Centro Asociado Palma de Mallorca

Exámenes Ingeniería Computadores II

Tutor: Antonio Rivero Cuesta

Exámenes

TEMA 1

Centro Asociado Palma de Mallorca

Febrero 2012 - 2ª Problema 1

Tutor: Antonio Rivero Cuesta

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 33% ocasiona una detención de un ciclo, consumiendo 112 nseg. (2 *ciclos* · 56 *nseg*)

El 8% ocasiona una detención de dos ciclos, por lo que consumen 168 nseg. (3 *ciclos* · 56 *nseg*).

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:

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

Centro Asociado Palma de Mallorca

Septiembre 2012 Problema 1

Tutor: Antonio Rivero Cuesta

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

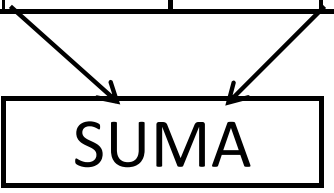
No se ejecuta RS 04

FR

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

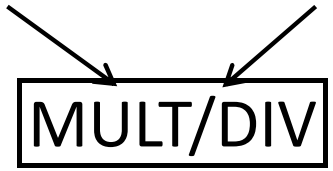
RS

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



RS

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



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

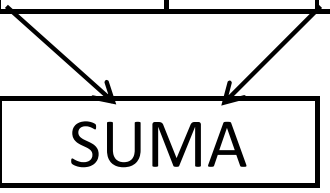
NO se ejecuta RS 02

FR

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

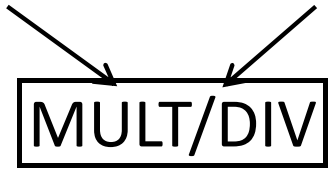
RS

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



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



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

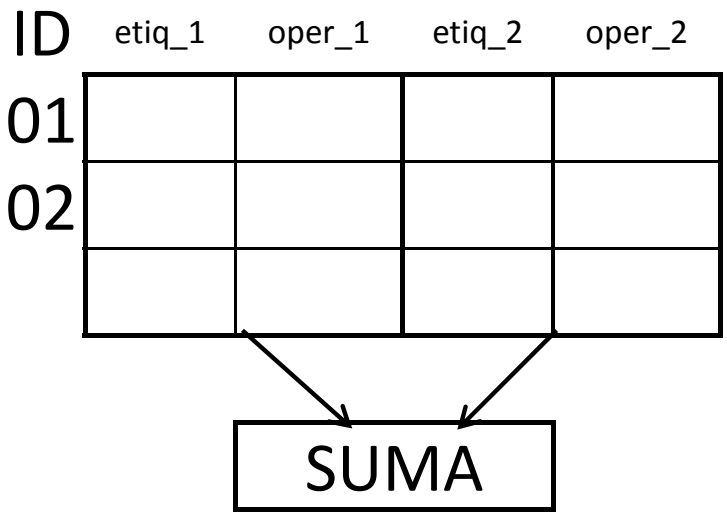
Ciclo 5: Se actualiza el valor de RS 02: 30.25

Se vacía RS 02

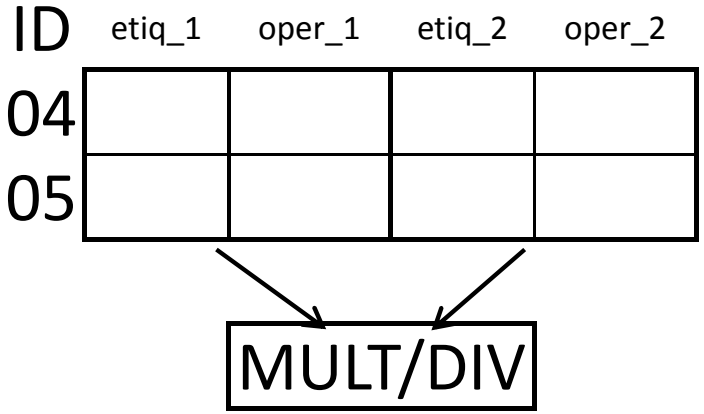
FR

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

RS



RS



Centro Asociado Palma de Mallorca

Septiembre 2012 - R Problema 1

Tutor: Antonio Rivero Cuesta

Tras añadir un nuevo procesador a un ordenador 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 un nuevo procesador.

¿Qué porcentaje del tiempo de ejecución original se ha reducido gracias a la mejora?

$$T_{original} = 0,45 + \left(0.55 \cdot 9\right) = 5,4$$

Por lo tanto, la ganancia obtenida aplicando la mejora es del 5,4.

Sustituyendo en la expresión de la ganancia:

$$S_p = \frac{p}{1 + f\,(p-1)} \rightarrow 5,4 = \frac{9}{1 + f\,(9-1)} \rightarrow f = 8,3\%$$

(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%

Centro Asociado Palma de Mallorca

Febrero 2013 - 1ª Problema 1

Tutor: Antonio Rivero Cuesta

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.

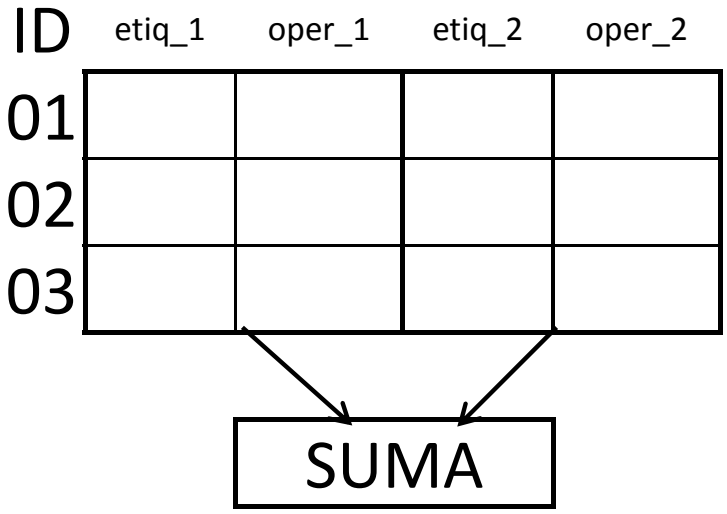
- Inicialmente, el valor de los registros es F0=2.0, F2=2.5, F4=8.0 y F6=4.0.

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

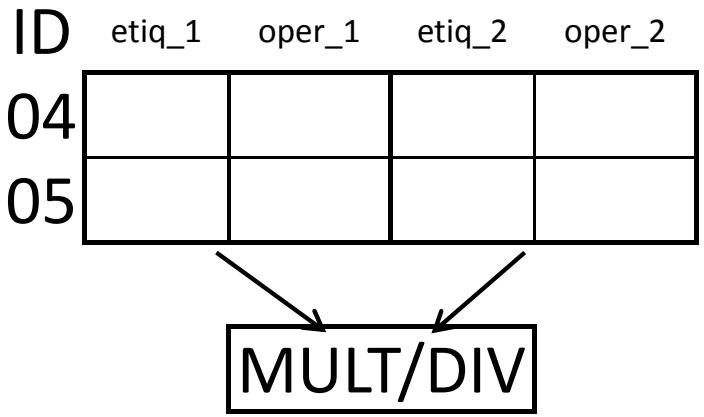
FR

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

RS



RS



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

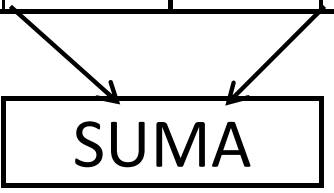
Se envía RS 01: 6.5 al CDB

FR

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

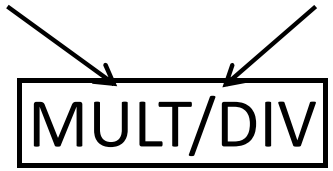
RS

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



RS

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



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

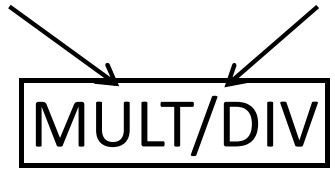
	bitOc.	etiqueta	dato
F0	Si	04	2.0
F2	Si	03	2.5
F4			6.5
F6	Si	02	4.0

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

RS

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



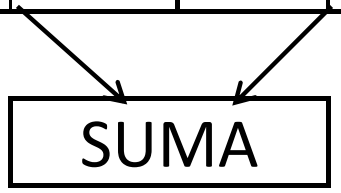
Ciclo 3: Se actualiza el valor de RS 04: 10.0
Se vacía RS 04
Se ejecuta RS 02 1/1
Se envía RS 02: 16.5 al CDB
NO se ejecuta RS 03

FR

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

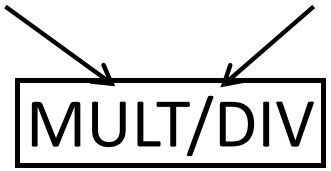
RS

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



RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				



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

Centro Asociado Palma de Mallorca

Febrero 2013 – 2ª Problema 2

Tutor: Antonio Rivero Cuesta

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?

a) 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 las 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).

Centro Asociado Palma de Mallorca

Septiembre 2013 - R Problema 1

Tutor: Antonio Rivero Cuesta

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 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 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.

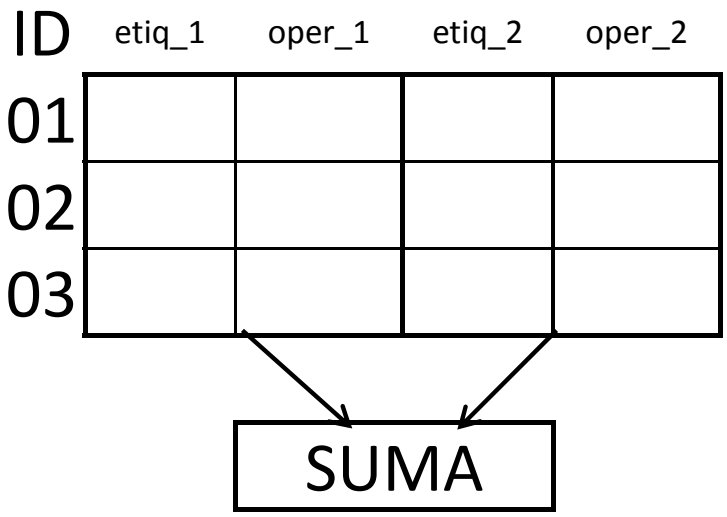
- Inicialmente, el valor de los registros es F0=4.0, F2=2.0, F4=3.0 y F6=2.0.

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

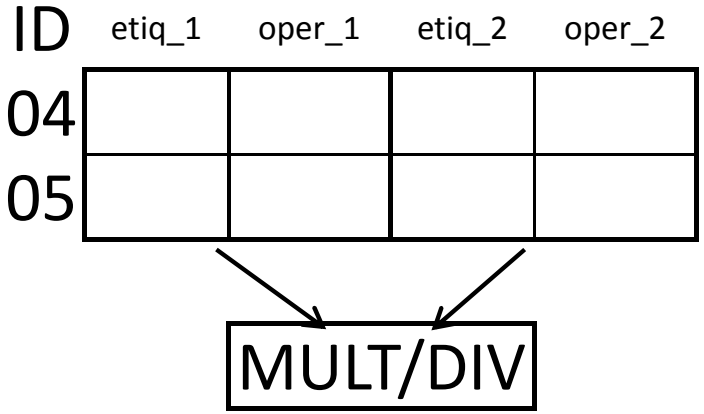
FR

	bitOc.	etiqueta	dato
F0			4.0
F2			2.0
F4			3.0
F6			2.0

RS



RS



Ciclo 1: Se distribuye i1 e i2

i1: MULTD F2,F2,F6

i2: MULTD F4,F2,F6

Se ejecuta RS 04 1/3

No se ejecuta RS 05

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	04	2.0
F4	Si	05	3.0
F6			2.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
02				
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i1: 04	00	2.0	00	2.0
i2: 05	04	xx	00	2.0

MULT/DIV

Ciclo 2: Se distribuye i3 e i4

i3: ADDD F2,F4,F6

i4: ADDD F6,F2,F6

Se ejecuta RS 04 2/3

No se ejecuta RS 05

No se ejecuta RS 01

No se ejecuta RS 02

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	01	2.0
F4	Si	05	3.0
F6	Si	02	2.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i3: 01	05	xx	00	2.0
i4: 02	04	xx	00	2.0
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i1: 04	00	2.0	00	2.0
i2: 05	04	xx	00	2.0

MULT/DIV

Ciclo 3: Se ejecuta RS 04 3/3

Se envía RS 04: 4.0 al CDB

No se ejecuta RS 05

No se ejecuta RS 01

No se ejecuta RS 02

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	01	2.0
F4	Si	05	3.0
F6	Si	02	2.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i3: 01	05	xx	00	2.0
i4: 02	04	xx	00	2.0
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i1: 04	00	2.0	00	2.0
i2: 05	04	xx	00	2.0

MULT/DIV

Ciclo 4: Se actualiza el valor de RS 04: 4.0

Se vacía RS 04

Se ejecuta RS 05 1/3

Se ejecuta RS 02 1/2

No se ejecuta RS 01

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	01	2.0
F4	Si	05	3.0
F6	Si	02	2.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i3: 01	05	xx	00	2.0
i4: 02	04	4.0	00	2.0

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
i2: 05	04	4.0	00	2.0

MULT/DIV

Ciclo 5: Se ejecuta RS 05 2/3
Se ejecuta RS 02 2/2
Se envía RS 02: 6.0 al CDB
No se ejecuta RS 01

FR

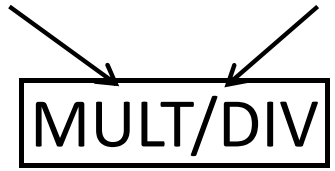
	bitOc.	etiqueta	dato
F0			4.0
F2	Si	01	2.0
F4	Si	05	3.0
F6	Si	02	2.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i3: 01	05	xx	00	2.0
i4: 02	04	4.0	00	2.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
i2: 05	04	4.0	00	2.0



Ciclo 6: Se ejecuta RS 05 3/3

Se envía RS 05: 8.0 al CDB

Se actualiza el valor de RS 02: 6.0

Se vacía RS 02

Se ejecuta RS 01 1/2

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	01	2.0
F4	Si	05	3.0
F6			6.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i3: 01	05	6.0	00	2.0
02				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
i2: 05	04	4.0	00	2.0

MULT/DIV

Ciclo 7: Se actualiza el valor de RS 05: 8.0
Se vacía RS 05
Se ejecuta RS 01 2/2
Se envía RS 01: 8.0 al CDB

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	01	2.0
F4			8.0
F6			6.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i3: 01	05	6.0	00	2.0
02				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

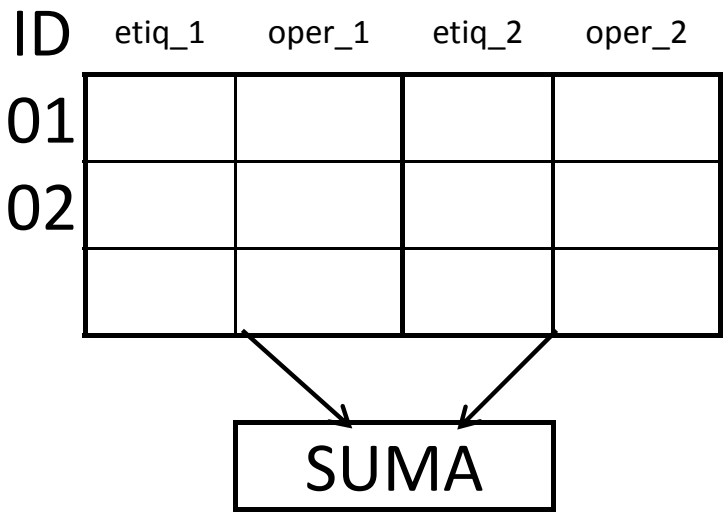
Ciclo 8: Se actualiza el valor de RS 01: 8.0

Se vacía RS 01

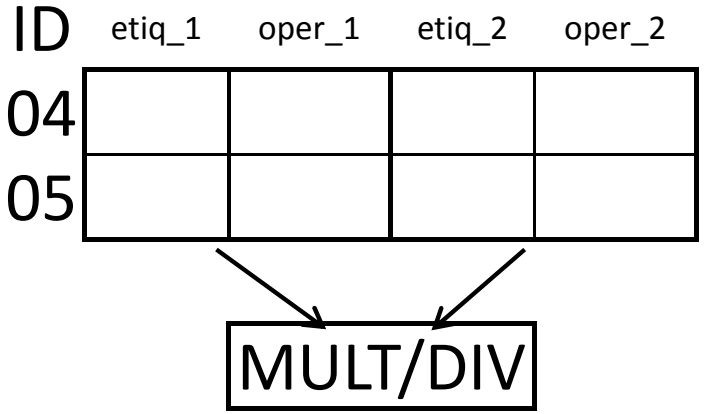
FR

	bitOc.	etiqueta	dato
F0			4.0
F2			8.0
F4			8.0
F6			6.0

RS



RS



Centro Asociado Palma de Mallorca

Febrero 2014 - 1ª Problema 1

Tutor: Antonio Rivero Cuesta

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

- 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.

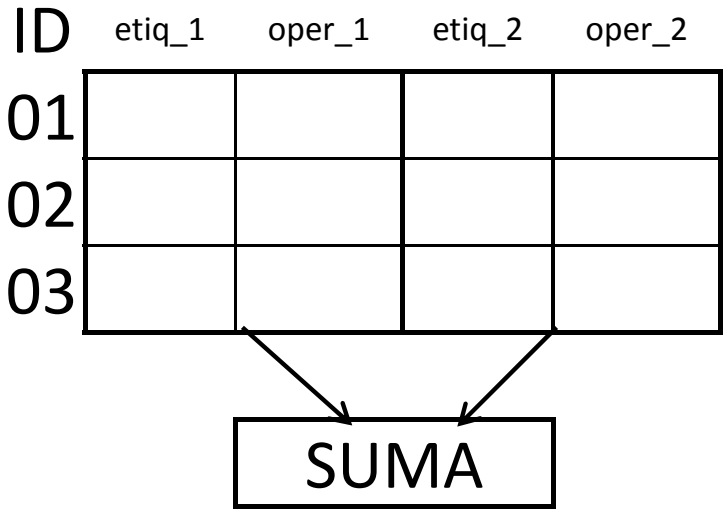
- Inicialmente, el valor de los registros es F0=3.0, F2=2.0, F4=6.0 y F6=5.0.

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

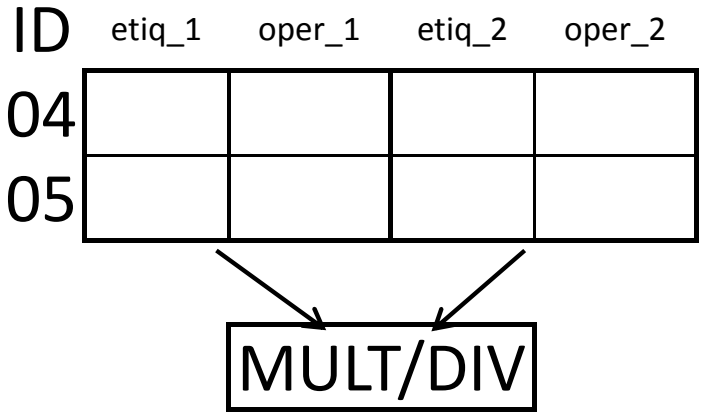
FR

	bitOc.	etiqueta	dato
F0			3.0
F2			2.0
F4			6.0
F6			5.0

RS



RS



Ciclo 1: Se distribuye i1 e i2
i1: ADDD F4,F2,F0
i2: MULTD F6,F2,F0
Se ejecuta RS 01 1/1
Se envía RS 01: 5.0 al CDB
Se ejecuta RS 04 1/2

FR

	bitOc.	etiqueta	dato
F0			3.0
F2			2.0
F4	Si	01	6.0
F6	Si	04	5.0

RS

	ID	etiq_1	oper_1	etiq_2	oper_2
i1:	01	00	2.0	00	3.0
	02				
	03				

SUMA

RS

	ID	etiq_1	oper_1	etiq_2	oper_2
i2:	04	00	2.0	00	3.0
	05				

MULT/DIV

Ciclo 2: Se distribuye i3 e i4

i3: MULTD F2,F4,F6

i4: ADDD F0,F2,F4

Se ejecuta RS 05 1/2

No se ejecuta RS 02

Se actualiza el valor de RS 01: 5.0

Se vacía RS 01

Se ejecuta RS 04 2/2

Se envía RS 04: 6.0 al CDB

FR

	bitOc.	etiqueta	dato
F0	Si	02	3.0
F2	Si	05	2.0
F4			5.0
F6	Si	04	5.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
i4: 02	05	xx	00	5.0
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i2: 04	00	2.0	00	3.0
i3: 05	00	5.0	00	5.0

MULT/DIV

Ciclo 3: Se actualiza el valor de RS 04: 6.0

Se vacía RS 04

Se ejecuta RS 05 2/2

Se envía RS 05: 25.0 al CDB

No se ejecuta RS 02

FR

	bitOc.	etiqueta	dato
F0	Si	02	3.0
F2	Si	05	2.0
F4			5.0
F6			6.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
i4: 02	05	xx	00	5.0
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
i3: 05	00	5.0	00	5.0

MULT/DIV

Ciclo 4: Se actualiza el valor de RS 05: 25.0
Se vacía RS 05
Se ejecuta RS 02 1/1
Se envía RS 02: 30.0 al CDB

FR

	bitOc.	etiqueta	dato
F0	Si	02	3.0
F2			25.0
F4			5.0
F6			6.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
01				
i4: 02	05	25.0	00	5.0

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

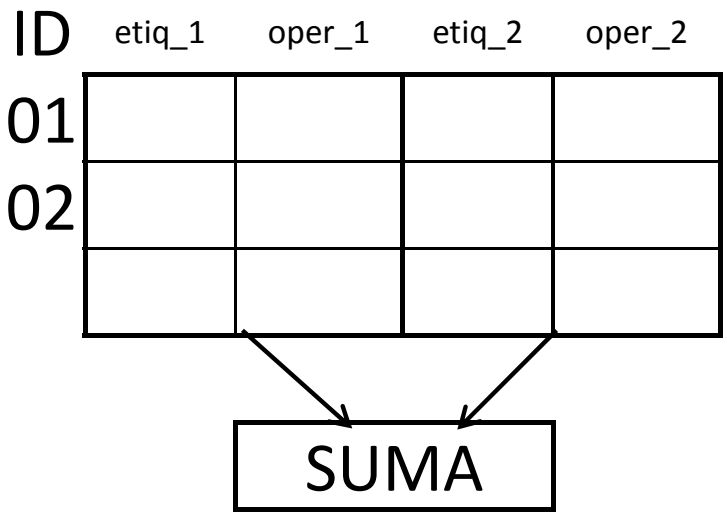
Ciclo 5: Se actualiza el valor de RS 02: 30.0

Se vacía RS 02

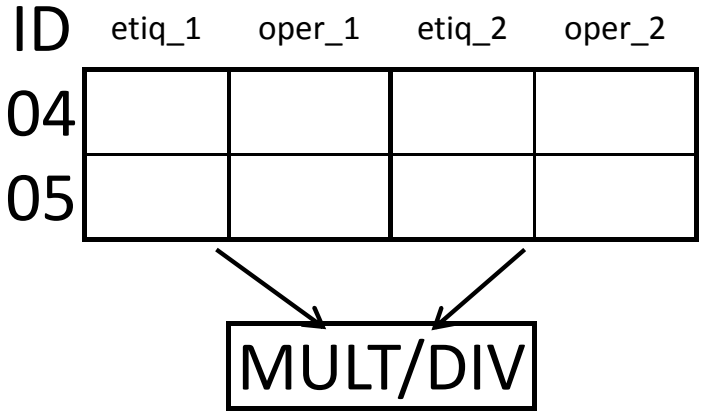
FR

	bitOc.	etiqueta	dato
F0			30.0
F2			25.0
F4			5.0
F6			6.0

RS



RS



Centro Asociado Palma de Mallorca

Febrero 2014 - 2ª Problema 1

Tutor: Antonio Rivero Cuesta

Un procesador sin segmentación necesita 210 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 9 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 210 nseg.

La segmentación de 7 etapas de este apartado se caracteriza por acortar el tiempo medio de ejecución de una instrucción a 39 nseg.:

$$\frac{210\ nseg}{7\ etapas} + 9\ nseg = 39\ nseg$$

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

$$\frac{210\ nseg}{39\ nseg} = 5{,}38\ veces\ más\ rápido$$

b) Un procesador B con una segmentación de 7 etapas, consumiendo cada una de ellas 25 nseg., 30 nseg., 40 nseg., 40 nseg., 50 nseg., 50 nseg. y 60 nseg. respectivamente, y siendo la sobrecarga por cada etapa de 9 nseg. Un 37% de todas las instrucciones de la segmentación son detenidas durante un ciclo de reloj y un 9% durante dos ciclos.

La etapa más lenta es la que dicta la velocidad de las restantes etapas, por lo que cada etapa consumirá 69 nseg. (60 nseg. más los 9 nseg. de retardo).

El 37% ocasiona una detención de un ciclo, consumiendo 138 nseg. (2 *ciclos* · 69 *nseg*)

El 9% ocasiona una detención de dos ciclos, por lo que consumen 207 nseg. (3 *ciclos* · 69 *nseg*).

El 54%, no provocan detenciones, empleando sólo un ciclo de reloj (69 nseg.).

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

$0{,}37 \cdot 69 \cdot 2 = 51{,}06$ *nseg.*
$0{,}09 \cdot 69 \cdot 3 = 18{,}63$ *nseg.*
$0{,}54 \cdot 69 \cdot 1 = 37{,}26$ *nseg.*

Total: $= 106{,}95$ *nseg.*

Por lo tanto, la aceleración obtenida por la máquina B con respecto a la máquina sin segmentar es de 1,40:

$$\frac{210\ nseg}{106,95\ nseg} = 1,96\ veces\ más\ rápido$$

Centro Asociado Palma de Mallorca

Septiembre 2014 Problema 1

Tutor: Antonio Rivero Cuesta

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   F2,F4,F0
i2: ADDD   F0,F6,F4
i3: MULTD  F2,F4,F6
i4: ADDD   F4,F2,F0
i5: ADDD   F2,F6,F2
```

- 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.

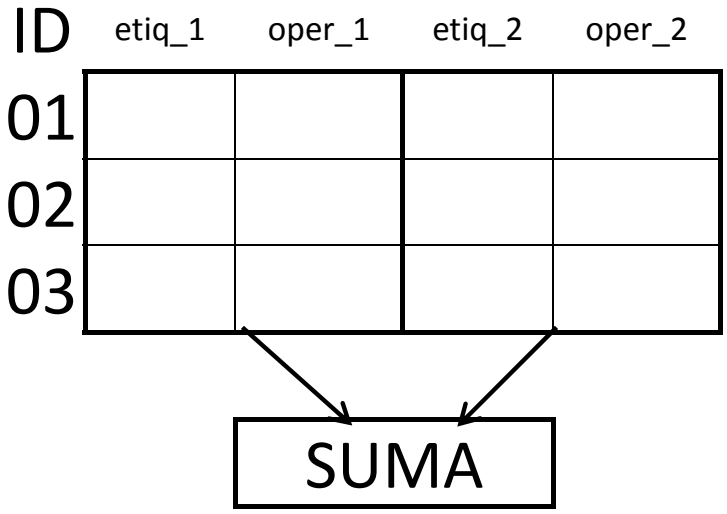
- Inicialmente, el valor de los registros es F0=4.0, F2=2.0, F4=1.0 y F6=3.0.

```
i1:  ADDD   F2,F4,F0
i2:  ADDD   F0,F6,F4
i3:  MULTD  F2,F4,F6
i4:  ADDD   F4,F2,F0
i5:  ADDD   F2,F6,F2
```

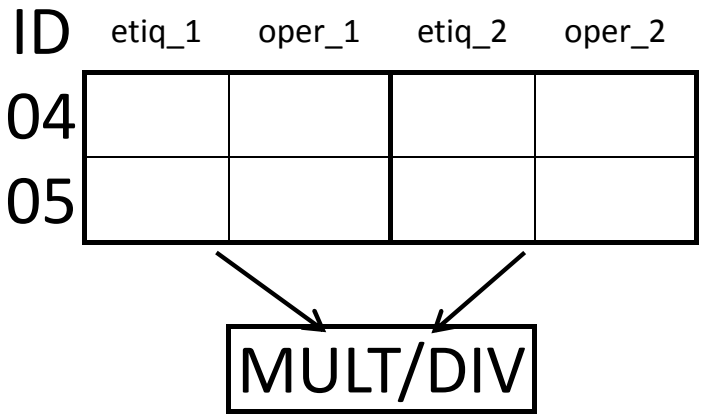
FR

	bitOc.	etiqueta	dato
F0			4.0
F2			2.0
F4			1.0
F6			3.0

RS



RS



Ciclo 1: Se distribuye i1 e i2

i1: ADDD F2,F4,F0

i2: ADDD F0,F6,F4

Se ejecuta RS 01 1/1

Se envía RS 01: 5.0 al CDB

Se ejecuta RS 02 1/1

Se envía RS 02: 4.0 al CDB

FR

	bitOc.	etiqueta	dato
F0	Si	02	4.0
F2	Si	01	2.0
F4			1.0
F6			3.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i1: 01	00	1.0	00	4.0
i2: 02	00	3.0	00	1.0
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

Ciclo 2: Se actualiza el valor de RS 01: 5.0
Se vacía RS 01
Se actualiza el valor de RS 02: 4.0
Se vacía RS 02
Se distribuye i3 e i4
i3: MULTD F2,F4,F6
i4: ADDD F4,F2,F0
Se ejecuta RS 04 1/2
NO se ejecuta RS 01

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	04	5.0
F4	Si	01	1.0
F6			3.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i4: 01	04	xx	00	4.0
02				
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i3: 04	00	1.0	00	3.0
05				

MULT/DIV

Ciclo 3: Se distribuye i5

i5: ADDD F2,F6,F2

Se ejecuta RS 04 2/2

Se envía RS 04: 3.0 al CDB

NO se ejecuta RS 01

NO se ejecuta RS 02

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	04	5.0
F4	Si	01	1.0
F6			3.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i4: 01	04	xx	00	4.0
i5: 02	00	3.0	04	xx
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i3: 04	00	1.0	00	3.0
05				

MULT/DIV

Ciclo 4: Se actualiza el valor de RS 04: 3.0
Se vacía RS 04
Se ejecuta RS 01 1/1
Se envía RS 01: 6.0 al CDB
Se ejecuta RS 02 1/1
Se envía RS 02: 6.0 al CDB

FR

	bitOc.	etiqueta	dato
F0			4.0
F2	Si	02	5.0
F4	Si	01	1.0
F6			3.0

RS

ID	etiq_1	oper_1	etiq_2	oper_2
i4: 01	00	3.0	00	3.0
i5: 02	00	3.0	00	3.0
03				

SUMA

RS

ID	etiq_1	oper_1	etiq_2	oper_2
04				
05				

MULT/DIV

Ciclo 5: Se actualiza el valor de RS 01: 6.0
Se vacía RS 01
Se actualiza el valor de RS 02: 6.0
Se vacía RS 02

FR

	bitOc.	etiqueta	dato
F0			4.0
F2			6.0
F4			6.0
F6			3.0

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

Centro Asociado Palma de Mallorca

Septiembre 2014 - R Problema 2

Tutor: Antonio Rivero Cuesta

Un procesador sin segmentación necesita 1125 nseg. para procesar cinco instrucciones. Con respecto a este procesador, calcular la aceleración que se obtiene en los dos casos siguientes:

a) Un procesador P1 dotado de una segmentación de 9 etapas, consumiendo cada etapa el mismo tiempo. Cada etapa ocasiona una sobrecarga de 7 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 1125/5 = 225 nseg.

La segmentación de 9 etapas de este apartado se caracteriza por acortar el tiempo medio de ejecución de una instrucción a 32 nseg.:

$$\frac{225\ nseg}{9\ etapas} + 7\ nseg = 32\ nseg$$

Por lo tanto, la aceleración obtenida por la máquina A con respecto a la máquina sin segmentar es 7,03:

$$\frac{225\ nseg}{32\ nseg} = 7,03\ veces\ más\ rápido$$

b) Un procesador P2 con una segmentación de 9 etapas, consumiendo cada una de ellas 30 nseg., 25 nseg., 45 nseg., 45 nseg., 50 nseg. , 50 nseg, 40 nseg, 40 nseg y 40 nseg respectivamente, y siendo la sobrecarga por cada etapa de 7 nseg. Un 42% de todas las instrucciones de la segmentación son detenidas durante un ciclo de reloj, un 9% durante dos ciclos y un 12% durante tres ciclos.

La etapa más lenta es la que dicta la velocidad de las restantes etapas, por lo que cada etapa consumirá 57 nseg. (50 nseg. más los 7 nseg. de retardo).

El 42% ocasiona una detención de un ciclo, consumiendo 114 nseg. (2 *ciclos* · 57 *nseg*)

El 9% ocasiona una detención de dos ciclos, por lo que consumen 171 nseg. (3 *ciclos* · 57 *nseg*).

El 12% ocasiona una detención de tres ciclos, por lo que consumen 228 nseg. (4 *ciclos* · 57 *nseg*).

El 37%, no provocan detenciones, empleando sólo un ciclo de reloj (57 nseg.).

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

$0{,}42 \cdot 57 \cdot 2 = 47{,}88$ *nseg.*
$0{,}09 \cdot 57 \cdot 3 = 15{,}39$ *nseg.*
$0{,}12 \cdot 57 \cdot 4 = 27{,}36$ *nseg.*
$0{,}37 \cdot 57 \cdot 1 = 21{,}09$ *nseg.*

Total: $= 111{,}72$ *nseg.*

Por lo tanto, la aceleración obtenida por la máquina B con respecto a la máquina sin segmentar es de 2,01:

$$\frac{225\ nseg}{111,72\ nseg} = 2,01\ veces\ más\ rápido$$

Exámenes

TEMA 2

Centro Asociado Palma de Mallorca

Septiembre 2012 Problema 2

Tutor: Antonio Rivero Cuesta

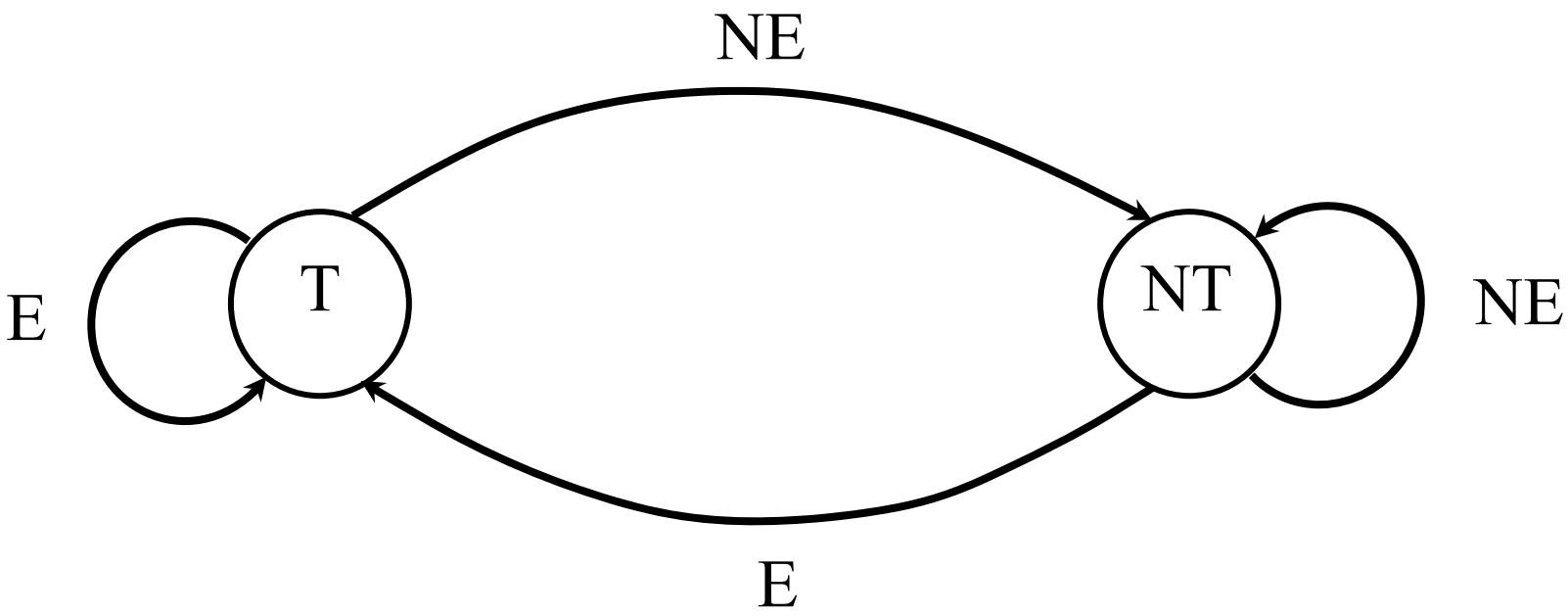
Suponga que un salto tiene la siguiente secuencia de resultados efectivos (E) y no efectivos (N):

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?



Predicción Smith:
T(Taken): 1
NT(Not Taken): 0

Resultado del salto
E: Efectivo
NE: No Efectivo

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
		N		
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
		N		
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
		E		
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
		E		
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
		E		
		N		
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
		N		
		N		

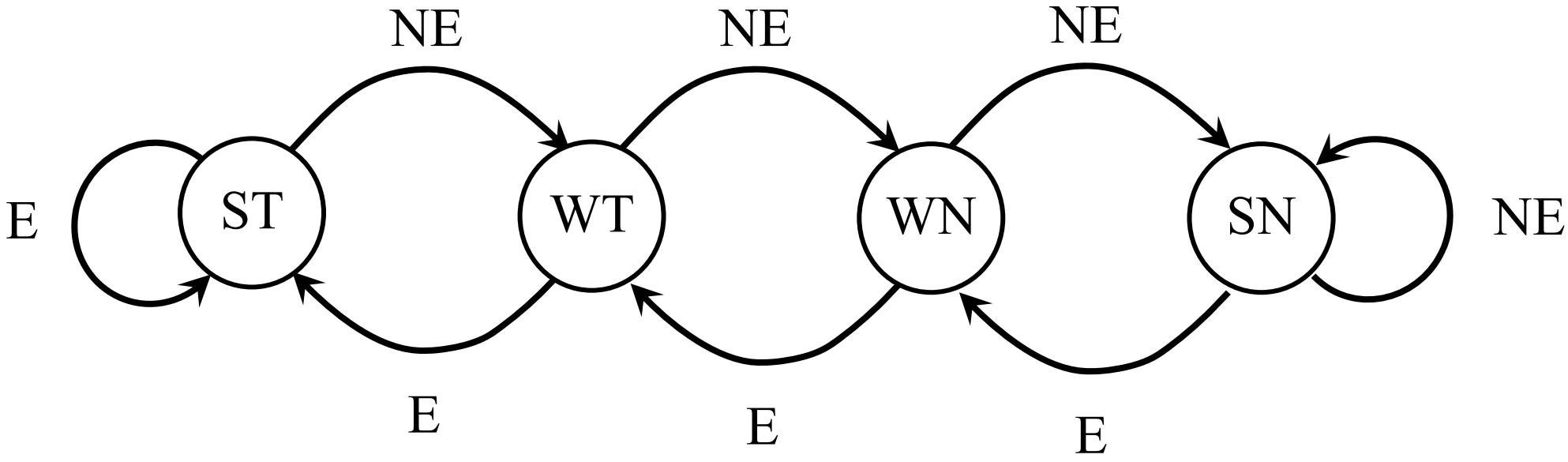
Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
		N		

Contador	Predicción	Resultado Real	Predicción correcta	Siguiente estado
T	E	E	SI	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	NT
NT	N	E	NO	T
T	E	E	SI	T
T	E	E	SI	T
T	E	N	NO	NT
NT	N	N	SI	T

De las 15 predicciones hay 10 predicciones correctas.

La precisión del predictor de 1 bit ha sido del 66,67%.

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).



Predicción $Smith_2$:
ST (Strongly Taken): 11
WT (Weakly Taken): 10
WN (Weakly Not Taken): 01
SN (Strongly Not Taken): 00

Resultado del salto:
E: Efectivo
NE: No Efectivo

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
		N		
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
		N		
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
		E		
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
		E		
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
		E		
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
		N		
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
		N		
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
		E		
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
		E		
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
		E		
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
		N		
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
		N		

Contador 2 bits	Predicción	Resultado Real	Predicción correcta	Siguiente estado
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01
WN - 01	N	E	NO	WT - 10
WT - 10	E	E	SI	ST - 11
ST - 11	E	E	SI	ST - 11
ST - 11	E	N	NO	WT - 10
WT - 10	E	N	NO	WN - 01

De las 15 predicciones ha habido 7 predicciones correctas.

La precisión del predictor ha sido del 46,67%.

Centro Asociado Palma de Mallorca

Septiembre 2012 - R Problema 2

Tutor: Antonio Rivero Cuesta

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
```

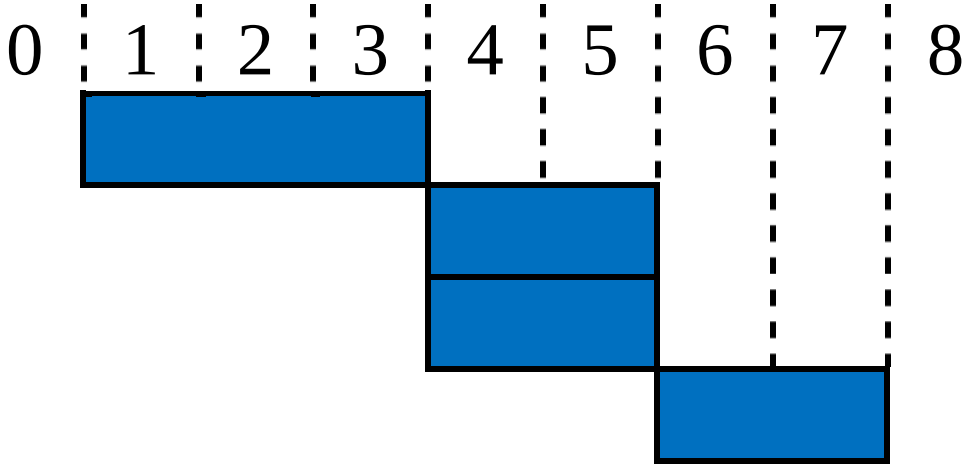
Se distribuyen, a razón de una por ciclo, a las dos estaciones de reserva y se emiten en cuanto sus operandos están disponibles.

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.

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

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



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.

Apartado 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.

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

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

	Datos	Válido	Ocup
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	Ocup	Índice
F1	2		
F2	3	1	2
F3		1	1
F4			
F5			

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

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

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

	Datos	Válido	Ocup
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 las estaciones de reserva para emisión de i2 e i3.

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

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

Ciclo 4: Emisión de i1 e i2.
Liberación de Fr1.

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

	Datos	Válido	Ocup
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 Fr2 en entrada de la instrucción i4 para poder emitirla.

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

	Datos	Válido	Ocup
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 de Fr3 en F3.
Liberación de Fr3.

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

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

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

Ciclo 8: Escritura de resultado de Fr4 en F5
Liberación de Fr4.

	Datos	Ocup	Índice
F1	2	0	
F2	8	0	2
F3	4	0	3
F4		0	
F5	10	0	4

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

Centro Asociado Palma de Mallorca

Febrero 2013 - 1ª Problema 3

Tutor: Antonio Rivero Cuesta

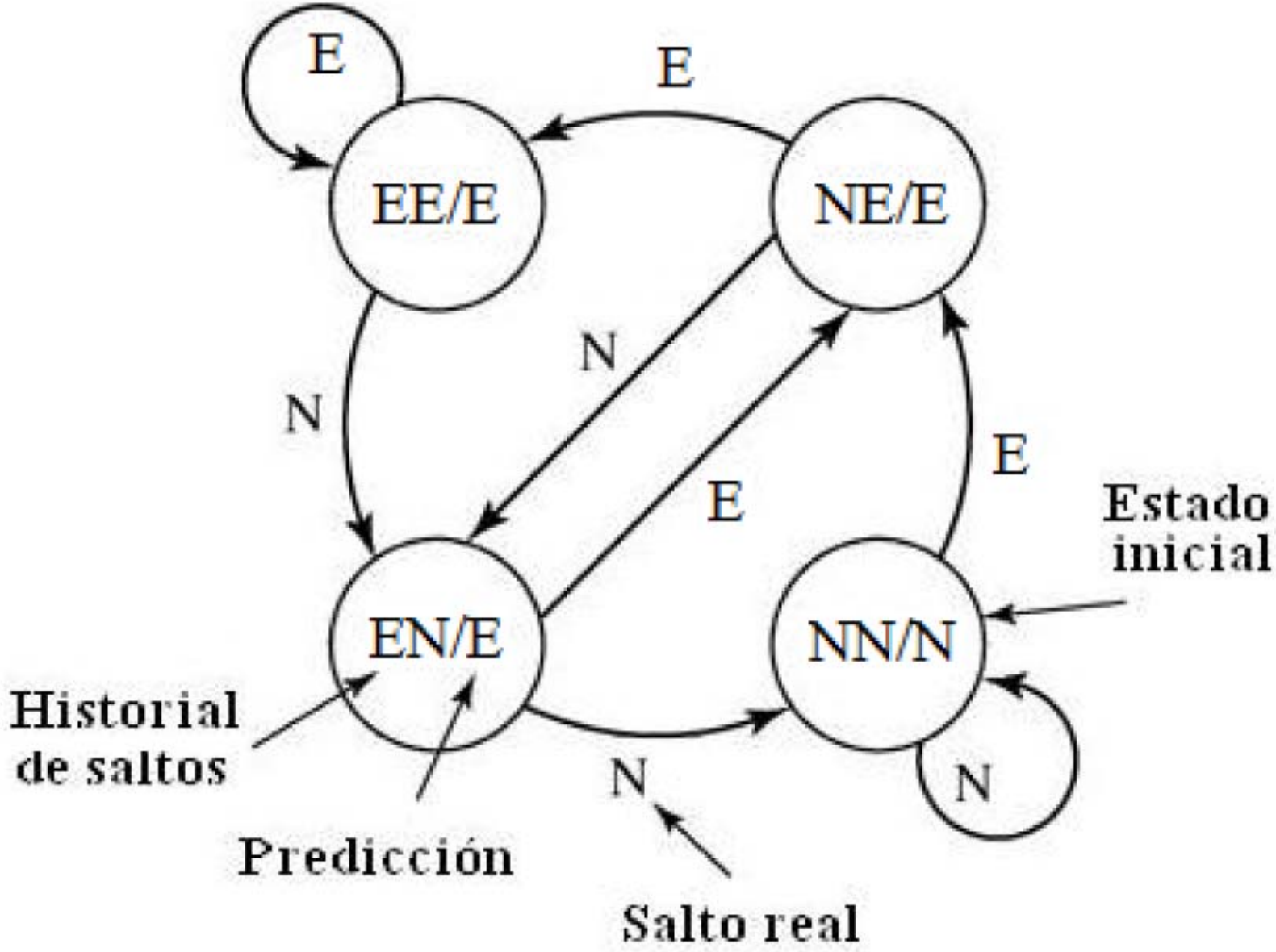
Considere el siguiente segmento de código que se ejecuta en el cuerpo principal de un bucle:

```
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.


E
E
EE/E
NE/E
N
N
E
E
Estado
inicial
EN/E
NN/N
Historial
de saltos
N
N
Predicción
Salto real

¿Cuál es la secuencia de predicciones para los salto S1 y S2 en cada iteración 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.

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								
S1_Predicho	N								
S1_Real									
Estado	NN								
S2_Predicho	N								
S2_Real									

	8	9	10	11	12	20	29	30	31
Estado	NN								
S1_Predicho	N								
S1_Real	E								
Estado	NN								
S2_Predicho	N								
S2_Real	N								

	8	9	10	11	12	20	29	30	31
Estado	NN	NE							
S1_Predicho	N	E							
S1_Real	E								
Estado	NN	NN							
S2_Predicho	N	N							
S2_Real	N								

	8	9	10	11	12	20	29	30	31
Estado	NN	NE							
S1_Predicho	N	E							
S1_Real	E	N							
Estado	NN	NN							
S2_Predicho	N	N							
S2_Real	N	N							

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN						
S1_Predicho	N	E	E						
S1_Real	E	N							
Estado	NN	NN	NN						
S2_Predicho	N	N	N						
S2_Real	N	N							

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN						
S1_Predicho	N	E	E						
S1_Real	E	N	E						
Estado	NN	NN	NN						
S2_Predicho	N	N	N						
S2_Real	N	N	E						

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE					
S1_Predicho	N	E	E	E					
S1_Real	E	N	E						
Estado	NN	NN	NN	NE					
S2_Predicho	N	N	N	E					
S2_Real	N	N	E						

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE					
S1_Predicho	N	E	E	E					
S1_Real	E	N	E	N					
Estado	NN	NN	NN	NE					
S2_Predicho	N	N	N	E					
S2_Real	N	N	E	N					

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN				
S1_Predicho	N	E	E	E	E				
S1_Real	E	N	E	N					
Estado	NN	NN	NN	NE	EN				
S2_Predicho	N	N	N	E	E				
S2_Real	N	N	E	N					

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN				
S1_Predicho	N	E	E	E	E				
S1_Real	E	N	E	N	E				
Estado	NN	NN	NN	NE	EN				
S2_Predicho	N	N	N	E	E				
S2_Real	N	N	E	N	N				

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN	NE			
S1_Predicho	N	E	E	E	E	E			
S1_Real	E	N	E	N	E				
Estado	NN	NN	NN	NE	EN	NN			
S2_Predicho	N	N	N	E	E	N			
S2_Real	N	N	E	N	N				

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN	NE			
S1_Predicho	N	E	E	E	E	E			
S1_Real	E	N	E	N	E	E			
Estado	NN	NN	NN	NE	EN	NN			
S2_Predicho	N	N	N	E	E	N			
S2_Real	N	N	E	N	N	E			

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN	NE	EE		
S1_Predicho	N	E	E	E	E	E	E		
S1_Real	E	N	E	N	E	E			
Estado	NN	NN	NN	NE	EN	NN	NE		
S2_Predicho	N	N	N	E	E	N	E		
S2_Real	N	N	E	N	N	E			

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN	NE	EE		
S1_Predicho	N	E	E	E	E	E	E		
S1_Real	E	N	E	N	E	E	N		
Estado	NN	NN	NN	NE	EN	NN	NE		
S2_Predicho	N	N	N	E	E	N	E		
S2_Real	N	N	E	N	N	E	N		

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN	NE	EE	EN	
S1_Predicho	N	E	E	E	E	E	E	E	
S1_Real	E	N	E	N	E	E	N		
Estado	NN	NN	NN	NE	EN	NN	NE	EN	
S2_Predicho	N	N	N	E	E	N	E	E	
S2_Real	N	N	E	N	N	E	N		

	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN	NE	EE	EN	
S1_Predicho	N	E	E	E	E	E	E	E	
S1_Real	E	N	E	N	E	E	N	E	
Estado	NN	NN	NN	NE	EN	NN	NE	EN	
S2_Predicho	N	N	N	E	E	N	E	E	
S2_Real	N	N	E	N	N	E	N	E	

	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	
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	

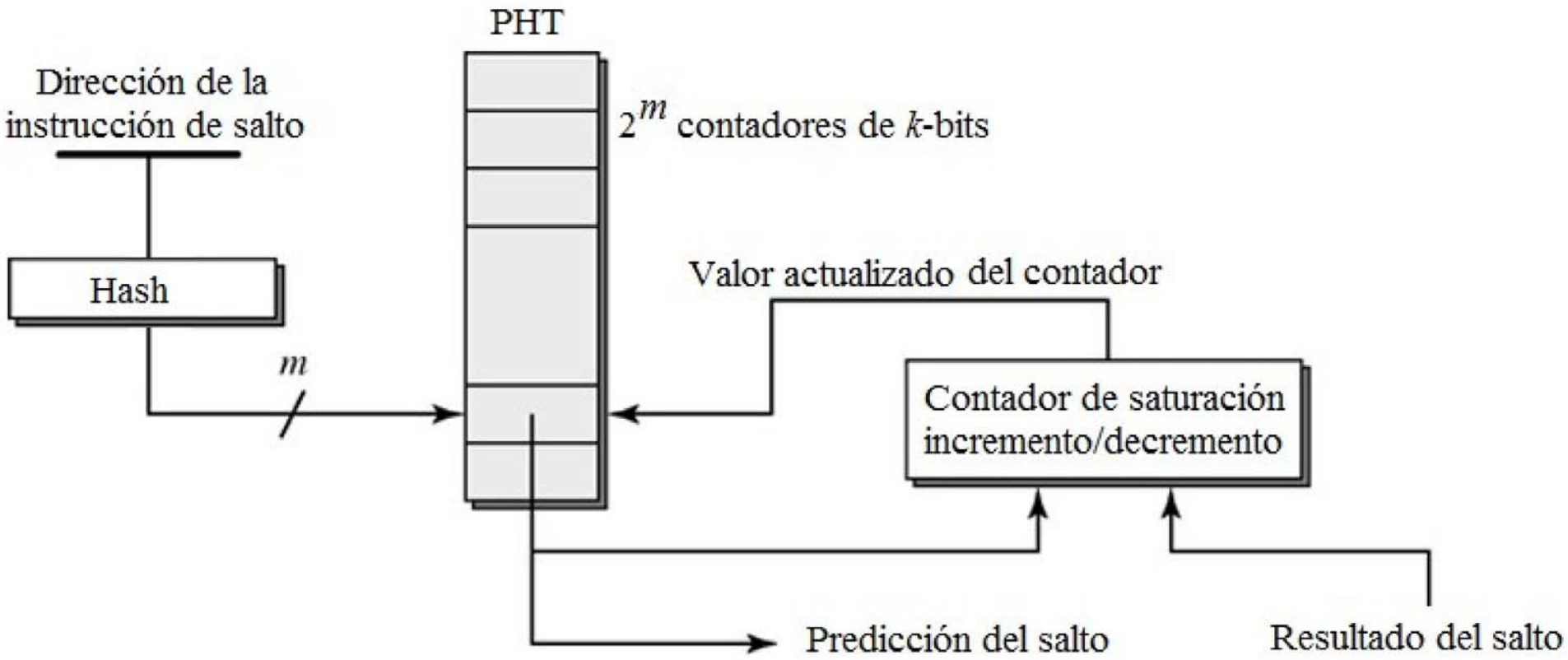
	8	9	10	11	12	20	29	30	31
Estado	NN	NE	EN	NE	EN	NE	EE	EN	NE
S1_Predicho	**N**	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**E**
S1_Real	E	N	E	N	E	E	N	E	N
Estado	NN	NN	NN	NE	EN	NN	NE	EN	NE
S2_Predicho	**N**	**N**	**N**	**E**	**E**	**N**	**E**	**E**	**E**
S2_Real	N	N	E	N	N	E	N	E	N

Centro Asociado Palma de Mallorca

Septiembre 2013 Problema 1

Tutor: Antonio Rivero Cuesta

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:

$$(\mathrm{PC}\ mod\ 2^m)$$

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*).

Solución

La función *hash* para acceder a la tabla es:

$$(\text{PC mod } 2^3)$$

El resultado de aplicar esa función a los valores de los contadores de programa son:

```
Salto S1: 0x0000 mod 8 = 0 mod 8 = 0
Salto S2: 0x1001 mod 8 = 9 mod 8 = 1
Salto S3: 0x0110 mod 8 = 6 mod 8 = 6
```

En este ejercicio, los únicos contadores que modificarán su contenido son los tres primeros de un total de 8 que cuenta la tabla.

A continuación se muestra la evolución de estos tres contadores junto con las predicciones y los resultados de los saltos.

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00								
Predicción S1	N								
Resultado S1									
Contador 1	00								
Predicción S2	--								
Resultado S2									
Contador 6	00								
Predicción S3	N								
Resultado S3									

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00								
Predicción S1	N								
Resultado S1	N								
Contador 1	00								
Predicción S2	--								
Resultado S2	--								
Contador 6	00								
Predicción S3	N								
Resultado S3	T								

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00							
Predicción S1	N	N							
Resultado S1	N								
Contador 1	00	00							
Predicción S2	--	N							
Resultado S2	--								
Contador 6	00	01							
Predicción S3	N	N							
Resultado S3	T								

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00							
Predicción S1	N	N							
Resultado S1	N	T							
Contador 1	00	00							
Predicción S2	--	N							
Resultado S2	--	N							
Contador 6	00	01							
Predicción S3	N	N							
Resultado S3	T	N							

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01						
Predicción S1	N	N	N						
Resultado S1	N	T							
Contador 1	00	00	00						
Predicción S2	--	N	--						
Resultado S2	--	N							
Contador 6	00	01	00						
Predicción S3	N	N	N						
Resultado S3	T	N							

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01						
Predicción S1	N	N	N						
Resultado S1	N	T	N						
Contador 1	00	00	00						
Predicción S2	--	N	--						
Resultado S2	--	N	--						
Contador 6	00	01	00						
Predicción S3	N	N	N						
Resultado S3	T	N	T						

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00					
Predicción S1	N	N	N	N					
Resultado S1	N	T	N						
Contador 1	00	00	00	00					
Predicción S2	--	N	--	N					
Resultado S2	--	N	--						
Contador 6	00	01	00	01					
Predicción S3	N	N	N	N					
Resultado S3	T	N	T						

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00					
Predicción S1	N	N	N	N					
Resultado S1	N	T	N	T					
Contador 1	00	00	00	00					
Predicción S2	--	N	--	N					
Resultado S2	--	N	--	T					
Contador 6	00	01	00	01					
Predicción S3	N	N	N	N					
Resultado S3	T	N	T	N					

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01				
Predicción S1	N	N	N	N	N				
Resultado S1	N	T	N	T					
Contador 1	00	00	00	00	01				
Predicción S2	--	N	--	N	--				
Resultado S2	--	N	--	T					
Contador 6	00	01	00	01	00				
Predicción S3	N	N	N	N	N				
Resultado S3	T	N	T	N					

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01				
Predicción S1	N	N	N	N	N				
Resultado S1	N	T	N	T	N				
Contador 1	00	00	00	00	01				
Predicción S2	--	N	--	N	--				
Resultado S2	--	N	--	T	--				
Contador 6	00	01	00	01	00				
Predicción S3	N	N	N	N	N				
Resultado S3	T	N	T	N	T				

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01	00			
Predicción S1	N	N	N	N	N	N			
Resultado S1	N	T	N	T	N				
Contador 1	00	00	00	00	01	01			
Predicción S2	--	N	--	N	--	N			
Resultado S2	--	N	--	T	--				
Contador 6	00	01	00	01	00	01			
Predicción S3	N	N	N	N	N	N			
Resultado S3	T	N	T	N	T				

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01	00			
Predicción S1	N	N	N	N	N	N			
Resultado S1	N	T	N	T	N	T			
Contador 1	00	00	00	00	01	01			
Predicción S2	--	N	--	N	--	N			
Resultado S2	--	N	--	T	--	T			
Contador 6	00	01	00	01	00	01			
Predicción S3	N	N	N	N	N	N			
Resultado S3	T	N	T	N	T	N			

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01	00	01		
Predicción S1	N	N	N	N	N	N	N		
Resultado S1	N	T	N	T	N	T			
Contador 1	00	00	00	00	01	01	10		
Predicción S2	--	N	--	N	--	N	--		
Resultado S2	--	N	--	T	--	T			
Contador 6	00	01	00	01	00	01	00		
Predicción S3	N	N	N	N	N	N	N		
Resultado S3	T	N	T	N	T	N			

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01	00	01		
Predicción S1	N	N	N	N	N	N	N		
Resultado S1	N	T	N	T	N	T	N		
Contador 1	00	00	00	00	01	01	10		
Predicción S2	--	N	--	N	--	N	--		
Resultado S2	--	N	--	T	--	T	--		
Contador 6	00	01	00	01	00	01	00		
Predicción S3	N	N	N	N	N	N	N		
Resultado S3	T	N	T	N	T	N	T		

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01	00	01	00	
Predicción S1	N	N	N	N	N	N	N	N	
Resultado S1	N	T	N	T	N	T	N		
Contador 1	00	00	00	00	01	01	10	10	
Predicción S2	--	N	--	N	--	N	--	T	
Resultado S2	--	N	--	T	--	T	--		
Contador 6	00	01	00	01	00	01	00	01	
Predicción S3	N	N	N	N	N	N	N	N	
Resultado S3	T	N	T	N	T	N	T		

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01	00	01	00	
Predicción S1	N	N	N	N	N	N	N	N	
Resultado S1	N	T	N	T	N	T	N	T	
Contador 1	00	00	00	00	01	01	10	10	
Predicción S2	--	N	--	N	--	N	--	T	
Resultado S2	--	N	--	T	--	T	--	T	
Contador 6	00	01	00	01	00	01	00	01	
Predicción S3	N	N	N	N	N	N	N	N	
Resultado S3	T	N	T	N	T	N	T	N	

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01	00	01	00	01
Predicción S1	N	N	N	N	N	N	N	N	N
Resultado S1	N	T	N	T	N	T	N	T	
Contador 1	00	00	00	00	01	01	10	10	11
Predicción S2	--	N	--	N	--	N	--	T	T
Resultado S2	--	N	--	T	--	T	--	T	
Contador 6	00	01	00	01	00	01	00	01	00
Predicción S3	N	N	N	N	N	N	N	N	N
Resultado S3	T	N	T	N	T	N	T	N	

Valor de *x*	8	9	10	11	12	17	20	29	31
Contador 0	00	00	01	00	01	00	01	00	01
Predicción S1	N	N	N	N	N	N	N	N	N
Resultado S1	N	T	N	T	N	T	N	T	T
Contador 1	00	00	00	00	01	01	10	10	11
Predicción S2	--	N	--	N	--	N	--	T	T
Resultado S2	--	N	--	T	--	T	--	T	T
Contador 6	00	01	00	01	00	01	00	01	00
Predicción S3	N	N	N	N	N	N	N	N	N
Resultado S3	T	N	T	N	T	N	T	N	T

Centro Asociado Palma de Mallorca

Febrero 2014 - 2 Problema 2

Tutor: Antonio Rivero Cuesta

Dispone de un procesador con una implementación de un predictor de dos niveles basado en historial global con $m = 3$ y $h = 0$ y donde las entradas de la PHT son contadores de Smith de 2 bits.

La función ***hash*** que se utiliza para acceder al predictor a partir de la dirección destino del salto es la que utilizan los procesadores actuales: (PC $mod\ 2^m$) donde *mod* es el operador resto de una división.

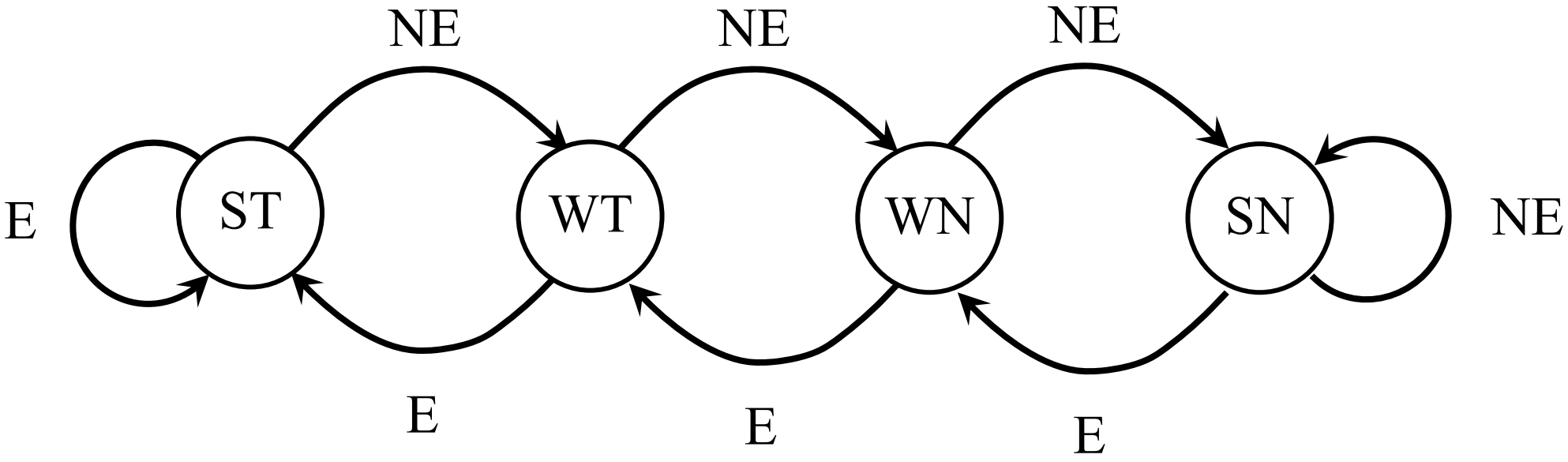
Dado el siguiente fragmento de código que se ejecuta dentro de un bucle

```
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 algoritmo.
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***).



Predicción Smith:
ST(Strongly Taken): 11
WT(Weakly Taken): 10
WN(Weakly Not Taken): 01
SN(Strongly Not Taken): 00

Resultado del salto:
E: Efectivo
NE: No Efectivo

Solución

La función *hash* para acceder a la tabla es:

$$(\text{PC mod } 2^3)$$

El resultado de aplicar esa función a los valores de los contadores de programa son:

```
Salto S1: 0x0000 mod 8 = 0 mod 8 = 0
Salto S2: 0x1001 mod 8 = 9 mod 8 = 1
Salto S3: 0x0110 mod 8 = 6 mod 8 = 6
```

	PC *mod* 2^3	Contador PHT								
x	Hash	Res	0	Pre	Res	1	Pre	Res	6	Pre
8	**6 mod 8 = 6**	NE	00	SN	--	00	SN	E	00	SN
9	**0 mod 8 = 0**	E	00	SN	NE	00	SN	NE	01	WN
10	**6 mod 8 = 6**	NE	01	WN	--	00	SN	E	00	SN
11	**0 mod 8 = 0** **9 mod 8 = 1**	E	00	SN	E	00	SN	NE	01	WN
12	**6 mod 8 = 6**	NE	01	WN	--	01	WN	E	00	WN
17	**0 mod 8 = 0** **9 mod 8 = 1**	E	00	SN	E	01	WN	NE	01	WN
20	**6 mod 8 = 6**	NE	01	WN	--	10	WT	E	00	WN
29	**0 mod 8 = 0** **9 mod 8 = 1**	E	00	SN	E	10	WT	NE	01	WN
31	**0 mod 8 = 0** **9 mod 8 = 1**	E	01	WN	E	11	ST	NE	00	WN

Exámenes

TEMA 3

Centro Asociado Palma de Mallorca

Febrero 2012 - 1ª Problema 2

Tutor: Antonio Rivero Cuesta

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.

a) Una solución válida es la siguiente.

	2 ciclos	2 ciclos	3 ciclos	3 ciclos	1 ciclo
	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

b) En base a la solución del apartado a se tendría:

	2 ciclos	2 ciclos	3 ciclos	3 ciclos	1 ciclo
	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

c) Desenrollamiento software.

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`

Instrucciones VLIW del procesador

	2 ciclos	2 ciclos	3 ciclos	3 ciclos	1 ciclo
	Carga / almacenamiento	**Carga / almacenamiento**	**Operaciones FP**	**Operaciones FP**	**Enteras/saltos**
1	LD F0,0(R1)				
2	LD F0,-8(R1)				
3	LD F0,-16(R1)		ADDD F4,F0,F2		
4	LD F0,-24(R1)		ADDD F4,F0,F2		
5	LD F0,-32(R1)		ADDD F4,F0,F2		
6	LD F0,-40(R1)	SD 0(R1),F4	ADDD F4,F0,F2	SUBI R1,R1,#8	BNEZ R1,Loop
7		SD -8(R1),F4	ADDD F4,F0,F2		
8		SD -16(R1),F4	ADDD F4,F0,F2		
9		SD -24(R1),F4			
10		SD -32(R1),F4			
11		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:

	2 ciclos	2 ciclos	3 ciclos	3 ciclos	1 ciclo
	Carga / almacenamiento	**Carga / almacenamiento**	**Operaciones FP**	**Operaciones FP**	**Enteras/saltos**
1	`LD F0,48(R1)`				
2	`LD F0,40(R1)`				
3	`LD F0,32(R1)`		`ADDD F4,F0,F2`		
4	`LD F0,24(R1)`		`ADDD F4,F0,F2`		
5	`LD F0,16(R1)`		`ADDD F4,F0,F2`		
6	`LD F0,8(R1)`	`SD 48(R1),F4`	`ADDD F4,F0,F2`	`SUBI R1,R1,#8`	`BNEZ R1,Loop`
7		`SD 48(R1),F4`	`ADDD F4,F0,F2`		
8		`SD 40(R1),F4`	`ADDD F4,F0,F2`		
9		`SD 32(R1),F4`			
10		`SD 24(R1),F4`			
11		`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 + 795 iteraciones de 1 ciclo: 805 ciclos
Instrucciones procesadas: 805 instrucciones

Centro Asociado Palma de Mallorca

Febrero 2012 - 2ª Problema 2

Tutor: Antonio Rivero Cuesta

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:

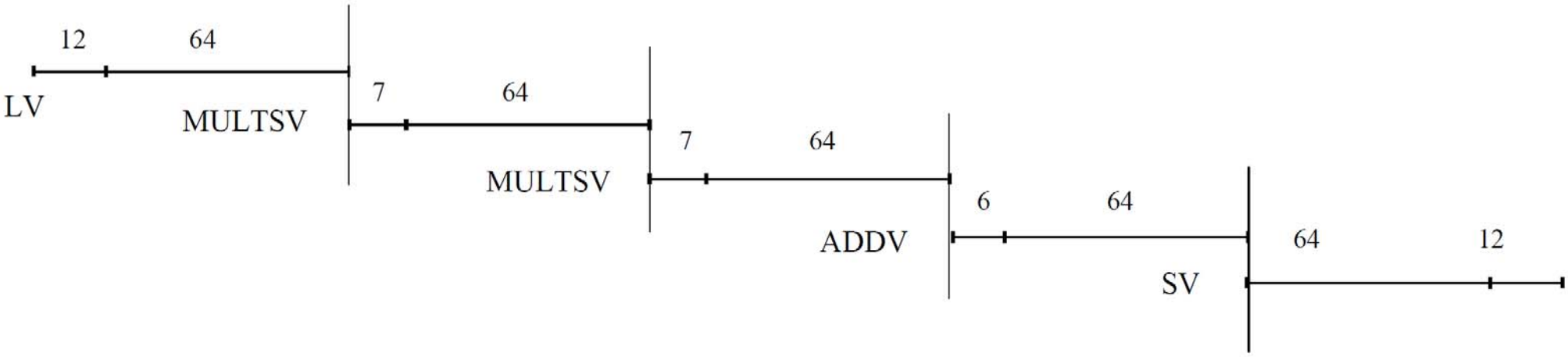
a) Sin considerar encadenamiento de resultados.
b) Permitiendo encadenamientos.
c) Considerando encadenamientos y dos unidades de multiplicación

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

$$R_\infty = \lim_{n\to\infty}\left(\frac{3n}{10+\left\lceil\frac{n}{64}+1\right\rceil\cdot(15+44)+5n}\right)=$$

$$\lim_{n\to\infty}\left(\frac{3n}{69+5,9219n}\right)=0,5066\ \ FLOP/ciclo$$

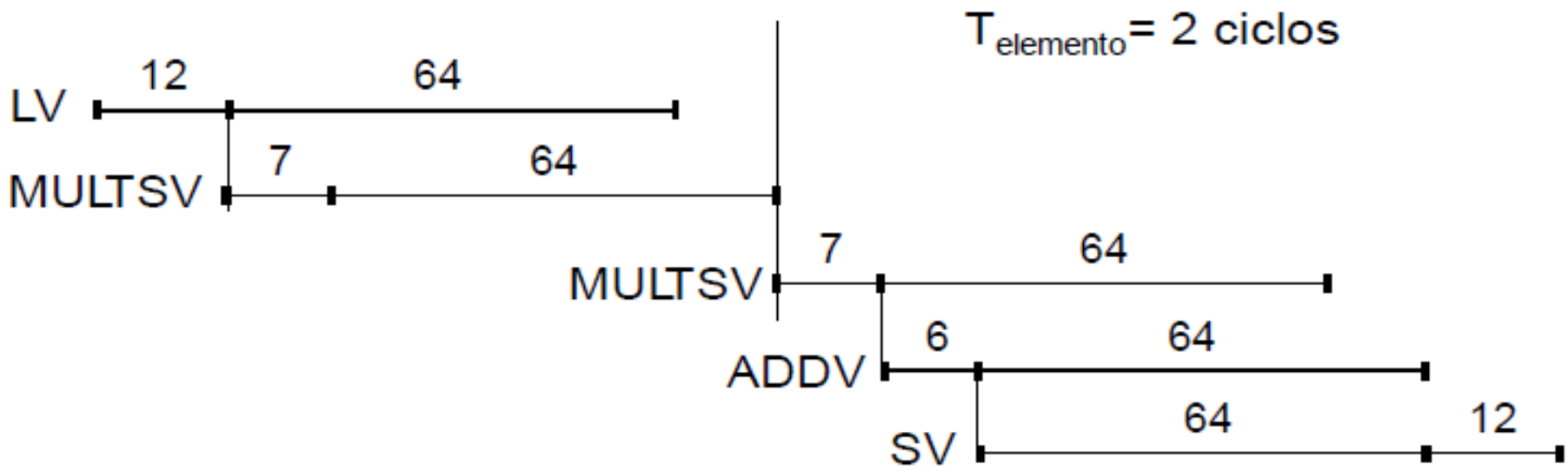
Para expresar R_∞ en FLOPS, habría que multiplicar el valor en FLOP/ciclo por la frecuencia del procesador. Se tendría así

$$R_\infty = 0{,}506 \text{ FLOP/ciclo} \cdot (500 \cdot 10^6) \text{ Hz}$$

$$R_\infty = 253 \text{ MFLOPS}$$

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

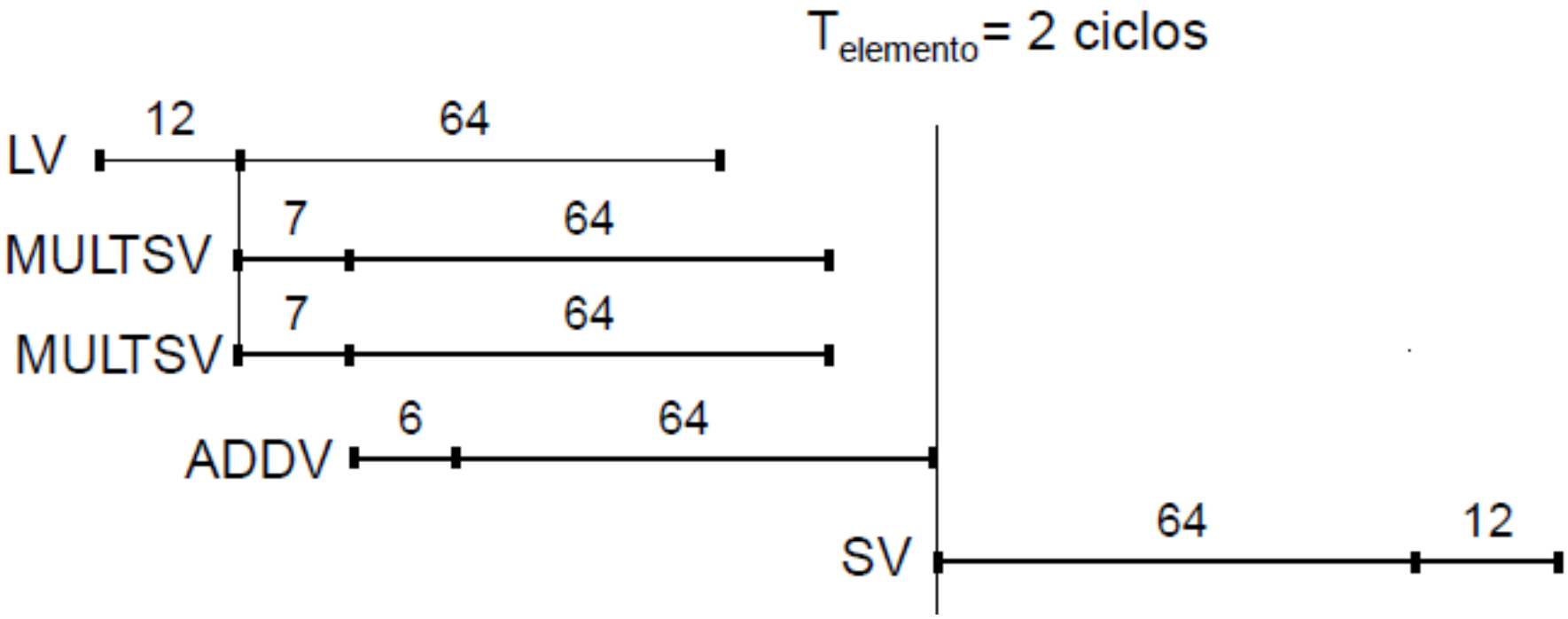
$$R_\infty = \lim_{n\to\infty}\left(\frac{3n}{T_n}\right) = \lim_{n\to\infty}\left(\frac{3n}{10+\left\lceil\frac{n}{64}\right\rceil\cdot(15+44)+2n}\right) = \lim_{n\to\infty}\left(\frac{3n}{10+\left\lceil\frac{n}{64}+1\right\rceil\cdot(15+44)+2n}\right) =$$

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

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
```


$T_{elemento}$ = 2 ciclos
LV
12
64
MULTSV
7
64
MULTSV
7
64
ADDV
6
64
SV
64
12

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

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

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

Centro Asociado Palma de Mallorca

Febrero 2013-1ª Problema 2

Tutor: Antonio Rivero Cuesta

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
```

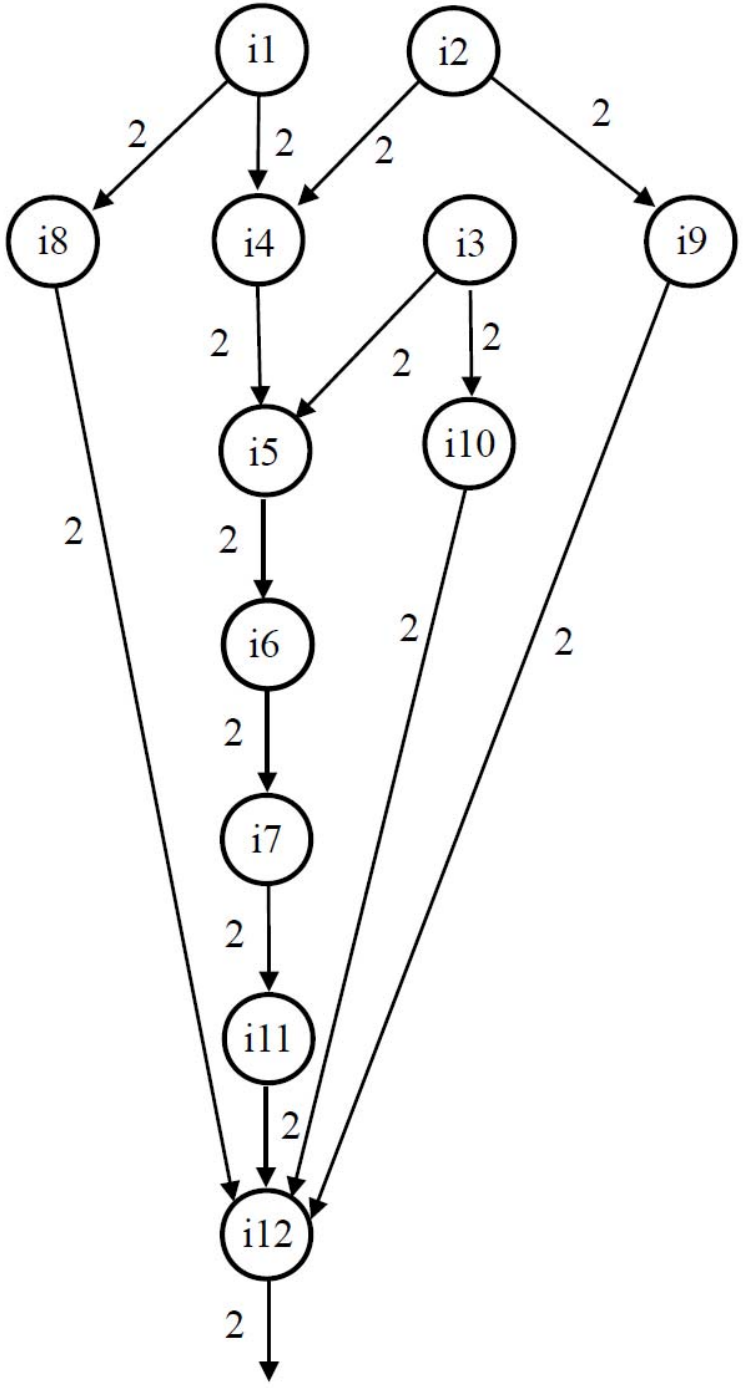
a) Obtenga el diagrama de flujo de datos de la secuencia. Considere que todas las instrucciones tienen una latencia de 2 ciclos.

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?

Apartado 1



Apartado 2

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

Apartado 3

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, 8(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	
	ADDD 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,#8	SD 24(R1),F30	JMP inicio

Apartado 4

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.

Centro Asociado Palma de Mallorca

Febrero 2013 - 2ª Problema 3

Tutor: Antonio Rivero Cuesta

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 100 MHz.
- T_{base} de 10 ciclos y T_{bucle} de 15 ciclos.

Se pretende ejecutar el siguiente bucle:

```
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 $N_{1/2}$.

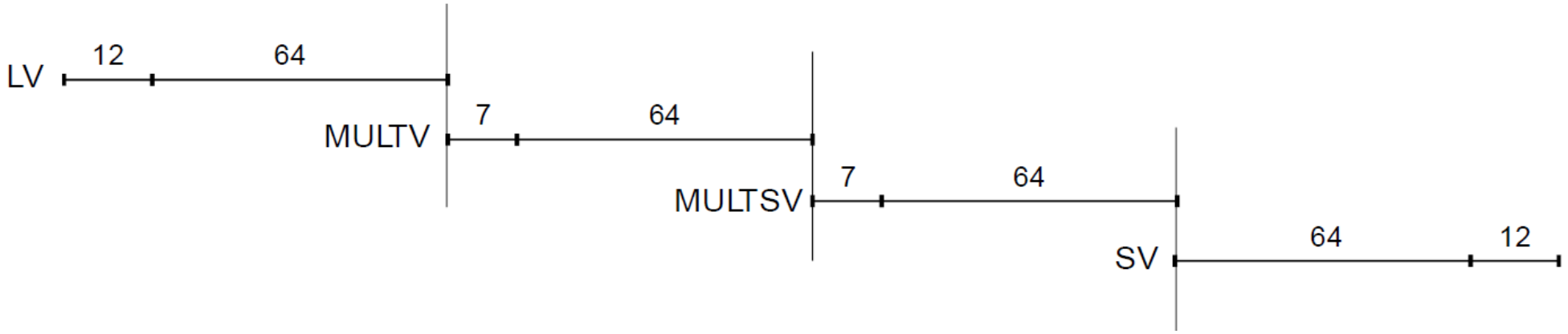
c) Ahora dispone de encadenamiento entre las unidades funcionales y solapamiento entre convoyes dentro de la misma iteración. ¿Cuál es el nuevo valor de ciclos por elemento?

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 4:  SV       R2, V4
```

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

$T_{elemento}$ = 4 ciclos



Dado que hay cuatro convoyes, $T_{elemento}$ es 4 ciclos y el $T_{arranque}$ total es igual a la suma de los tiempos de arranque visibles de los cinco convoyes.

Esto es:

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

$$T_{arranque} = (12 + 2*7 + 12) \text{ ciclos} = 38 \text{ ciclos}$$

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

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

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

$$T_{1000} = 10 + \left\lceil \frac{1000}{64} \right\rceil \cdot (15 + 38) + 4 \cdot 100 = 10 + 2 \cdot (15 + 38) + 4 \cdot 100 = 516 \ \ ciclos$$

$$R_{1000} = \frac{2 \cdot 100}{T_{100}} = \frac{200}{516} = 0,3875 \ {}^{FLOP}\!/_{ciclo}$$

El rendimiento expresado en FLOP/ciclo es

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

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 son dos, una multiplicación vectorial y una multiplicación vectorial-escalar, se tiene

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

$$\lim_{n\to\infty}\left(\frac{2n}{63+4,828\cdot n}\right)=0,41425\ \ FLOP/ciclo$$

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:

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

Por otra parte, sabemos que:

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

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

$$R_n = \left(\frac{2n}{10 + \left\lceil \frac{n}{64} + 1 \right\rceil \cdot (15+38) + 4n} \right) = \left(\frac{2n}{63 + 4{,}828 \cdot n} \right)$$

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

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

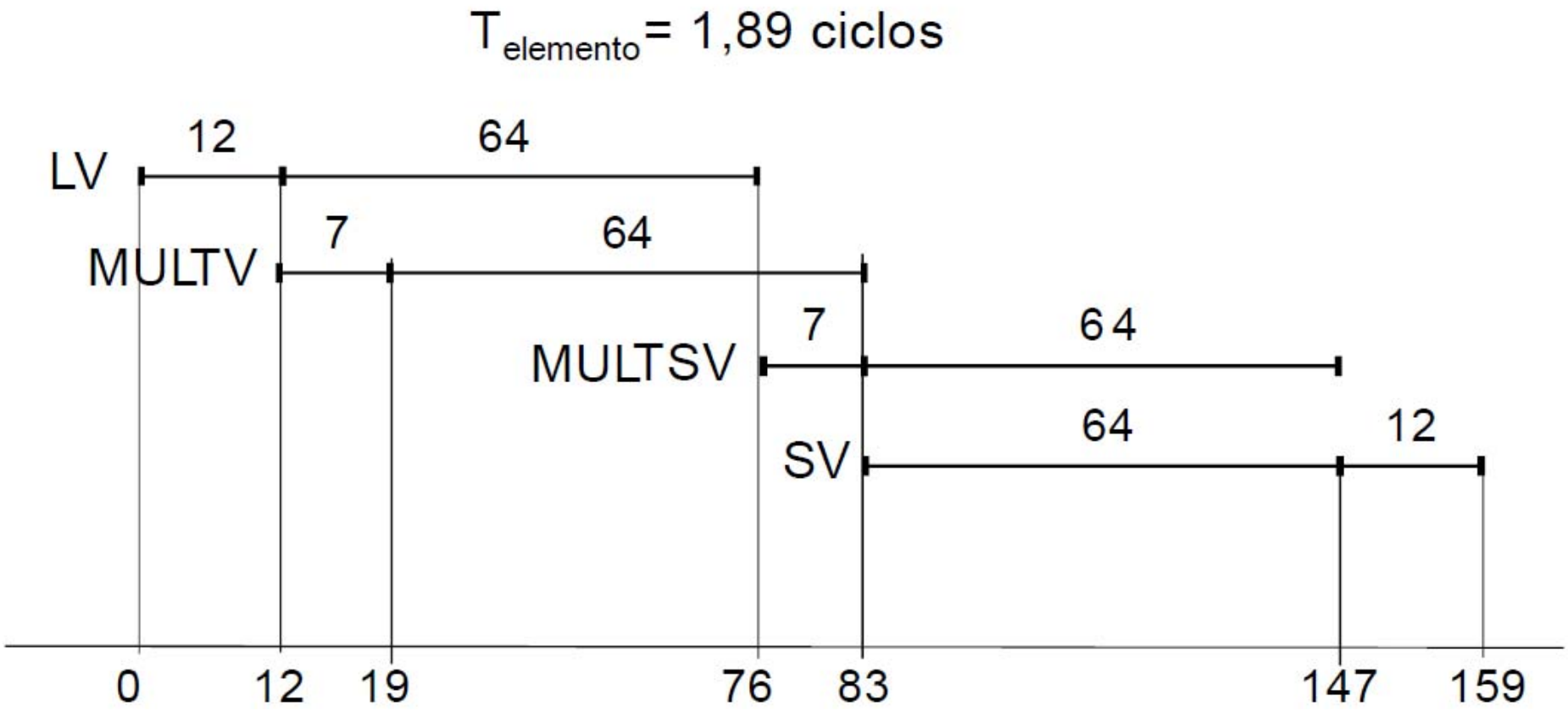
Por lo tanto:

$N_{1/2} = 13$.

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

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

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:

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

$$T_{arranque} = (12 + 2*7 + 12) \text{ ciclos} = 38 \text{ ciclos}$$

Por lo tanto, se tiene:

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

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

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

Centro Asociado Palma de Mallorca

Septiembre 2013-R Problema 3

Tutor: Antonio Rivero Cuesta

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).
- Una operación en coma flotante (3 ciclos).
- Una operación entera/salto (1 ciclo).

Las instrucciones de manipulación de predicados se consideran operaciones enteras.

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.

	2 ciclos	**3 ciclos**	**1 ciclo**
	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,Fb (p4)	
12		ADDD Fx,Fx,Fb (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.

	2 ciclos	**3 ciclos**	**1 ciclo**
	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,Fb (p4)	
10		ADDD Fx,Fx,Fb (p3)	
11			
12			fin:
13			BNEZ Ra, inicio
14			SD 0(Rx), Fx

Exámenes

TEMA 4

Centro Asociado Palma de Mallorca

Febrero 2012 - 1 Problema 3

Tutor: Antonio Rivero Cuesta

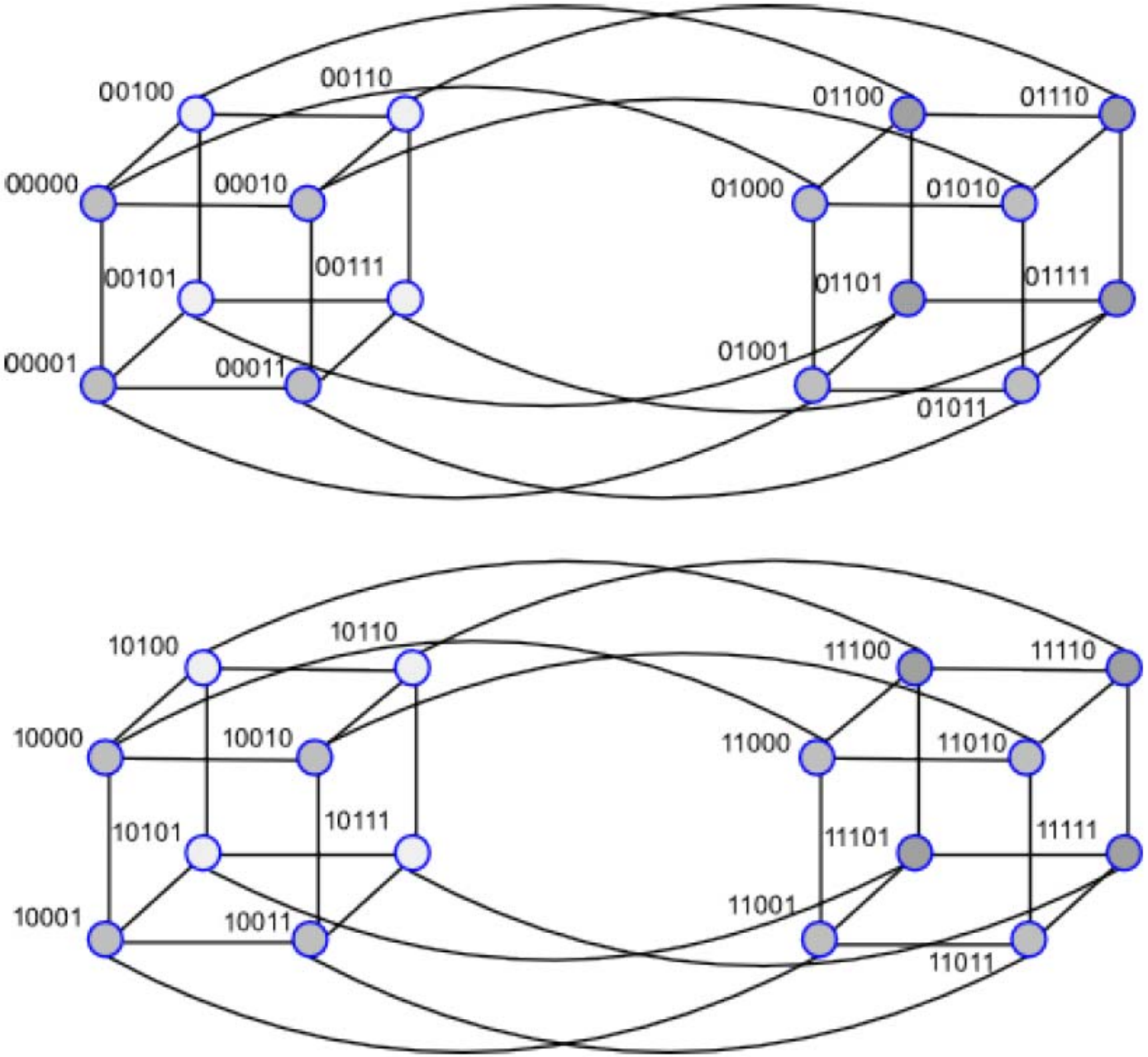
Dada una red con topología de hipercubo con dimensión $d = 5$, se pide que:

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 más corto para comunicar ambos procesadores.

c) Describa y calcule la conectividad de arco de dicha red.

a)


00100
00110
01100
01110
00000
00010
01000
01010
00101
00111
01101
01111
00001
00011
01001
01011
10100
10110
11100
11110
10000
10010
11000
11010
10101
10111
11101
11111
10001
10011
11001
11011

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.

Centro Asociado Palma de Mallorca

Febrero 2012 - 2 Problema 3

Tutor: Antonio Rivero Cuesta

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.

a) La secuencia de acciones y etiquetas se describe en la siguiente tabla:

	CPU A		**CPU B**	
Instrucciones	**Etiquetas**	**Acciones**	**Etiquetas**	**Acciones**
			inválida(X)	
A: read(X)	compartida(X)	C_lectura(X)	compartida(X)	
A: write(X)	sucia(X)	C_escritura(X)	Inválida X)	

b) Los posibles problemas son que la CPU B realice una escritura o una lectura sobre la variable X justo después de la lectura de la CPU A sobre X y antes de la escritura de la CPU A.

El caso de la lectura de la CPU B sobre X no presenta problemas ya que el valor que tiene la CPU A no se modifica y las operaciones realizadas siguen siendo atómicas.

	CPU A		CPU B	
Instrucciones	**Etiquetas**	**Acciones**	**Etiquetas**	**Acciones**
			inválida(X)	
A: read(X)	compartida(X)	C_lectura(X)	compartida(X)	
B: read(X)	compartida(X)		compartida(X)	
A: write(X)	sucia(X)	C_escritura(X)	inválida (X)	

c) Sin embargo, el caso de una escritura de la CPU B sobre X, sí presenta problemas. La escritura de CPU B sobre X invalida la copia de caché de la CPU A.

Cuando la CPU A quiera realizar la escritura, el dato será servido por la caché de la CPU B en lugar de la copia local de A, dando un resultado diferente del esperado (a no ser que el valor escrito por B sea igual al escrito por A).

	CPU A		CPU B	
Instrucciones	**Etiquetas**	**Acciones**	**Etiquetas**	**Acciones**
			inválida(X)	
A: read(X)	compartida(X)	C_lectura(X)	compartida(X)	
B: write(X)	inválida(X)		sucia(X)	C_escritura(X)
A: write(X)	sucia(X)	C_escritura(X)	inválida (X)	

Centro Asociado Palma de Mallorca

Septiembre 2012 Problema 3

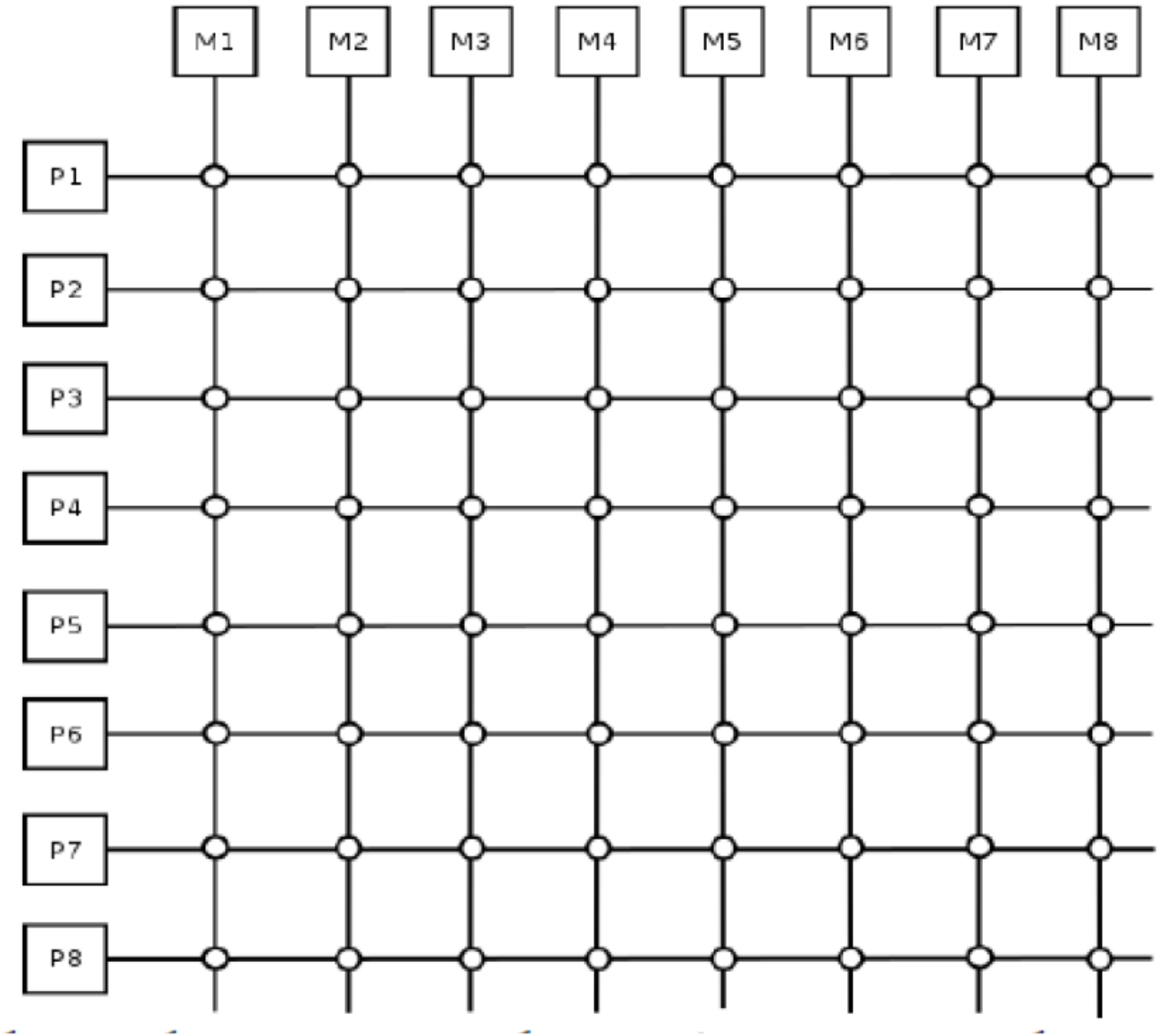
Tutor: Antonio Rivero Cuesta

Dibuje una red de tipo ***crossbar*** de 8 x 8 elementos y describa cada uno de los componentes que forman la red.

¿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?

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***.


M1
M2
M3
M4
M5
M6
M7
M8
P1
P2
P3
P4
P5
P6
P7
P8

Apartado 1

Los elementos de la red son:

- Los procesadores (representados por P1...P8).
- Los elementos de memoria (representados por M1...M8).
- Los conmutadores (representados con círculos)
- Las líneas de conexión entre conmutadores (representadas por líneas).

Apartado 2

Las principales diferencias son el número de elementos conectados y el tipo de conexión entre ellos.

En el caso de la red ***crossbar*** se conectan 8 procesadores con 8 elementos de memoria (16 elementos en total), y en el caso de la red ***mesh*** se conectan 64 elementos.

Las conexiones de la red ***crossbar*** se pueden considerar punto a punto, mientras que las conexiones de la red ***mesh*** dependen del número de saltos que se deba realizar entre los elementos que se desea comunicar.

Apartado 3

Utilizando el algoritmo de enrutamiento ***store-and-forward***, el tiempo de transferencia de un mensaje sigue la siguiente fórmula:

$$t = t_s + (m \cdot t_w + t_h) \cdot l$$

donde ***t*** es el tiempo de transferencia del mensaje, t_s es el tiempo de inicialización, ***m*** es el tamaño del mensaje, t_w el tiempo de transferencia por palabra, t_h el tiempo de salto y ***l*** el número de saltos.

Para el caso de la red ***crossbar***, sabemos que las comunicaciones entre cualquiera de los elementos requieren de un único salto, por lo que la transferencia del mensaje será:

$$10 + (10 \cdot 3 + 1) \cdot 1 = 41ms.$$

En el caso de la red ***mesh***, el máximo número de saltos corresponde al diámetro de la red que en este caso es $2 \cdot \left(\sqrt{p} - 1\right) = 2 \cdot \left(\sqrt{64} - 1\right) = 14$. Por tanto:

$$10 + (10 \cdot 3 + 1) \cdot 14 = 444ms$$

Centro Asociado Palma de Mallorca

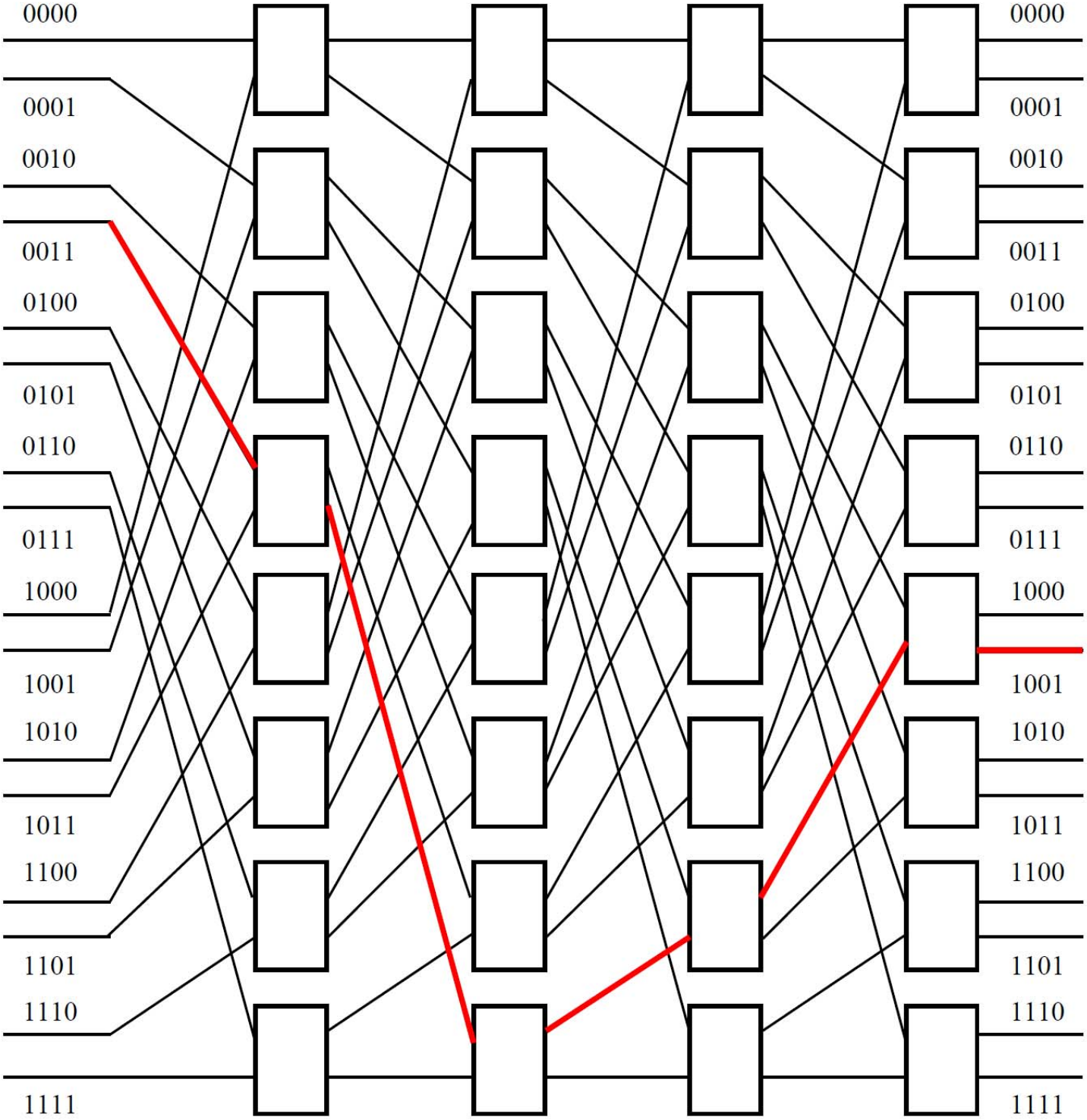
Septiembre 2012 - R Problema 3

Tutor: Antonio Rivero Cuesta

Dibuje una red de tipo Omega de 16 entradas y 16 salidas.

Describa razonadamente el protocolo para enviar un mensaje desde el nodo de entrada 3 al nodo de salida 9.

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.


0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Para ir del procesador 0011 al 1001:

Desde el procesador de origen 0011 hasta el procesador destino 1001.

1 INFERIOR

0 SUPERIOR

0 SUPERIOR

1 INFERIOR

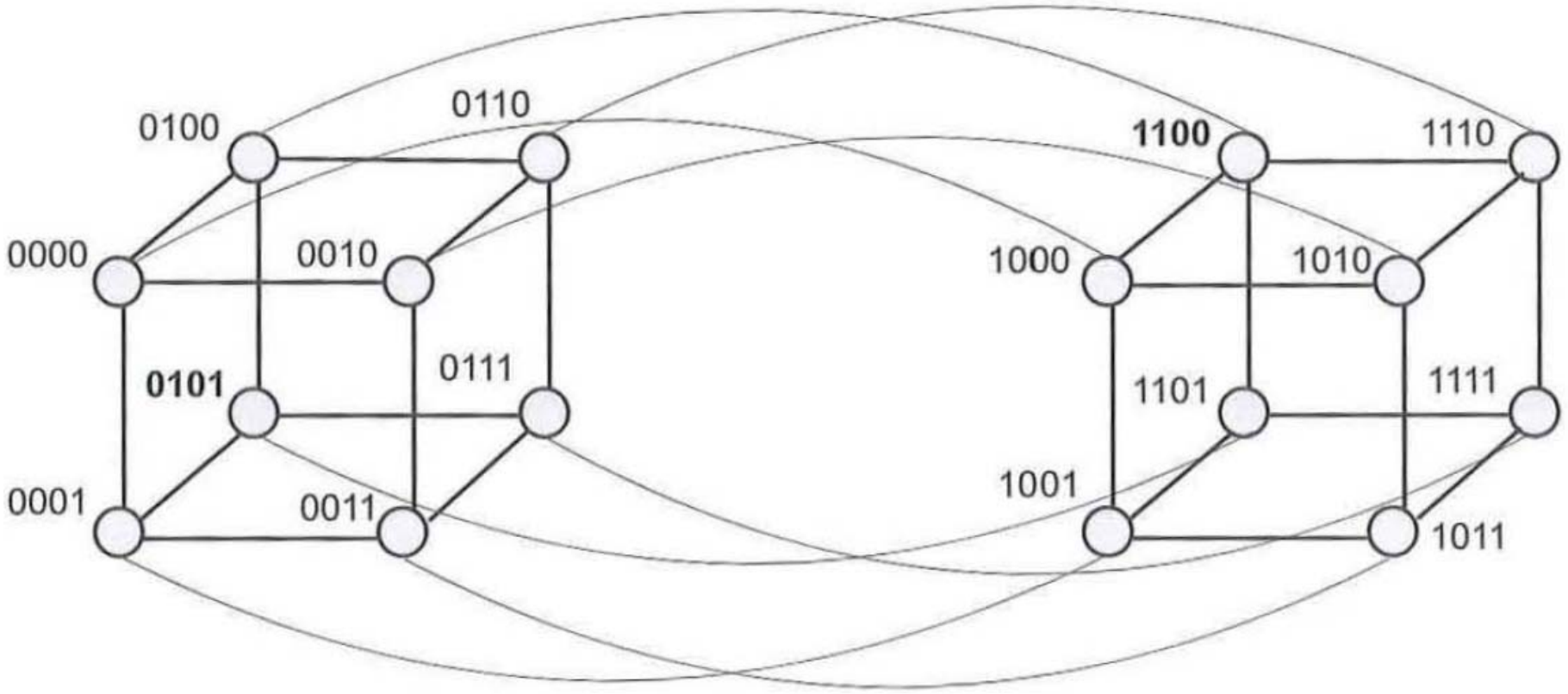
Es lo que está en trazo grueso en ROJO en la figura.

Centro Asociado Palma de Mallorca

Febrero 2013 - 2 Problema 1

Tutor: Antonio Rivero Cuesta

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.

a) Es una red estática hipercubo de dimensión 4.

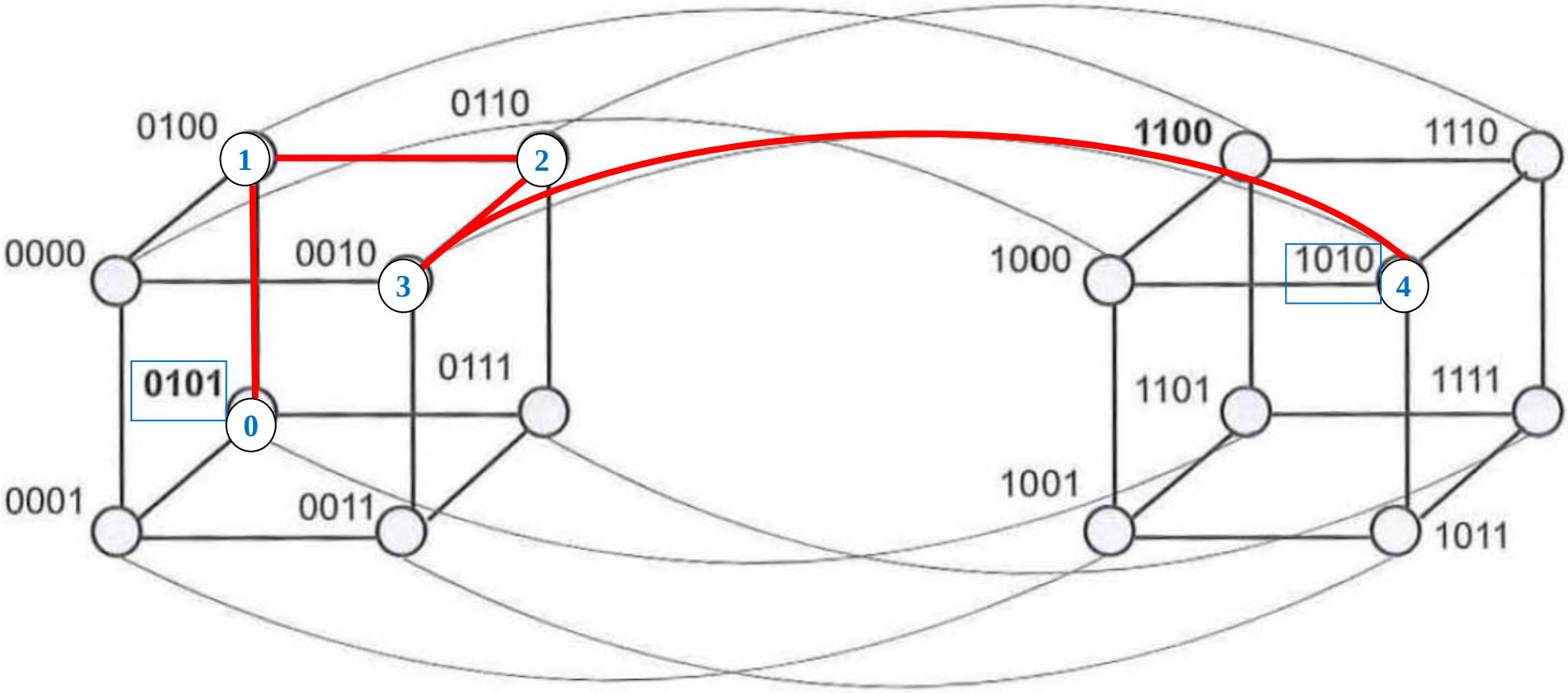
b) La distancia Hamming: $0101 \otimes 1010 = 1111$.
Es 4, porque tenemos cuatro unos. Por lo tanto:

Del nodo 010**1** nos movemos a 010**0**

Del nodo 01**0**0 nos movemos a 01**1**0

Del nodo 0**1**10 nos movemos a 0**0**10

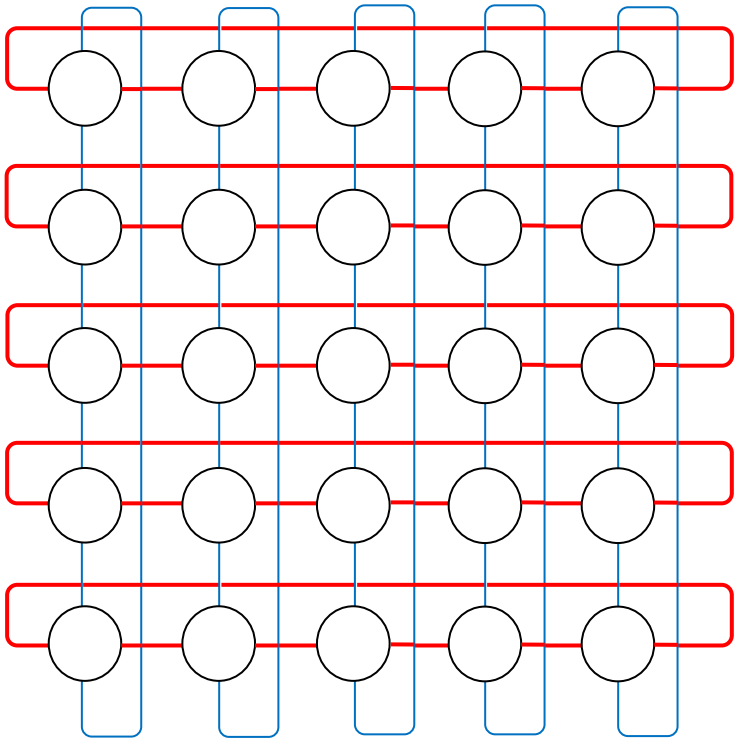
Del nodo **0**010 nos movemos a **1**010


0100
0110
1100
1110
1
2
0000
0010
3
1000
1010
4
0101
0111
1101
1111
0
0001
0011
1001
1011

Dibuje una red estática con los siguientes valores en sus parámetros:

- Diámetro = 4;
- Conectividad de arco = 4;
- Coste = 50.

Diámetro = 4; Conectividad de arco = 4; Coste = 50.

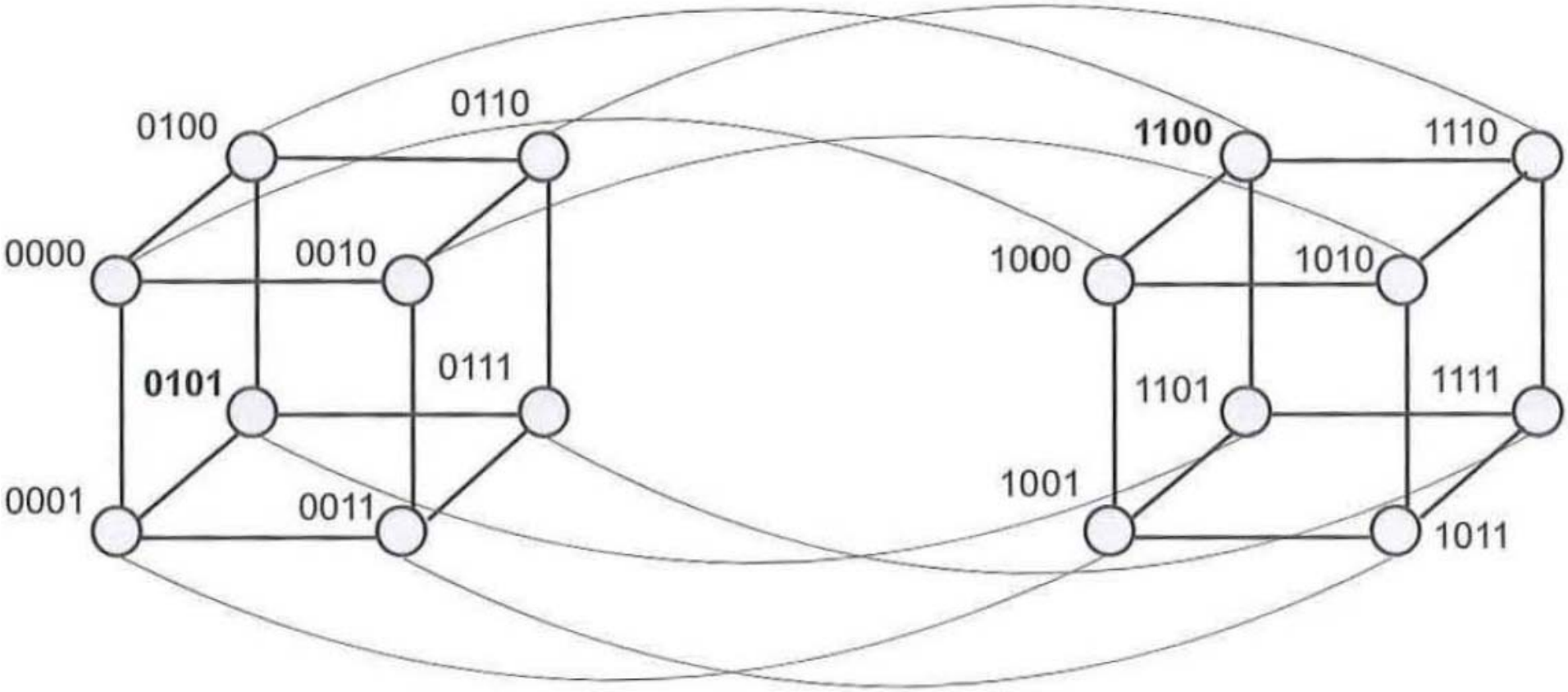


Centro Asociado Palma de Mallorca

Septiembre 2012 - R Problema 3

Tutor: Antonio Rivero Cuesta

Dada la siguiente red estática:



a) 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.

La distancia Hamming: $0000 \otimes 1111 = 1111$.

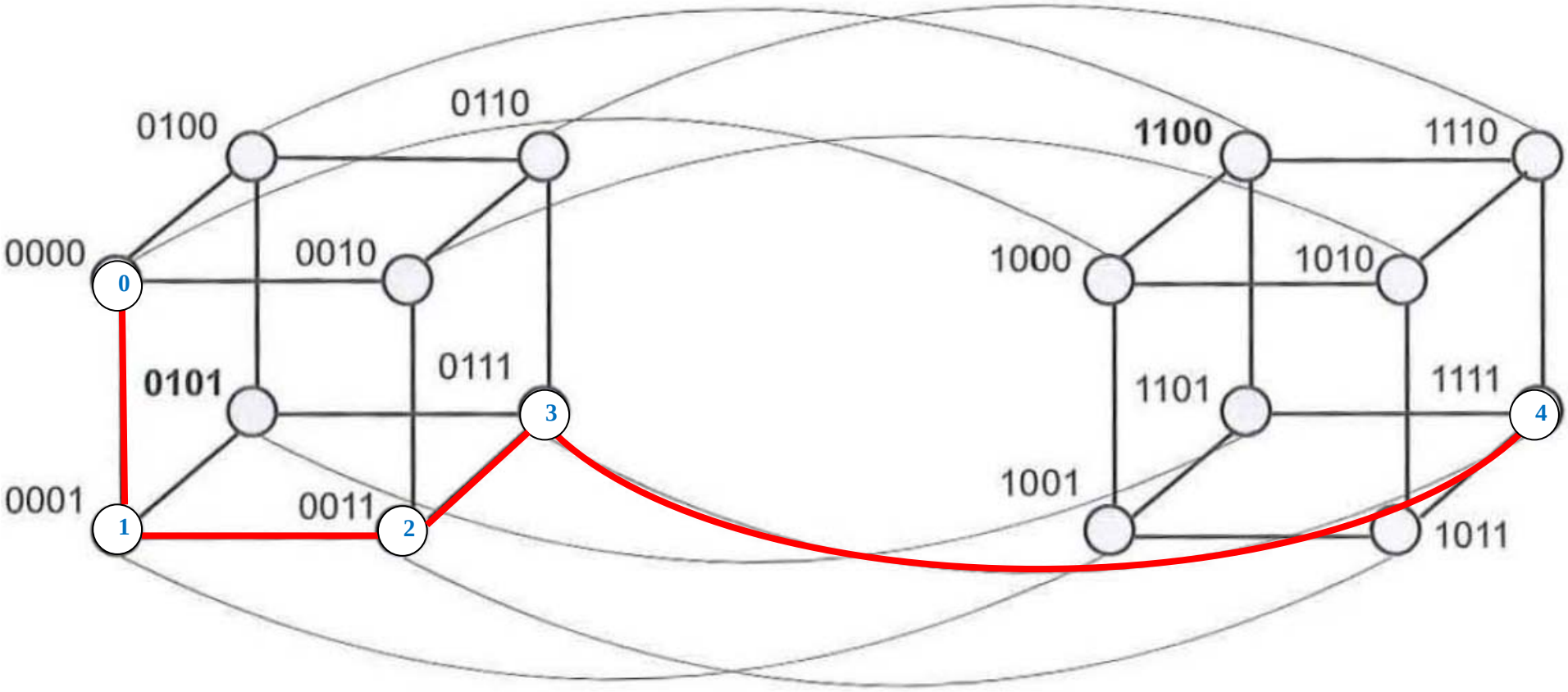
Es 4, porque tenemos cuatro unos. Por lo tanto:

Del nodo 000**0** nos movemos a 000**1**

Del nodo 000**1** nos movemos a 00**1**1

Del nodo 00**1**1 nos movemos a 0**1**11

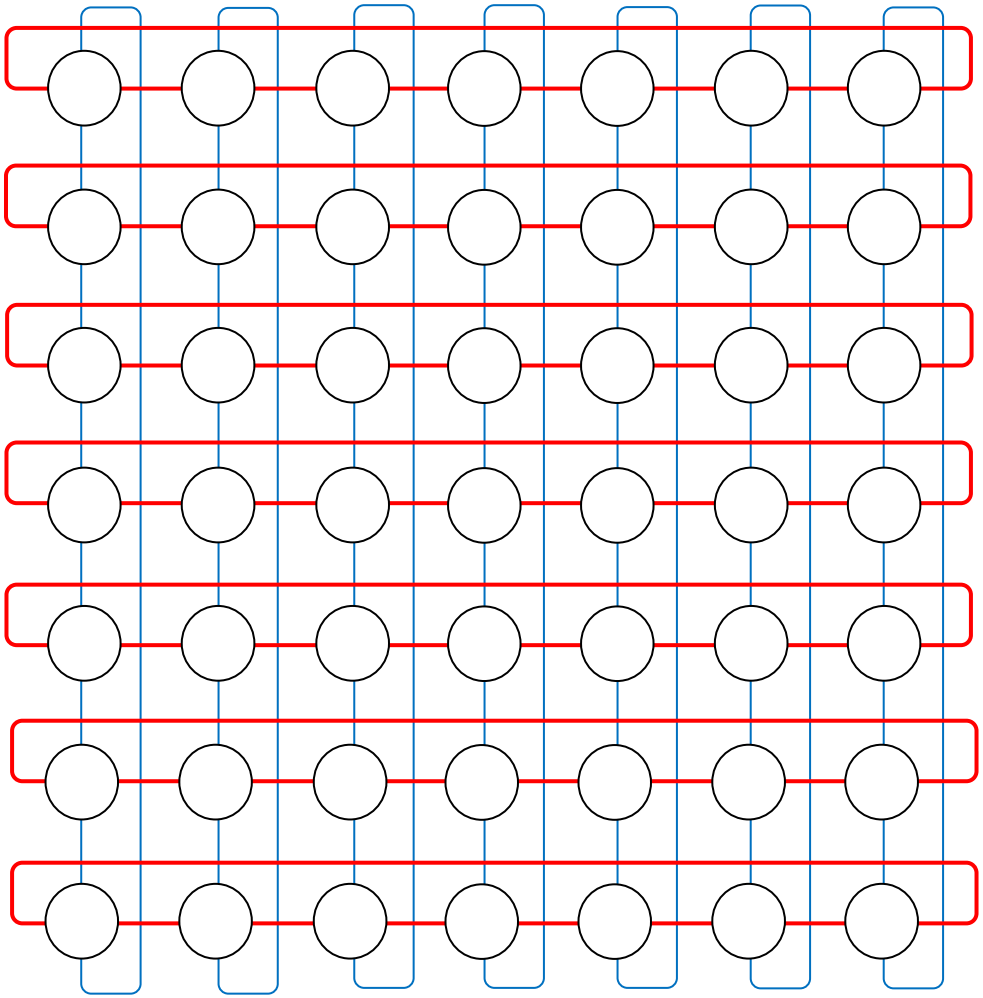
Del nodo 0**1**11 nos movemos a **1**111


0100
0110
1100
1110
0000
0010
1000
1010
0
0101
0111
1101
1111
3
4
0001
0011
1001
1011
1
2

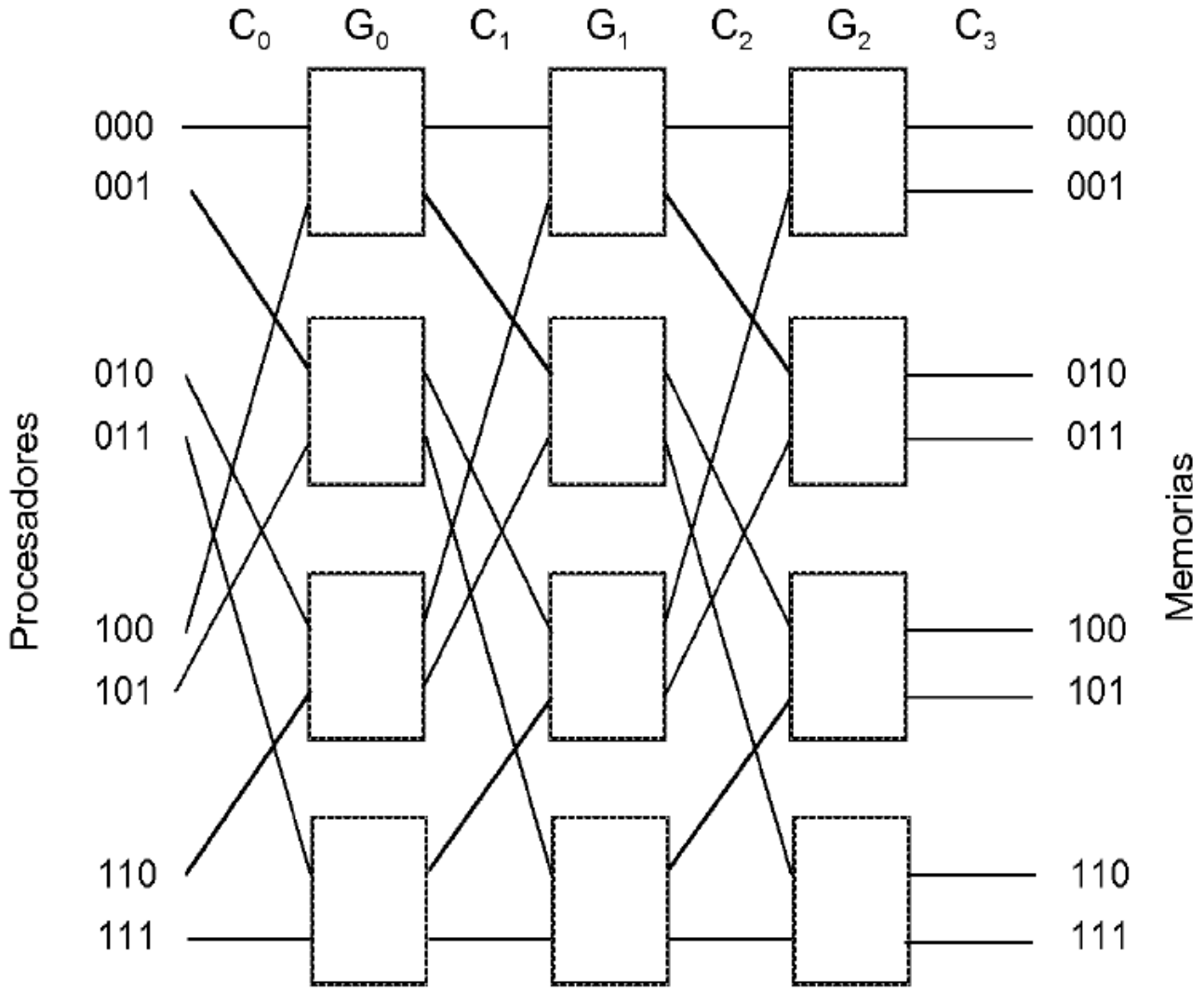
b) Dibuje una red estática con los siguientes valores en sus parámetros:

- Diámetro = 6;
- Conectividad de arco = 4;
- Coste = 98
- Diámetro = $2 \cdot \left\lfloor \frac{\sqrt{p}}{2} \right\rfloor$
- Conectividad de arco = 4.
- Coste = $2 \cdot p$

Diámetro = 6; Conectividad de arco = 4; Coste = 98.



Dada la siguiente red dinámica:



¿De qué red se trata?

Es una red omega de 8 entradas y 8 salidas.

Explique razonadamente la conmutación de cada conmutador de la red para enviar un mensaje del procesador 101 al banco de memoria 100

Para enviar un mensaje del procesador 101 al banco de memoria 100, hacemos lo siguiente:

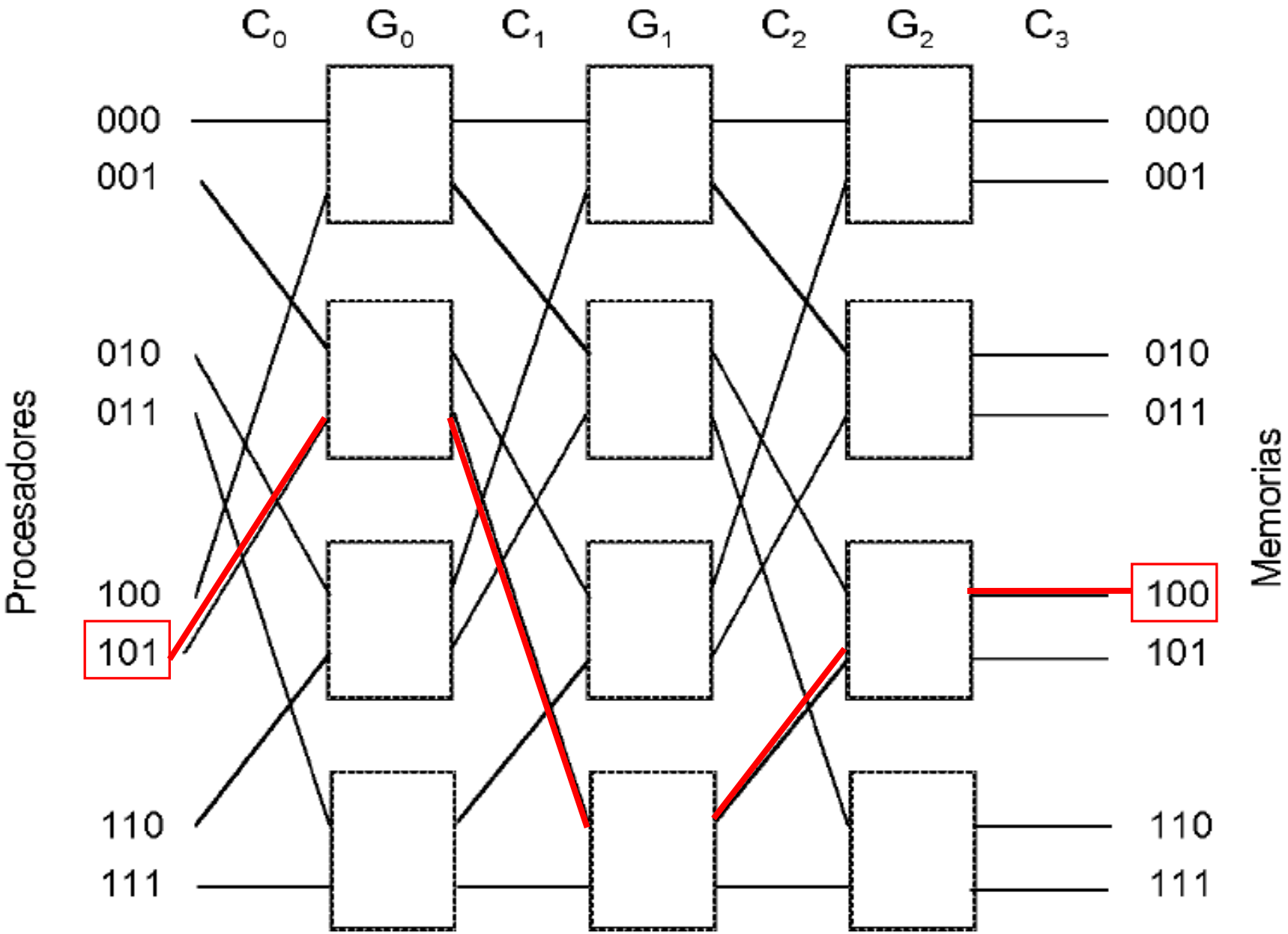
Los conmutadores de cada etapa deciden el camino por el que transmitir el paquete dependiendo del valor del bit de la dirección destino correspondiente a la etapa actual. Si

el bit es 0, se encamina por la salida superior, y si es 1, se utiliza la salida inferior.

1 INFERIOR

0 SUPERIOR

0 SUPERIOR


C0
G0
C1
G1
C2
G2
C3
000
001
010
011
100
101
110
111
Procesadores
000
001
010
011
100
101
110
111
Memorias

Centro Asociado Palma de Mallorca

Febrero 2014 – 2ª Problema 4

Tutor: Antonio Rivero Cuesta

Dibuje una red estática con los siguientes valores en sus parámetros:

Diámetro = 8; Conectividad de arco = 4; Coste = 162;

Se trata de una red bidimensional mesh cerrada.