efectivo (T-Taken) .
- Muestre mediante una tabla la secuencia de predicciones utilizando un contador de saturación de 2 bits (predictor de Smith) para la secuencia dada. Suponga que el estado inicial del contador es fuertemente efectivo (ST – Strongly Taken) .
- ¿Cuál es la precisión de la predicción que han obtenido ambos contadores para la secuencia de saltos?

Problema 3 (3 puntos)

1. Dibuje una red de tipo *crossbar* de 8 x 8 elementos y describa cada uno de los componentes que forman la red.
2. ¿Cuáles son las principales diferencias entre la red dibujada y una red bidimensional de tipo *mesh* cuadrada de tamaño equivalente, es decir, que forme una matriz de 8 x 8?
3. Explique de manera razonada el cálculo del máximo tiempo de transferencia de un mensaje de 10 palabras en ambas redes (*crossbar* y *mesh*), teniendo en cuenta que el tiempo de inicialización del mensaje son 10ms, el tiempo de salto es 1ms y el tiempo de transferencia por palabra son 3ms. El algoritmo de enrutamiento utilizado es *store-and-forward*.

Problema 1 - 2012 septiembre

Mostrar la evolución de los Registros en coma flotante (FR) y las estaciones de Reserva (RS) para todos los ciclos que sean necesarios en la ejecución del siguiente fragmento de código utilizando el algoritmo de Tomasulo.

```
i1: ADDD   F6,F4,F2
i2: MULTD  F0,F4,F6
i3: MULTD  F6,F6,F2
i4:  ADDD  F2,F6,F0
```

- Considere las siguientes hipótesis de partida:
- Para reducir el número de ciclos máquina se permite que la FLOS distribuya hasta dos instrucciones en cada ciclo según el orden del programa.
- Una instrucción puede comenzar su ejecución en el mismo ciclo en que se distribuye a una estación de reserva.

- La operación **suma** tiene una latencia de **un ciclo** y la de **multiplicación** de **dos ciclos**.
- Se permite que una instrucción reenvíe su resultado a instrucciones dependientes durante su último ciclo de ejecución. De esta forma una instrucción a la espera de un resultado puede comenzar su ejecución en el siguiente ciclo si se detecta una coincidencia.

- Los valores de etiqueta 01, 02 y 03 se utilizan para identificar las tres estaciones de reserva de la unidad funcional de suma, mientras que 04 y 05 se utilizan para identificar las dos estaciones de reserva de la unidad funcional de multiplicación/división. Estos valores de etiqueta son los ID de las estaciones de reserva.
- Inicialmente, el valor de los registros es F0=8.0, F2=3.5, F4=2.0 y F6=3.0.

Ciclo 1: Se distribuye i1 e i2
i1: ADDD F6,F4,F2
i2: MULTD F0,F4,F6
Se ejecuta RS 01 1/1
Se envía RS 01: 5.5 al CDB
No se ejecuta RS 04

FR

	bitOc.	etiqueta	dato
F0	Si	04	8.0
F2			3.5
F4			2.0
F6	Si	01	3.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i1: 01	00	2.0	00	3.5
02				
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i2: 04	00	2.0	01	xx
05				

MULT/DIV

Ciclo 2: Se distribuye i3 e i4
i3: MULTD F6,F6,F2
i4: ADDD F2,F6,F0
Se actualiza el valor de RS 01: 5.5
Se vacía RS 01
Se ejecuta RS 04 1/2
Se ejecuta RS 05 1/2
NO se ejecuta RS 02

FR

	bitOc.	etiqueta	dato
F0	Si	04	8.0
F2	Si	02	3.5
F4			2.0
F6	Si	05	3.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
i4: 02	05	xx	04	xx
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i2: 04	00	2.0	00	5.5
i3: 05	00	5.5	00	3.5

MULT/DIV

Ciclo 3: Se ejecuta RS 04 2/2
Se ejecuta RS 05 2/2
Se envía RS 04: 11.0 al CDB
Se envía RS 05: 19.25 al CDB
NO se ejecuta RS 02

FR

	bitOc.	etiqueta	dato
F0	Si	04	8.0
F2	Si	02	3.5
F4			2.0
F6	Si	05	3.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
i4: 02	05	xx	04	xx

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i2: 04	00	2.0	00	5.5
i3: 05	00	5.5	00	3.5

MULT/DIV

Ciclo 4: Se actualiza el valor de RS 04: 11.0
Se actualiza el valor de RS 05: 19.25
Se vacía RS 04
Se vacía RS 05
Se ejecuta RS 02 1/1
Se envía RS 02: 30.25 al CDB

FR

	bitOc.	etiqueta	dato
F0			11.0
F2	Si	02	3.5
F4			2.0
F6			19.25

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
i4: 02	05	19.25	04	11.0

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

Ciclo 5: Se actualiza el valor de RS 02: 30.25
Se vacía RS 02

FR

	bitOc.	etiqueta	dato
F0			11.0
F2			30.25
F4			2.0
F6			19.25

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
02				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

Problema 2 Septiembre 2012 Original

Suponga que un salto tiene la siguiente secuencia de resultados efectivos (E) y no efectivos (N):

E, E, E, N, N, E, E, E, N, N, E, E, E, N, N

- Muestre mediante una tabla la secuencia de predicciones utilizando un contador de saturación de 1 bit (predictor de Smith) para la secuencia dada. Suponga que el estado inicial del contador es efectivo (T-Taken).
- Muestre mediante una tabla la secuencia de predicciones utilizando un contador de saturación de 2 bits (predictor de Smith) para la secuencia dada. Suponga que el estado inicial del contador es fuertemente efectivo (ST – Strongly Taken).
- ¿Cuál es la precisión de la predicción que han obtenido ambos contadores para la secuencia de saltos?

Solución

La siguientes tablas muestra la evolución del contador de saturación de 1 y 2 bits, la predicción del salto, el resultado real del salto, si la predicción fue correcta o no y la transición al siguiente estado.

Contador	Predicción	Resultado real	¿Predicción correcta?	Siguiente estado
T	E	E	Sí	T
T	E	E	Sí	T
T	E	E	Sí	T
T	E	N	No	NT
NT	N	N	Sí	NT
NT	N	E	No	T
T	E	E	Sí	T
T	E	E	Sí	T
T	E	N	No	N
NT	N	N	Sí	N
NT	N	E	No	T
T	E	E	Sí	T
T	E	E	Sí	T
T	E	N	No	N
NT	N	N	Sí	T

Dado que de las 15 predicciones ha habido 10 predicciones correctas, la precisión del predictor de 1 bit ha sido del 66,67%.

Contador de 2 bits	Predicción	Resultado real	¿Predicción correcta?	Siguiente estado
ST	E	E	Sí	ST
ST	E	E	Sí	ST
ST	E	E	Sí	ST
ST	E	N	No	WT
WT	E	N	No	WN
WN	N	E	No	WT
WT	E	E	Sí	ST
ST	E	E	Sí	ST
ST	E	N	No	WT
WT	E	N	No	WN
WN	N	E	No	WT
WT	E	E	Sí	ST
ST	E	E	Sí	ST
ST	E	N	No	WT
WT	E	N	No	WN

Dado que de las 15 predicciones ha habido 7 predicciones correctas, la precisión del predictor ha sido del 46,67%.

TIPO DE EXAMEN: RESERVA - NACIONAL Y UNIÓN EUROPEA - SEPTIEMBRE 2012

Apellidos:.....Nombre:.....DNI:.....

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja, y ENTRÉGUELA junto con el resto del examen. Lea atentamente todos los enunciados. **SE PERMITE UN LIBRO Y CALCULADORA NO PROGRAMABLE.**

Problema 1 (3 puntos)

Tras añadir un nuevo procesador a un computador se logra un aumento de la velocidad de ejecución en un factor de 9. Se observa que tras aplicar esta mejora, el 55% del tiempo de ejecución se está utilizando el nuevo procesador. ¿Qué porcentaje del tiempo de ejecución original se ha reducido gracias a la mejora?

Problema 2 (4 puntos)

Considere un sencillo procesador superescalar dotado de un RRF con acceso indexado y con dos estaciones de reserva de 4 entradas asociadas, respectivamente, a una unidad de multiplicación (3 ciclos, segmentada) y a dos unidades funcionales de suma/resta (2 ciclos, ambas segmentadas). Suponga que las instrucciones siguientes:

```
i1: MULTD   F3,F1,F2
i2: ADDD    F2,F3,F1
i3: SUBD    F3,F3,F1
i4: ADDD    F5,F1,F2
```

se distribuyen, a razón de una por ciclo, a las dos estaciones de reserva y se emiten en cuanto sus operandos están disponibles. Teniendo en cuenta que se pueden emitir y terminar dos instrucciones simultáneamente, se pide:

a) Un cronograma con la secuencia temporal de ejecución de las instrucciones en el que, ciclo a ciclo, se puede apreciar cuándo se distribuyen, cuándo se emiten y cuándo finaliza su ejecución en las unidades funcionales.

b) Dibuje, ciclo a ciclo, cómo evolucionan los contenidos del ARF y del RRF para esas instrucciones si, inicialmente, `F1=2.0` y `F2=3.0`. El ARF y el RRF constan de cinco entradas. La secuenciación de los ciclos debe coincidir con la del apartado anterior.

Problema 3 (3 puntos)

1. Dibuje una red de tipo Omega de 16 entradas y 16 salidas.
2. Describa razonadamente el protocolo para enviar un mensaje desde el nodo de entrada 3 al nodo de salida 9.
3. Suponiendo que el tercer conmutador de la segunda etapa no funciona correctamente (impidiendo de esta manera cualquier tipo de conexión donde esté involucrado), indique el número y las conexiones que quedan bloqueadas, y qué porcentaje representan respecto del total de la red.

Problema 1 septiembre 2012 reserva

Tras añadir un nuevo procesador a un computador se logra un aumento de la velocidad de ejecución en un factor 9. Se observa que tras aplicar esta mejora, el 55% del tiempo de ejecución se está utilizando el nuevo procesador. ¿Qué porcentaje del tiempo de ejecución original se ha reducido gracias a la mejora?

El recurso utilizado mejora en 9 el rendimiento.

Porcentaje del tiempo de ejecución que se aplica la mejora=55%

Suponiendo que normalizamos el tiempo actual de ejecución a 1 (esto es, aplicando la mejora), el tiempo total de ejecución antes de aplicar la mejora es:

$T_{original}=0.45+(0.55*9)=5.4$

Por lo tanto, la ganancia obtenida aplicando la mejora es del 5,2. Sustituyendo en la expresión de la ganancia:

$5.4=9/(1+f(9-1))$ despejando $f = 8.3\%$

(f es la fracción del tiempo donde no se puede aplicar la mejora).

Por lo tanto, el porcentaje de tiempo que se ha convertido al modo rápido es:

(100%-8.3%)=91.7%

Problema 2 Septiembre 2012 reserva

Considere un sencillo procesador superescalar dotado de un RRF con acceso indexado y con dos estaciones de reserva de 4 entradas asociadas, respectivamente, a una unidad de multiplicación (3 ciclos, segmentada) y a dos unidades funcionales de suma/resta (2 ciclos, ambas segmentadas). Suponga que las instrucciones siguientes:

```
i1: MULTD   F3,F1,F2
i2: ADDD    F2,F3,F1
i3: SUBD    F3,F3,F1
i4: ADDD    F5,F1,F2
```

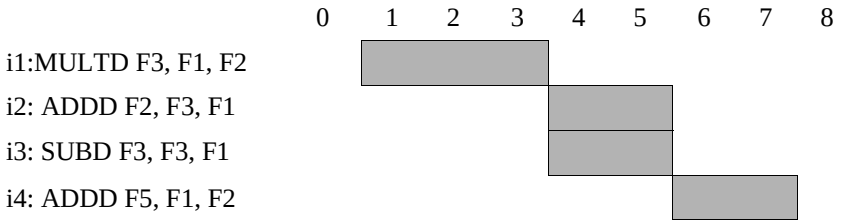
se distribuyen, a razón de una por ciclo, a las dos estaciones de reserva y se emiten en cuanto sus operandos están disponibles. Teniendo en cuenta que se pueden emitir y terminar dos instrucciones simultáneamente, se pide:

a) Un cronograma con la secuencia temporal de ejecución de las instrucciones en el que, ciclo a ciclo, se puede apreciar cuándo se distribuyen, cuándo se emiten y cuándo finaliza su ejecución en las unidades funcionales.

b) Dibuje, ciclo a ciclo, cómo evolucionan los contenidos del ARF y del RRF para esas instrucciones si, inicialmente, `F1=2.0` y `F2=3.0`. El ARF y el RRF constan de cinco entradas. La secuenciación de los ciclos debe coincidir con la del apartado anterior.

Solución

a) La secuencia temporal de ejecución de las cuatro instrucciones es la siguiente:



Se considera que en el ciclo 0 se distribuya la primera instrucción a la estación de reserva. Observe que aunque las instrucciones ya se encuentren en las estaciones de reserva listas para ser emitidas a las unidades funcionales, deben esperar a que se generen los operandos con el fin de respetar las dependencias verdaderas.

b

Ciclo 0: Llegada de i1 a la estación de reserva.
Renombramiento de F3 como Fr1.

	Datos	**Ocupado**	**Índice**
F1	2		
F2	3		
F3		1	1
F4			
F5			

	Datos	**Válido**	**Ocupado**
Fr1	--	0	1
Fr2			
Fr3			
Fr4			
Fr5			

Ciclo 1: Llegada de i2 a la estación de reserva.
Renombramiento de F2 como Fr2.
Emisión a la unidad funcional y comienzo de ejecución de i1.

	Datos	Ocupado	Índice
F1	2		
F2	3	1	2
F3		1	1
F4			
F5			

	Datos	Válido	Ocupado
Fr1		0	1
Fr2		0	1
Fr3			
Fr4			
Fr5			

Ciclo 2: Llegada de i3 a la estación de reserva.
Nuevo renombramiento de F3 como Fr3. Segundo ciclo de ejecución de i1.

	Datos	Ocupado	Índice
F1	2		
F2	3	1	2
F3		1	3
F4			
F5			

	Datos	Válido	Ocupado
Fr1		0	1
Fr2		0	1
Fr3		0	1
Fr4			
Fr5			

Ciclo 3: Llegada de i4 a la estación de reserva.
Renombramiento de F5 como Fr4. Finalización de i1.
Escritura de resultado en Fr1 y copia a estaciones de reserva para emisión de i2 e i3.

	Datos	Ocupado	Índice
F1	2		
F2	3	1	2
F3		1	3
F4			
F5		1	4

	Datos	Válido	Ocupado
Fr1	6	1	1
Fr2		0	1
Fr3		0	1
Fr4		0	1
Fr5			

Ciclo 4: Emisión de i2 e i3.
Liberación de Fr1.

	Datos	Ocupado	Índice
F1	2		
F2	3	1	2
F3		1	3
F4			
F5		1	4

	Datos	Válido	Ocupado
Fr1	6	1	0
Fr2		0	1
Fr3		0	1
Fr4		0	1
Fr5			

Ciclo 5: Finalización de i2. Escritura de resultado en Fr2.
Finalización de i3. Escritura de resultado en Fr3.
Copia del valor de Rf2 en entrada de la instrucción i4 para poder emitirla.

	Datos	Ocupado	Índice
F1	2		
F2	3	1	2
F3		1	3
F4			
F5		1	4

	Datos	Válido	Ocupado
Fr1	6	1	0
Fr2	8	1	1
Fr3	4	1	1
Fr4		0	1
Fr5			

Ciclo 6: Emisión de i4.

Escritura de resultado de Fr2 en F2. Liberación de Fr2. Escritura de resultado en Fr3 en F3. Liberación de Fr3.

	Datos	Ocupado	Índice
F1	2		
F2	8	0	2
F3	4	0	3
F4			
F5		1	4

	Datos	Válido	Ocupado
Fr1	6	0	0
Fr2	8	1	0
Fr3	4	1	0
Fr4		0	1
Fr5			

Ciclo 7: Finalización de i4.

Escritura de resultado en Fr4.

	Datos	Ocupado	Índice
F1	2		
F2	8	0	2
F3	4	0	3
F4			
F5		1	4

	Datos	Válido	Ocupado
Fr1	6	0	0
Fr2	8	1	0
Fr3	4	1	0
Fr4	10	1	1
Fr5			

Ciclo 8: Escritura de Fr4 en F5.

Liberación de Fr4.

	Datos	Ocupado	Índice
F1	2	0	
F2	18	0	2
F3	4	0	3
F4		0	
F5	36	0	4

	Datos	Válido	Ocupado
Fr1	6	0	0
Fr2	18	1	0
Fr3	4	1	0
Fr4	36	1	1
Fr5	--	0	1

Problema 1 (3 puntos)

Mostrar la evolución de los registros en coma flotante (FR) y de las estaciones de reserva (RS) para todos los ciclos que sean necesarios en la ejecución, utilizando el algoritmo de Tomasulo, del siguiente fragmento de código:

```
i1:  MULTD  F0,F6,F2
i2:  ADDD   F4,F2,F6
i3:  ADDD   F6,F4,F0
i4:  ADDD   F2,F6,F0
```

Considere las siguientes hipótesis de partida:

- Para reducir el número de ciclos máquina se permite que la FLOS distribuya hasta dos instrucciones en cada ciclo según el orden del programa.
- Una instrucción puede comenzar su ejecución en el mismo ciclo en que se distribuye a una estación de reserva.
- La operación suma tiene una latencia de un ciclo y la de multiplicación de dos ciclos.
- Se permite que una instrucción reenvíe su resultado a instrucciones dependientes durante su último ciclo de ejecución. De esta forma una instrucción a la espera de un resultado puede comenzar su ejecución en el siguiente ciclo si detecta una coincidencia.
- Los valores de etiqueta 01, 02 y 03 se utilizan para identificar las tres estaciones de reserva de la unidad funcional de suma, mientras que 04 y 05 se utilizan para identificar las dos estaciones de reserva de la unidad funcional de multiplicación/división. Estos valores de etiqueta son los ID de las estaciones de reserva.
- Inicialmente, el valor de los registros es F0=2.0, F2=2.5, F4=8.0 y F6=4.0.

Problema 2 (4 puntos)

Dado el siguiente código:

```
inicio: LD    F0,0(R2)    % i1
        LD    F2,0(R4)    % i2
        LD    F4,0(R6)    % i3
        ADDD  F0,F0,F2    % i4
        ADDD  F0,F0,F4    % i5
        DIVD  F0,F0,F8    % i6
        SD    0(R1),F0    % i7
        ADDI  R2,R2,#8    % i8
        ADDI  R4,R4,#8    % i9
        ADDI  R6,R6,#8    % i10
        ADDI  R1,R1,#8    % i11
        JUMP  inicio      % i12
```

a) Obtenga el diagrama de flujo de datos de la secuencia. Considere que todas las instrucciones tienen una latencia de 2 ciclos.

b) Transforme la secuencia en instrucciones VLIW para que pueda ser ejecutada por un procesador con un formato de instrucción que permite emitir simultáneamente operaciones a cualquiera de las cuatro unidades funcionales (son polivalentes). Considere las mismas latencias que en el apartado anterior y que puede efectuar los reordenamientos que crea oportunos, siempre que no afecten al resultado.

c) Desenrolle el bucle original 4 veces y planifíquelo en forma de instrucciones VLIW teniendo en cuentas las características del procesador y las latencias. Utilice todos los registros que sean necesarios y compacte el código VLIW lo máximo posible.

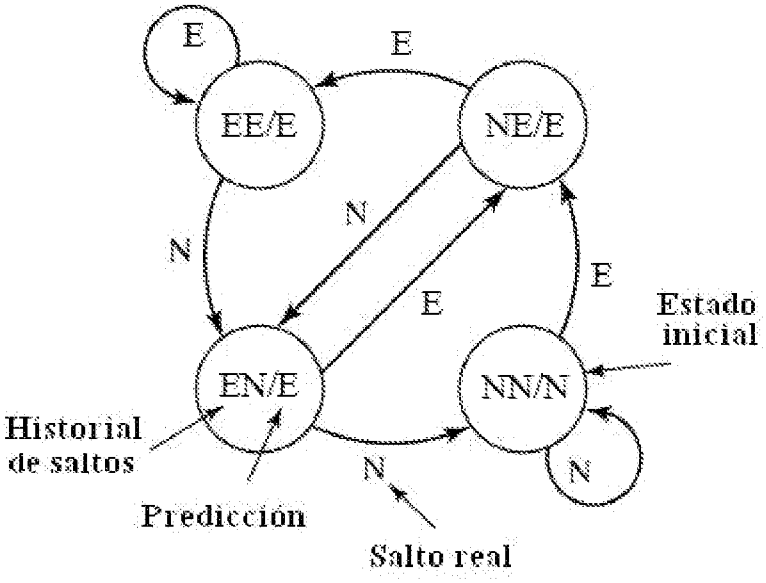
d) ¿Qué mejora en el rendimiento se ha obtenido si se compara el código del apartado b con el del c?

Problema 3 (3 puntos)

Considere el siguiente segmento de código que se ejecuta en el cuerpo principal de un bucle:

```
if (x es par) then                  // salto S1
      incrementar a;                // salto S1 tomado
if (x es múltiplo de 10) then       // salto S2
      incrementar b;                // salto S2 tomado
```

y que la siguiente lista de 9 valores para la variable `x` es procesada en 9 iteraciones del bucle: 8, 9, 10, 11, 12, 20, 29, 30, 31. La máquina de estados situada a continuación representa una ligera variante de un predictor de Smith de 2 bits de historial y se utiliza para predecir la ejecución de los saltos que hay en el código.



¿Cuál es la secuencia de predicciones para los salto `S1` y `S2` en cada iteracción del bucle? Tenga en cuenta que el historial de cada salto es exclusivo de ese salto y no se debe mezclar con el historial del otro salto.

Problema 1 - 2013 febrero

Mostrar la evolución de los Registros en coma flotante (FR) y las estaciones de Reserva (RS) para todos los ciclos que sean necesarios en la ejecución del siguiente fragmento de código utilizando el algoritmo de Tomasulo.

```
i1: MULTD F0,F6,F2
i2: ADDD  F4,F2,F6
i3: ADDD  F6,F4,F0
i4: ADDD  F2,F6,F0
```

- Considere las siguientes hipótesis de partida:
- Para reducir el número de ciclos máquina se permite que la FLOS distribuya hasta dos instrucciones en cada ciclo según el orden del programa.
- Una instrucción puede comenzar su ejecución en el mismo ciclo en que se distribuye a una estación de reserva.

- La operación **suma** tiene una latencia de **un ciclo** y la de **multiplicación** de **dos ciclos**.
- Se permite que una instrucción reenvíe su resultado a instrucciones dependientes durante su último ciclo de ejecución. De esta forma una instrucción a la espera de un resultado puede comenzar su ejecución en el siguiente ciclo si se detecta una coincidencia.

- Los valores de etiqueta 01, 02 y 03 se utilizan para identificar las tres estaciones de reserva de la unidad funcional de suma, mientras que 04 y 05 se utilizan para identificar las dos estaciones de reserva de la unidad funcional de multiplicación/división. Estos valores de etiqueta son los ID de las estaciones de reserva.
- Inicialmente, el valor de los registros es F0=2.0, F2=2.5, F4=8.0 y F6=4.0.

i1: MULTD F0,F6,F2
i2: ADDD F4,F2,F6
i3: ADDD F6,F4,F0
i4: ADDD F2,F6,F0

FR

	bitOc.	etiqueta	dato
F0			2.0
F2			2.5
F4			8.0
F6			4.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
02				
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

Ciclo 1: Se distribuye i1 e i2
i1: MULTD F0,F6,F2
i2: ADDD F4,F2,F6
Se ejecuta RS 01 1/1
Se ejecuta RS 04 1/2
Se envía RS 01: 6.5 al CDB

FR

	bitOc.	etiqueta	dato
F0	Si	04	2.0
F2			2.5
F4	Si	01	8.0
F6			4.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i2: 01	00	2.5	00	4.0
02				
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i1: 04	00	4.0	00	2.5
05				

MULT/DIV

Ciclo 2: Se distribuye i3 e i4
i3: ADDD F6,F4,F0
i4: ADDD F2,F6,F0
Se actualiza el valor de RS 01: 6.5
Se vacía RS 01
Se ejecuta RS 04 2/2
Se envía RS 04: 10.0 al CDB
NO se ejecuta RS 02, 03

FR

	bitOc.	etiqueta	dato
F0	Si	04	2.0
F2	Si	03	2.5
F4			6.5
F6	Si	02	4.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
i3: 02	04	xx	00	6.5
i4: 03	02	xx	00	6.5

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i1: 04	00	4.0	00	2.5
05				

MULT/DIV

Ciclo 3: Se actualiza el valor de RS 04: 10.0
Se vacía RS 04
Se ejecuta RS 02 1/1
Se envía RS 02: 16.5 al CDB
NO se ejecuta RS 03

FR

	bitOc.	etiqueta	dato
F0			10.0
F2	Si	03	2.5
F4			6.5
F6	Si	02	4.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
i3: 02	04	10.0	00	6.5
i4: 03	02	xx	00	6.5

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

Ciclo 4: Se actualiza el valor de RS 02: 16.5
Se vacía RS 02
Se ejecuta RS 03 1/1
Se envía RS 03: 23.0 al CDB

FR

	bitOc.	etiqueta	dato
F0			10.0
F2	Si	03	2.5
F4			6.5
F6			16.5

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
02				
i4: 03	02	16.5	00	6.5

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

Ciclo 5: Se actualiza el valor de RS 03: 23.0
Se vacía RS 03

FR

	bitOc.	etiqueta	dato
F0			10.0
F2			23.0
F4			6.5
F6			16.5

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
02				
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

Problema 2 Febrero 2013 1ª Semana

Dado el siguiente código:

```
inicio: LD    F0,0(R2)    % i1
        LD    F2,0(R4)    % i2
        LD    F4,0(R6)    % i3
        ADDD  F0,F0,F2    % i4
        ADDD  F0,F0,F4    % i5
        DIVD  F0,F0,F8    % i6
        SD    0(R1),F0    % i7
        ADDI  R2,R2,#8    % i8
        ADDI  R4,R4,#8    % i9
        ADDI  R6,R6,#8    % i10
        ADDI  R1,R1,#8    % i11
        JUMP  inicio      % i12
```

a) Obtenga el diagrama de flujo de datos de la secuencia. Considere que todas las instrucciones tienen una latencia de 2 ciclos.

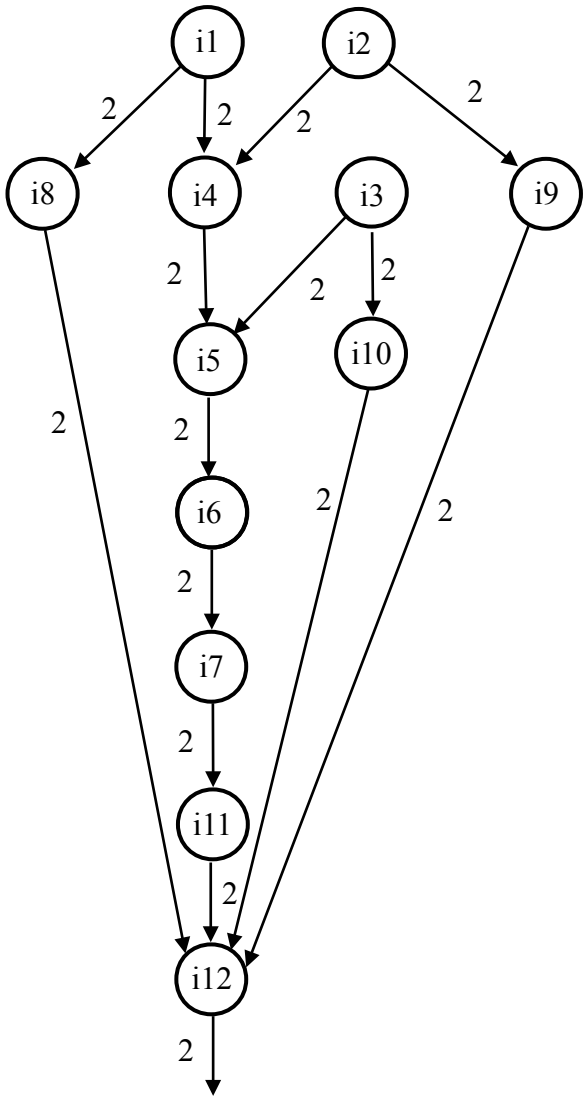
b) Transforme la secuencia en instrucciones VLIW para que pueda ser ejecutada por un procesador con un formato de instrucción que permite emitir simultáneamente operaciones a cualquiera de las cuatro unidades funcionales (son polivalentes). Considere las mismas latencias que en el apartado anterior y que puede efectuar los reordenamientos que crea oportunos, siempre que no afecten al resultado.

c) Desenrolle el bucle original 4 veces y planifíquelo en forma de instrucciones VLIW teniendo en cuentas las características del procesador y las latencias. Utilice todos los registros que sean necesarios y compacte el código VLIW lo máximo posible.

d) ¿Qué mejora en el rendimiento se ha obtenido si se compara el código del apartado b con el del c?

Solución

a)



b) Son posibles varias soluciones según se realice el ordenamiento de las operaciones en las instrucciones. En la solución que se propone, se han agrupado las instrucciones de incremento de los índices con la salvedad de R1. Si se optase por incrementar R1 antes de la instrucción de almacenamiento SD 0(R1), F0 sería necesario realizar un decremento negativo para compensar, es decir, SD -8(R1), F0.

	Unidad funcional 1	**Unidad funcional 2**	**Unidad funcional 3**	**Unidad funcional 4**
inicio:	LD F0, 0(R2)	LD F2, 0(R4)	LD F4, 0(R6)	ADDI R2, R2, #8
				ADDI R4, R4, #8
	ADDD F0, F0, F2			ADDI R6, R6, #8
	ADDD F0, F0, F4			
	DIVD F0, F0, F8			
	SD 0(R1), F0	ADDI R1, R1, #8		JMP inicio

c) La secuencia de código que se obtiene tras desenrrollar 4 veces el bucle original es la siguiente:

```
inicio: LD    F0,0(R2)          // Iter 1
        LD    F2,0(R4)          // Iter 1
        LD    F4,0(R6)          // Iter 1
        LD    F10,8(R2)         // Iter 2
        LD    F12,8(R4)         // Iter 2
        LD    F14,8(R6)         // Iter 2
        LD    F20,16(R2)        // Iter 3
        LD    F22,16(R4)        // Iter 3
        LD    F24,16(R6)        // Iter 3
        LD    F30,24(R2)        // Iter 4
        LD    F32,24(R4)        // Iter 4
        LD    F34,24(R6)        // Iter 4
        ADDD  F0,F0,F2          // Iter 1
        ADDD  F0,F0,F4          // Iter 1
        DIVD  F0,F0,F8          // Iter 1
        ADDD  F10,F10,F12       // Iter 2
        ADDD  F10,F10,F14       // Iter 2
        DIVD  F10,F10,F8        // Iter 2
        ADDD  F20,F20,F22       // Iter 3
        ADDD  F20,F20,F24       // Iter 3
        DIVD  F20,F20,F8        // Iter 3
        ADDD  F30,F30,F32       // Iter 4
        ADDD  F30,F30,F34       // Iter 4
        DIVD  F30,F30,F8        // Iter 4
        SD    0(R1),F0          // Iter 1
        SD    8(R1),F10         // Iter 2
        SD    16(R1),F20        // Iter 3
        SD    24(R1),F30        // Iter 4
        ADDI  R2,R2,#32
        ADDI  R4,R4,#32
        ADDI  R6,R6,#32
        ADDI  R1,R1,#32
        JUMP  inicio
```

Una transformación en código VLIW realizando la máxima compactación posible se muestra en la siguiente tabla. Los colores representan: 1ª iteración (negro) e instrucciones auxiliares, 2ª iteración (rojo), 3ª iteración (verde), 4ª iteración (morado).

	Unidad funcional 1	Unidad funcional 2	Unidad funcional 3	Unidad funcional 4
inicio:	LD F0, 0(R2)	LD F2, 0(R4)	LD F10, 8(R2)	LD F12, 8(R4)
	LD F20, 16(R2)	LD F22, 16(R4)	LD F30, 24(R2)	LD F32, 24(R4)
	ADDD F0, F0, F2	LD F4, 0(R6)	ADDD F10, F10, F12	LD F14, 8(R6)
	ADDD F20, F20, F22	LD F24, 16(R6)	ADDD F30, F30, F32	LD F34, 24(R6)
	ADDD F0, F0, F4		ADDD F10, F10, F14	
	ADDD F20, F20, F24		ADDD F30, F30, F34	
	DIVD F0, F0, F8		DIVD F10, F10, F8	ADDI R2, R2, #32
	DIVD F20, F20, F8		DIVD F30, F30, F8	ADDI R4, R4, #32
	SD 0(R1), F0		SD 8(R1), F10	ADDI R6, R6, #32
	SD 16(R1), F20	ADDI R1, R1, #32	SD 24(R1), F30	JMP inicio

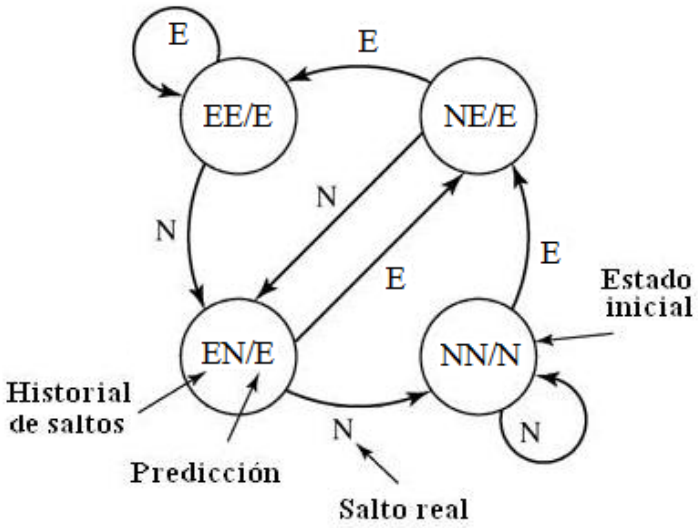
d) El código consume 10 instrucciones VLIW ocupando un total de 160 bytes (10 instrucciones * 16 bytes por instrucción). En comparación con el código VLIW sin desenrrollar, el rendimiento, prácticamente, se ha incrementado por 4 ya que el número de ciclos por iteración es de 11 en comparación con el caso sin desenrrollar que es 10 ciclos pero procesando 1 único elemento, y no 4 como en el caso que nos ocupa.

Problema 3 Febrero 2013 1ª Semana

Considere el siguiente segmento de código que se ejecuta en el cuerpo principal de un bucle:

```
if (x es par) then                  // salto S1
      incrementar a;                // salto S1 tomado
if (x es múltiplo de 10) then       // salto S2
      incrementar b;                // salto S2 tomado
```

y que la siguiente lista de 9 valores para la variable `x` es procesada en 9 iteraciones del bucle: 8, 9, 10, 11, 12, 20, 29, 30, 31. La máquina de estados situada a continuación representa una ligera variante de un predictor de Smith de 2 bits de historial y se utiliza para predecir la ejecución de los saltos que hay en el código.



¿Cuál es la secuencia de predicciones para los salto `S1` y `S2` en cada iteracción del bucle? Tenga en cuenta que el historial de cada salto es exclusivo de ese salto y no se debe mezclar con el historial del otro salto.

Solución

Para la resolución del ejercicio es fundamental entender el diagrama de estados que muestra el enunciado. Básicamente, el diagrama indica en función del resultado del salto actual y del historial de los dos saltos previos, la predicción de lo que va a ocurrir con el siguiente salto. Es clave separar los historiales de cada instrucción de salto, es decir, la evolución de la predicción de cada salto es independiente del otro.

Páginas 227 y 228 y ejercicio 5.7 del texto base de la asignatura.

En la siguiente secuencia se puede apreciar la diferencia entre la predicción que se realiza de la efectividad o no del salto S1 y S2 y la situación real que se produce.

	8	**9**	**10**	**11**	**12**	**20**	**29**	**30**	**31**
Estado:	NN	NE	EN	NE	EN	NE	EE	EN	NE
S1 predicho:	**N**	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**E**
S1 real:	E	N	E	N	E	E	N	E	N
Estado:	NN	NN	NN	NE	EN	NN	NE	EN	NE
S2 predicho:	**N**	**N**	**N**	**E**	**E**	**N**	**E**	**E**	**E**
S2 real:	N	N	E	N	N	E	N	E	N

Para el salto S1:

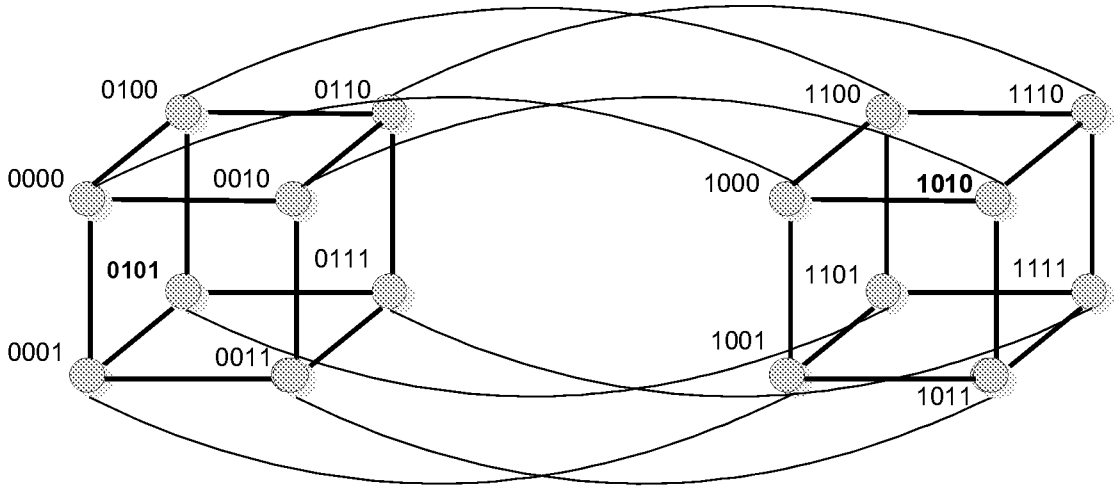
Inicialmente, el autómata se encuentra en el estado NN, es decir, predice que el siguiente salto no se tomará (N). El primer valor de `x` es 8 (par) lo que provoca que el salto sea efectivo, con lo que se alimenta al autómata con el valor E y se avanza al estado NE, indicando que la predicción para el próximo salto es que se produzca (salida E). Estando en ese estado, el valor de `x` pasa a ser 9 por lo que no se produce el salto (entrada N al autómata) y se pasa al estado EN que predice que el próximo salto sí se producirá (salida E). Así se sigue hasta completar la secuencia de valores de `x`.

Para el salto S2:

Igual que en el caso anterior, el autómata se encuentra en el estado NN pero permanece en ese estado hasta que x es 10; en ese momento salta al estado NE y…..

Problema 1 (3 puntos)

Dada la siguiente red estática:



a) ¿Qué tipo de red es?
b) Se desea transmitir un mensaje desde el procesador a = 0101 al procesador b = 1010. Si se comienza a buscar el camino por el bit menos significativo, explique razonadamente cuál es el camino que debe seguir el mensaje.
c) Dibuje una red estática con los siguientes valores en sus parámetros:

Diámetro = 4 Conectividad de arco = 4 Coste = 50

Problema 2 (3 puntos)

Dado el siguiente fragmento de código:

```
ADDI R5,R0,#1
LD   R6,0(R5)
LD   R8,8(R5)
LD   R9,16(R5)
LD   R7,24(R5)
ADD  R1,R8,R9
ADD  R8,R9,R7
SD   0(R8),R1
LD   R8,0(R6)
```

a) Señale todas las dependencias de datos existentes en el fragmento.
b) ¿Existen dependencias de memoria? En caso afirmativo indique cuáles y a qué se deben.
c) Renombre el código e indique qué dependencias permanecen.
d) ¿Cómo se gestionan en un procesador superescalar las dependencias de datos y de memoria que permanecen tras el renombramiento?

Problema 3 (4 puntos)

En un procesador vectorial con las siguientes características:

- Registros con una longitud vectorial máxima de 64 elementos.
- Una unidad de suma vectorial con tiempo de arranque de 6 ciclos.
- Una unidad de multiplicación con tiempo de arranque de 7 ciclos.
- Una unidad de carga/almacenamiento con tiempo de arranque de 12 ciclos.
- La frecuencia de trabajo del procesador es 100 MHz.
- T_{base} de 10 ciclos y T_{bucle} de 15 ciclos.

se pretende ejecutar el siguiente código vectorial para un vector con una longitud de 64 elementos:

```
LV      V1,R1
MULTV   V2,V1,V3
MULTSV V4,V2,F0
SV      R2,V4
```

a) Suponiendo que no existe encadenamiento entre las unidades, calcule R_{100}, T_{100} y R_{∞}.

b) Una medida habitual del impacto de los costes adicionales es $N_{1/2}$ que es la longitud del vector necesaria para alcanzar la mitad del valor de R_{∞}. Calcule $N_{1/2}$.

c) Ahora dispone de encadenamiento entre las unidades funcionales y solapamiento entre convoyes dentro de la misma iteración. ¿Cuál es el nuevo valor de ciclos por elemento?

Problema 2 Febrero 2013 2ª Semana

Dado el siguiente fragmento de código:

```
ADDI R5,R0,#1
LD   R6,0(R5)
LD   R8,8(R5)
LD   R9,16(R5)
LD   R7,24(R5)
ADD  R1,R8,R9
ADD  R8,R9,R7
SD   0(R8),R1
LD   R8,0(R6)
```

a) Señale todas las dependencias de datos existentes en el fragmento.
b) ¿Existen dependencias de memoria? En caso afirmativo indique cuáles y a qué se deben.
c) Renombre el código e indique qué dependencias permanecen.
d) ¿Cómo se gestionan en un procesador superescalar las dependencias de datos y de memoria que permanecen tras el renombramiento?

Solución

a) En el fragmento de código, las dependencias de datos existentes son:

```
i1: ADDI   R5,R0,#1
i2: LD     R6,0(R5)      // dependencia RAW con i1 por R5
i3: LD     R8,8(R5)      // dependencia RAW con i1 por R5
i4: LD     R9,16(R5)     // dependencia RAW con i1 por R5
i5: LD     R7,24(R5)     // dependencia RAW con i1 por R5
i6: ADD    R1,R8,R9      // dependencia RAW con i3 por R8
                         // dependencia RAW con i4 por R9
i7: ADD    R8,R9,R7      // dependencia RAW con i4 por R9
                         // dependencia RAW con i5 por R7
                         // dependencia WAR con i6 por R8
                         // dependencia WAW con i3 por R8
i8: SD     0(R8),R1      // dependencia RAW con i6 por R1
                         // dependencia RAW con i3 por R8
                         // dependencia RAW con i7 por R8
i9: LD     R8,0(R6)      // dependencia RAW con i2 por R6
                         // dependencia WAW con i3 por R8
                         // dependencia WAW con i7 por R8
                         // dependencia WAR con i6 por R8
                         // dependencia WAR con i8 por R8
```

b) Existen dependencias ambiguas de memoria de las instrucciones `i2`, `i3`, `i4` e `i5` con `i8`. Las instrucciones de carga `i2`, `i3`, `i4` e `i5` leen las posiciones de memoria `M[0+R5]`, `M[8+R5]`, `M[16+R5]`, y `M[24+R5]`, respectivamente, mientras que la instrucción de almacenamiento `i8` tiene que escribir en `M[0+R8]`, lo que implica la existencia de un riesgo WAR.

Entre la instrucción `i8` e `i9` existe otra dependencia ambigua de tipo RAW ya que se puede dar el caso de que el contenido de `R8` y `R5` coincidan y, por lo tanto, el destino y fuente de ambas instrucciones.

c) El código con los registros renombrados y en el que han desaparecido las dependencias falsas de datos es:

```
i1: ADDI   Rr1,R0,#1
i2: LD     Rr2,0(Rr1)    // dependencia RAW con i1 por Rr1
i3: LD     Rr3,8(Rr1)    // dependencia RAW con i1 por Rr1
i4: LD     Rr4,16(Rr1)   // dependencia RAW con i1 por Rr1
i5: LD     Rr5,24(Rr1)   // dependencia RAW con i1 por Rr1
i6: ADD    Rr6,Rr3,Rr4   // dependencia RAW con i3 por Rr3
                         // dependencia RAW con i4 por Rr4
i7: ADD    Rr7,Rr4,Rr5   // dependencia RAW con i4 por Rr4
                         // dependencia RAW con i5 por Rr5
i8: SD     0(Rr7),Rr6    // dependencia RAW con i6 por Rr6
                         // dependencia RAW con i7 por Rr7
i9: LD     Rr9,0(Rr2)    // dependencia RAW con i3 por Rr3
                         // dependencia RAW con i2 por Rr2
```

d) Las dependencias de datos RAW se elimina mediante el mecanismo que establecen las estaciones de reserva y que obligan a que los operandos están disponibles para poder emitir una instrucción.

El renombramiento de registros no elimina ninguna de las dependencias de memoria ya que al emitirse la instrucciones no es posible conocer si hay coincidencia en las direcciones. Para eliminar las dependencias de memoria falsas se establece el mecanismo de terminación ordenada con el buffer de almacenamiento. Las WAW se evitan, claramente, mediante un almacenamiento ordenado. Los riesgos WAR se evitan garantizando que una instrucción de almacenamiento posterior en el código a una carga, no escribe antes en memoria gracias al almacenamiento diferido.

Los riesgos de memoria RAW se gestionan mediante hardware adicional que permite la detección de la coincidencia de memoria y el reenvío del dato del almacenamiento (origen) hacia la carga (destino).

Problema 3 Febrero 2013 2

En un procesador vectorial con las siguientes características:

- Registros con una longitud vectorial máxima de 64 elementos.
- Una unidad de suma vectorial con tiempo de arranque de 6 ciclos.
- Una unidad de multiplicación con tiempo de arranque de 7 ciclos.
- Una unidad de carga/almacenamiento con tiempo de arranque de 12 ciclos.
- La frecuencia de trabajo del procesador es 100 MHz.
- T_{base} de 10 ciclos y T_{bucle} de 15 ciclos.

se pretende ejecutar el siguiente código vectorial para un vector con una longitud de 64 elementos:

```
LV     V1,R1
MULTV  V2,V1,V3
MULTSV V4,V2,F0
SV     R2,V4
```

a) Suponiendo que no existe encadenamiento entre las unidades, calcule R_{100}, T_{100} y R_{∞}.

b) Una medida habitual del impacto de los costes adicionales es $N_{1/2}$ que es la longitud del vector necesaria para alcanzar la mitad del valor de R_{∞}. Calcule $N_{1/2}$.

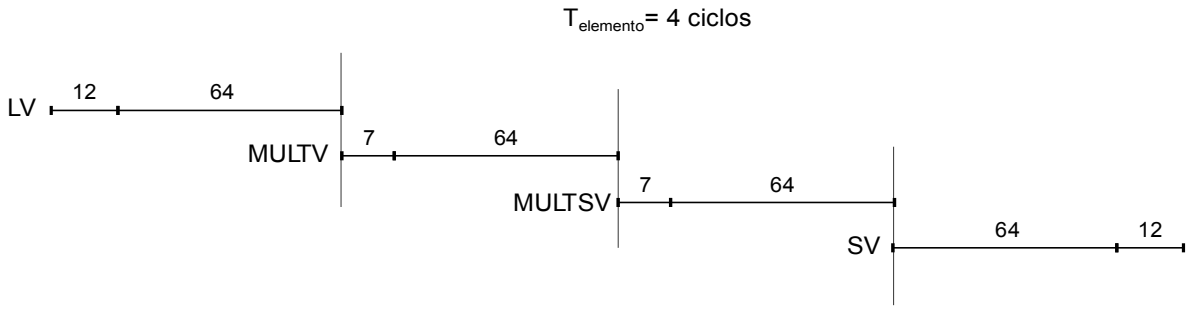
c) Ahora dispone de encadenamiento entre las unidades funcionales y solapamiento entre convoyes dentro de la misma iteración. ¿Cuál es el nuevo valor de ciclos por elemento?

Solución

a) Analizando los riesgos estructurales se obtiene el siguiente código vectorial:

Convoy 1:	LV	V1, R1
Convoy 2:	MULTV	V2, V1, V3
Convoy 3:	MULTSV	V4, V2, F0
Convoy 5:	SV	R2, V4

La secuencia de ejecución de los cuatro convoyes si se considera que VLR es 64 es la que se muestra en la siguiente figura.



Dado que hay 4 convoyes, T*elemento* es 4 ciclos y el T*arranque* total es igual a la suma de los tiempos de arranque visibles de los cuatro convoyes. Esto es

T*arranque* = T*arranque* LV + 2*T*arranque* MULTV + T*arranque* SV

T*arranque* = (12 + 2*7 + 12) ciclos = 38 ciclos

Sustituyendo los valores conocidos de T*arranque* y T*elemento* en la expresión que determina el tiempo de ejecución de un bucle vectorizado para vectores de longitud *n* se tiene

$$T_n = 10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15 + 38) + 4 \cdot n$$

que para el caso particular de *n*=100 es

$$T_{100} = 10 + \left\lceil \frac{100}{64} \right\rceil \cdot (15 + 38) + 4 \cdot 100$$

$$T_{100} = 10 + 2 \cdot (15 + 38) + 4 \cdot 100$$

$$T_{1000} = 516 \text{ ciclos}$$

$$R_{100} = \frac{2 \cdot 100}{T_{100}} = \frac{200}{516} = 0.3875 \, {}^{FLOP}\!/_{ciclo}$$

El R_∞ expresado en FLOP/ciclo es

$$R_\infty = \lim_{n \to \infty} \left(\frac{2 \cdot n}{\mathrm{T_n}} \right)$$

$$R_\infty = \lim_{n \to \infty} \left(\frac{2 \cdot n}{10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15 + 38) + 4 \cdot n} \right)$$

Para simplificar los cálculos, la expresión $\lceil n/64 \rceil$ se puede reemplazar por una cota superior dada por $(n/64 + 1)$. Sustituyendo esta cota en R_∞ y teniendo en cuenta que el número de operaciones vectoriales que se realizan es de dos, una multiplicación vectorial y una multiplicación vectorial-escalar, se tiene

$$R_\infty = \lim_{n \to \infty} \left(\frac{2 \cdot n}{10 + \left(\frac{n}{64} + 1 \right) \cdot (15 + 38) + 4 \cdot n} \right)$$

$$R_\infty = \lim_{n \to \infty} \left(\frac{2 \cdot n}{63 + 4{,}828 \cdot n} \right)$$

$$R_\infty = 0{,}41425 \text{ FLOP/ciclo}$$

Para expresar R_∞ en FLOPS, habría que multiplicar el valor en FLOP/ciclo por la frecuencia del procesador. Se tendría así

$$R_\infty = 0{,}41425 \text{ FLOP/ciclo} \cdot (100 \cdot 10^6) \text{Hz}$$

$$R_\infty = 41{,}425 \text{ MFLOPS}$$

b) Tal y como indica el enunciado, $N_{1/2}$ es el número de elementos tal que $R_{N_{1/2}} = \frac{R_\infty}{2}$. Tendremos así:

$$R_{N_{1/2}} = \frac{R_\infty}{2} = \frac{0{,}41425 \text{ FLOP/ciclo}}{2} = 0{,}207125 \text{ FLOP/ciclo}$$

Por otra parte, sabemos que:

$$R_n = \frac{(\text{Operaciones en coma flotante} \cdot n \text{ elementos})}{T_n}$$

$$R_n = \frac{2 \cdot n}{10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15 + 38) + 4 \cdot n}$$

$$R_n = \frac{2 \cdot n}{10 + \left(\frac{n}{64} + 1 \right) \cdot (15 + 38) + 4 \cdot n}$$

$$R_n = \frac{2 \cdot n}{63 + 4{,}828 \cdot n}$$

Basta con igualar las expresiones anteriores de $R_{N_{1/2}}$ y R_n para obtener el valor de *n* que las satisface:

$$\frac{2 \cdot n}{63 + 4{,}828 \cdot n} = 0{,}207125 \text{ FLOP/ciclo}$$

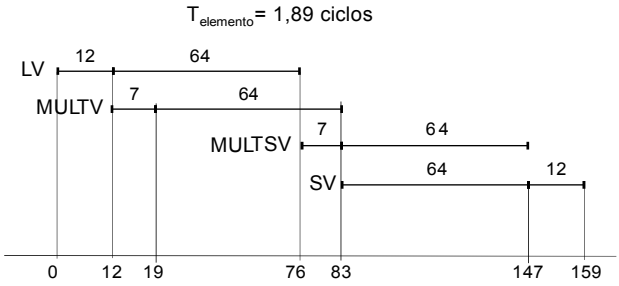
$$n = 13{,}05$$

Por lo tanto, $N_{1/2} = 13$.

c) Para calcular $T_{elemento}$ en situaciones con solapamiento hay que recordar que:

$$T_{elemento} = (T_n - T_{arranque}) / n$$

Por un lado, tenemos el siguiente diagrama de ejecución:



que nos indica que el tiempo total de ejecución utilizando un VLR de 64 es 159 ciclos.

El tiempo de arranque total visible es:

T*arranque* = T*arranque* LV + 2*T*arranque* MULTV + T*arranque* SV

T*arranque* = (12 + 2*7 + 12) ciclos = 38 ciclos

Por lo tanto, se tiene:

$$T_{elemento} = (T_n - T_{arranque}) / n$$

$$T_{elemento} = (159 - 38) / 64$$

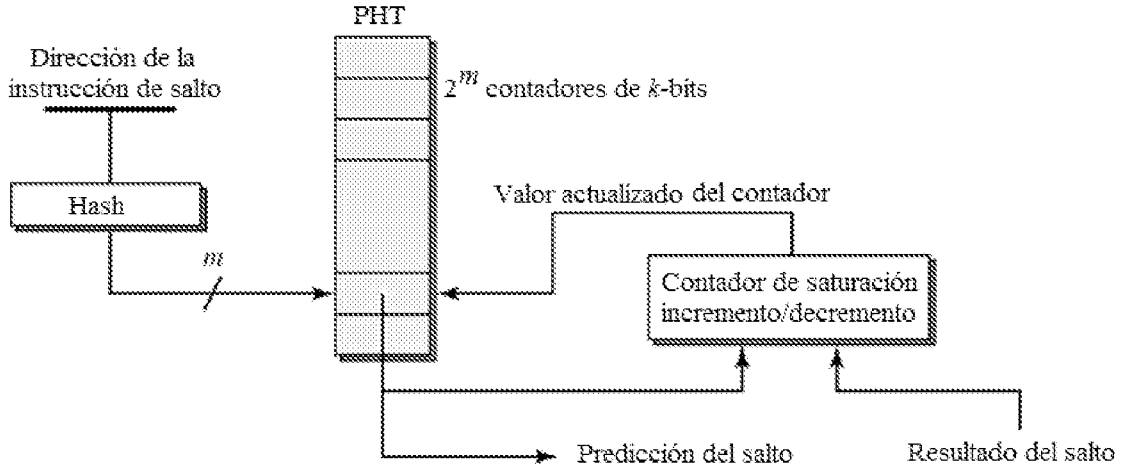
$$T_{elemento} = 1{,}89$$

TIPO DE EXAMEN: ORIGINAL – NACIONAL/UNIÓN EUROPEA - SEPTIEMBRE

INSTRUCCIONES: Lea atentamente todos los enunciados. **SE PERMITE UN LIBRO CON ANOTACIONES Y SUBRAYADOS Y CALCULADORA NO PROGRAMABLE.**

Problema 1 (3 puntos)

Sea un predictor dinámico de saltos basado en el algoritmo de Smith, cuya estructura se muestra en la siguiente figura:



La función *hash* para acceder a la tabla de contadores a partir de la dirección destino del salto es (PC *mod* 2^m). Dado el siguiente fragmento de código que se ejecuta dentro de un bucle:

```
if (x es impar) then {          // salto S1 (PC = 0x0000)
    incrementar a;              // salto S1 tomado
    if (x es primo) then        // salto S2 (PC= 0x1001)
    incrementar a;              // salto S2 tomado
}
if (x es par) then              // salto S3 (PC = 0x0110)
    incrementar a;              // salto S3 tomado
```

y la siguiente lista de valores para la variable `x`, la cual es procesada en nueve iteraciones del bucle:

```
8, 9, 10, 11, 12, 17, 20, 29, 31
```

se pide que mediante una tabla indique la evolución de los contadores afectados por la ejecución del predictor si $m = 3$ y $k = 2$. En la tabla especifique el contador que corresponde a cada salto, el valor del contador en binario, la predicción y el resultado de cada salto para las nueve iteraciones. El estado inicial de la tabla es con todos los contadores a `00`, es decir, se predice el salto como fuertemente no efectivo (SN – *strongly not taken*).

Problema 2 (4 puntos)

Dado el siguiente código:

```
inicio:  LD    F0,0(R2)
         LD    F2,0(R4)
         LD    F4,0(R6)
         ADDD  F0,F0,F2
         ADDD  F0,F0,F4
         DIVD  F0,F0,F8
         SD    0(R1),F0
         ADDI  R2,R2,#8
         ADDI  R4,R4,#8
         ADDI  R6,R6,#8
         ADDI  R1,R1,#8
         JUMP  inicio
```

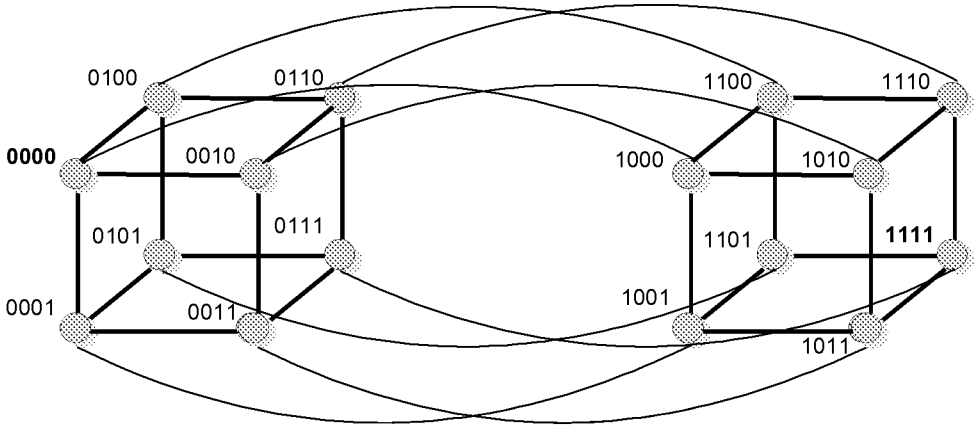
a) Genere la secuencia de código vectorial equivalente sin incluir las instrucciones escalares necesarias para recorrer los vectores.

b) Calcule el Tarranque y el Telemento para ejecutar la secuencia de código vectorial si dispone de una unidad de carga/almacenamiento (12 ciclos de Tarranque), una unidad de suma vectorial (6 ciclos) y otra de división vectorial (7 ciclos) en los siguientes supuestos:

- Sin encadenamiento.
- Con encadenamiento.
- Con solapamiento entre convoyes.

Problema 3 (3 puntos)

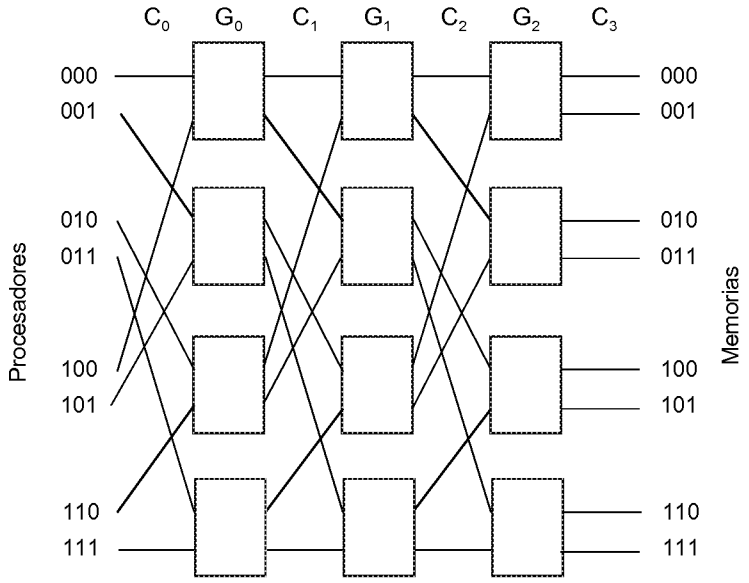
Dada la siguiente red estática:



1. Se desea transmitir un mensaje desde el procesador a = `0000` al procesador b = `1111`. Si se comienza a buscar el camino por el bit menos significativo, explique razonadamente cuál es el camino que debe seguir el mensaje.
2. Dibuje una red estática con los siguientes valores en sus parámetros:

Diámetro = 6 Conectividad de arco = 4 Coste = 98

Dada la siguiente red dinámica:



3. ¿De qué red se trata? Explique razonadamente la conmutación de cada conmutador de la red para enviar un mensaje del procesador 101 al banco de memoria 100.

TIPO DE EXAMEN: RESERVA – NACIONAL/ UNIÓN EUROPEA - SEPTIEMBRE

INSTRUCCIONES: Lea atentamente todos los enunciados. **SE PERMITE UN LIBRO CON ANOTACIONES Y SUBRAYADOS Y CALCULADORA NO PROGRAMABLE.**

Problema 1 (3 puntos)

Utilizando el algoritmo de Tomasulo, muestre la evolución de los registros en coma flotante (FR) y las estaciones de reserva (RS) para todos los ciclos que sean necesarios en la ejecución del siguiente fragmento de código:

```
i1:  MULTD   F2,F2,F6
i2:  MULTD   F4,F2,F6
i3:  ADDD    F2,F4,F6
i4:  ADDD    F6,F2,F6
```

Considere las siguientes hipótesis de partida:

- Para reducir el número de ciclos máquina se permite que la FLOS distribuya hasta dos instrucciones en cada ciclo según el orden del programa.
- Una instrucción puede comenzar su ejecución en el mismo ciclo en que se distribuye a una estación de reserva.
- La operación suma tiene una latencia de dos ciclos y la de multiplicación de tres ciclos.
- Se permite que una instrucción reenvíe su resultado a instrucciones dependientes durante su último ciclo de ejecución. De esta forma una instrucción a la espera de un resultado puede comenzar su ejecución en el siguiente ciclo si detecta una coincidencia.
- Los valores de etiqueta 01, 02 y 03 se utilizan para identificar las tres estaciones de reserva de la unidad funcional de suma, mientras que 04 y 05 se utilizan para identificar las dos estaciones de reserva de la unidad funcional de multiplicación/división. Estos valores de etiqueta son los ID de las estaciones de reserva.
- Inicialmente, el valor de los registros es `F0`=4.0, `F2`=2.0, `F4`=3.0 y `F4`=2.0

Problema 2 (4 puntos)

Considere el siguiente bucle:

```
for (i=1; i<=10; i++) {
      b[i]=a[i]*c;
      c=c+1;
      if (c>10) then goto etiqueta;
}
etiqueta:......
```

Calcule la penalización debida a los saltos, en función del valor inicial de `c` (número entero), considerando que el procesador utiliza:

a) Predicción fija (siempre se considera que se va a producir el salto)

b) Predicción estática (si es desplazamiento es negativo, es decir, hacia atrás, se considera que se produce el salto; y si el desplazamiento es positivo, se predice como no efectivo).

c) Predicción dinámica con 1 bit de historial para cada salto (1=saltar; 0=no saltar; valor inicial=1).

La penalización por saltos incorrectamente predichos es de 4 ciclos y para los saltos correctamente predichos, la penalización es 0.

Problema 3 (3 puntos)

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`, `R4c` 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.