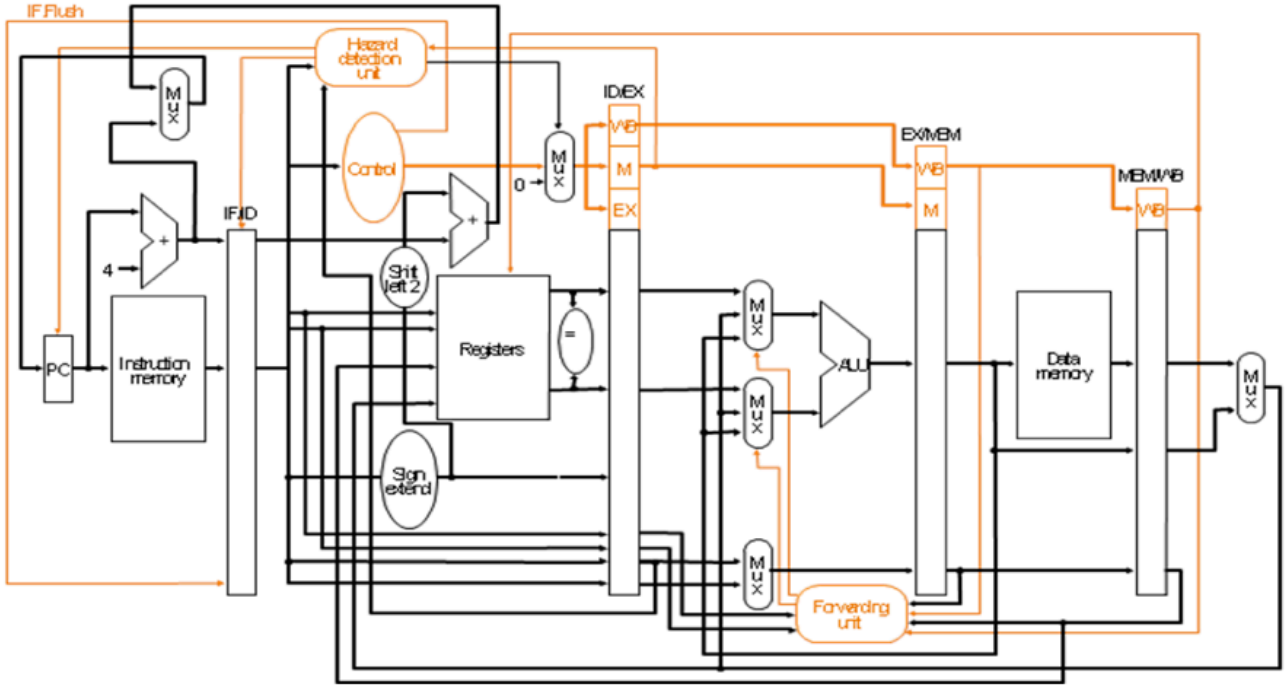
Problemas de Repaso: Temas 1 y 2

P1 .- El microprocesador MIPS, estudiado en teoría, posee una unidad de adelantamiento de datos (forwarding unit) que solo permite adelantar los datos de las etapas MEM y WB a la etapa de ejecución (EX):



NOTA: Los registros de las diferentes etapas se llaman PC, IF/ID, ID/EX, EX/MEM y MEM/W respectivamente. El multiplexor de la entrada superior de la ALU se denomina AnticiparA y el de la entrada inferior de la ALU se denomina AnticiparB. En estos multiplexores de tres entradas la entrada superior se corresponde con la selección 00, la del centro con la 01 y la inferior con 10.

La lógica (pseudocódigo) correspondiente a la unidad de adelantamientos es:

Adelantamientos desde la etapa MEM	***Adelantamientos desde la etapa WB***
if (EX/MEM.EscrReg **and** EX/MEM.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd = ID/EX.Reg.Rs) **then** AnticiparA = 10 **else** AnticiparA = 00 **if** (EX/MEM.EscrReg **and** EX/MEM.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd = ID/EX.Reg.Rt) **then** AnticiparB = 10 **else** AnticiparB = 00	**if** (MEM/WB.EscrReg **and** MEM/WB.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd ≠ ID/EX.Reg.Rs **and** MEM/WB.Reg.Rd = ID/EX.Reg.Rs) **then** AnticiparA = 01 **else** AnticiparA = 00 **if** (MEM/WB.EscrReg **and** MEM/WB.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd ≠ ID/EX.Reg.Rt **and** MEM/WB.Reg..Rd = ID/EX.Reg.Rt) **then** AnticiparB = 01 **else** AnticiparB = 00

Tenga en cuenta que, como se representa en la figura, para mejorar las penalizaciones en los saltos se ha cambiado el diseño inicial, introduciendo en la etapa ID, un comparador que permite evaluar la condición de saltos, un sumador para calcular el destino de salto y un multiplexor para actualizar el PC caso de que sea necesario al final de esta etapa.

Problemas de Repaso: Temas 1 y 2

Suponga que se ejecuta la siguiente secuencia de código:

I1: Sub r1, r2, r3
I2: Add r5, r4, r1
I3: Beq r1, r5, 10

a) Indique los riesgos que presenta la secuencia de código anterior:

RAW : I2 con I1 por R1
RAW : I3 con I1 por R1
RAW : I3 con I2 por R5
Riesgo de control por salto BEQ

b) Complete el cronograma indicando los adelantamientos que se han podido realizar.

	1	2	3	4	5	6	7	8
SUB R1,R2,R3	IF	ID	EX	M				
ADD R5,R4,R1		IF	ID	EX				
BEQ R1,R5, 10			IF	ID				

Sol:

	1	2	3	4	5	6	7	8
SUB R1,R2,R3	IF	ID	EX	M	W			
ADD R5,R4,R1		IF	ID	EX	M	W		
BEQ R1,R5, 10			IF	ID	ID	**ID**	EX	MEM

Se puede resolver el riesgo RAW entre SUB y ADD por R1 con el adelantamiento desde EX/MEM cuando EX/MEM.Reg.Rd = ID/EX.Reg.Rt = R1 y se activa la línea 01 del multiplexor anticipar A.

El riesgo de datos RAW de BEQ no se resuelve con lo implementado en la tabla y será necesario parar 2 ciclos el pipeline para que funcione correctamente.

c) En el ciclo 4 (Instrucción ADD en la etapa EX) indique los valores de

ID/EX.Reg.Rs =
ID/EX.Reg.Rt =
ID/EX.Reg.Rd =
EX/MEM.Reg.Rd =
MEM/W.Reg.Rd =
Multiplexor AnticiparA=
Multiplexor AnticiparB=

Sol:
ID/EX.Reg.Rs = r4
ID/EX.Reg.Rt = r1
ID/EX.Reg.Rd = r5
EX/MEM.Reg.Rd = r1
MEM/W.Reg.Rd =x
Multiplexor AnticiparA= 00
Multiplexor AnticiparB= 10

d) En caso de ser necesario parar la ejecución de una instrucción en el *pipeline* de este procesador, sin que se detenga el resto, indique como lo realizaría.

Con la unidad de detección de riesgos al detectar la dependencia de BEQ para lo que es necesario utilizar como fuente las señales:
IF/ID Reg.Rs
IF/ID Reg.Rt
ID/EX.EscrReg
ID/EX.Reg.Rd
EX/MEM.EscrReg
EX/MEM.Reg.Rd

y cuando se detecta el riesgo, actuar (entrada disable del reg) sobre los registros PC, IF/ID para mantener la captura. Por otra parte se inserta un 0 (Código NOP) como entrada en el multiplexor de control y permite que avance un NOP desde la etapa 2.
Se permite el funcionamiento de las instrucciones cargadas en las etapas 3 a 5.

Problemas de Repaso: Temas 1 y 2

P2.- A la hora de desarrollar un sistema basado en procesador MIPS segmentado, como el estudiado en clase (5 etapas: IF, ID, EX, M y WB), se están considerando 2 tipos de arquitectura: Von Neumann y Harvard. Para evaluar ambas arquitecturas se ejecuta el siguiente programa:

```
I1: ld r1, 100(r0)
I2: ld r2, 104(r0)
I3: ld r3, 108(r0)
I4: add r4, r2, r1
I5: add r5, r4, r3
I6: store r5, 112(r0)
```

Se pide completar los cronogramas adjuntos. Considere que las memorias son ideales, no hay adelantamiento de datos, y que es posible escribir y leer un mismo registro en un ciclo de reloj.

Von Neumann

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I1:	IF*	ID	EX	M*	WB												
I2:		IF*	ID	EX	M*	WB											
I3:			IF*	ID	EX	M*	WB										
I4:							IF*	ID	EX	M	WB						
I5:								IF*	(ID)	(ID)	ID	EX	M	WB			
I6:									IF*	(ID)	(ID)	(ID)	(ID)	ID	EX	M*	WB

Harvard

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I1:	IF	ID	EX	M	WB												
I2:		IF	ID	EX	M	WB											
I3:			IF	ID	EX	M	WB										
I4:				IF	(ID)	ID	EX	M	WB								
I5:					(IF)	IF	(ID)	(ID)	ID	EX	M	WB					
I6:							IF	(IF)	(IF)	(ID)	(ID)	ID	EX	M	WB		

Problemas de Repaso: Temas 1 y 2

P3. Considerar el siguiente procesador RISC con un CPI de 1,4 ciclos (teniendo en cuenta fallos de caché, TLB y riesgos de datos insalvables por adelantamiento) segmentado en 9 etapas:

- **F1** Captura de instrucción 1. Envía la dirección de la instrucción a la I-caché.
- **F2** Captura de instrucción 2. Lee la instrucción de la I-caché.
- **D1** Decodificación de la instrucción. Calcula la dirección destino en saltos.
- **R1** Lectura de operandos desde registros.
- **A1** Comienza la operación en la ALU. Resuelve la condición de salto y actualiza BTB
- **A2** Termina la operación con la ALU.
- **M1** Captura de datos 1. Envía la dirección efectiva a la D-caché.
- **M2** Captura de datos 2. Lee/escribe el dato de/en la D-caché.
- **W1** Escribe resultado en un registro.

Un código de benchmarking utilizado, posee un porcentaje de saltos condicionales del 10%, de los cuales el 30% son efectivos, ~~y además el 2% de las instrucciones son saltos incondicionales~~. Suponiendo que no existen otros riesgos que los de control, se pide:

a) Determinar el CPI real si la predicción es estática i) efectiva o ii) no efectiva. En ambos casos, cuando se detecta la instrucción de salto, el procesador no se detiene hasta poder ejecutar la predicción elegida.

b) Determinar el CPI real si se mejora el caso anterior con una estructura BTB (Branch Target Buffer). Donde los saltos condicionales aciertan la predicción por el BTB el 90% (el 10% restante corresponde a fallos de predicción y arranque en frio (primera ocurrencia) del salto). ~~Y para el caso de los saltos incondicionales el fallo por primera ocurrencia representa el 1%~~.

Ejercicio a1. Para Justificar su respuesta, utilice las 4 tablas adjuntas que analizan la penalización en ciclos para los casos de predicción estática efectiva y no efectiva

Predicción estática No efectiva/Salto No efectivo: **0** ciclos de penalización

	T1	T2	T3													
Branch	F1	F2	D1(1	R1	A1(2	A2	M1	M2	W1							
N		F1	F2	D1	R1	A1	A2	M1	M2	W1						
N+1			F1	F2	D1	R1	A1	A2	M1	M2	W1					
N+2				F1	F2	D1	R1	A1	A2	M1	M2	W1				
N+3					F1	F2	D1	R1	A1	A2	M1	M2	W1			
N+3						F1	F2	D1	R1	A1	A2	M1	M2	W1		

1) Calcula dirección de destino. 2) resuelve condición

Predicción estática No efectiva/Salto efectivo: **4** ciclos de penalización

	T1	T2	T3													
Branch	F1	F2	D1(1	R1	A1(2	A2	M1	M2	W1							
N		F1	F2	D1	R1	--										
N+1			F1	F2	D1	--										
N+2				F1	F2	--										
N+3					F1	--										
T						F1	F2	D1	R1	A1	A2	M1	M2	W1		
T+1							F1	F2	D1	R1	A1	A2	M1	M2	W1	

Problemas de Repaso: Temas 1 y 2

Predicción estática efectiva/Salto No efectivo: **2** ciclos de penalización

Branch	F1	F2	D1(1	R1	A1(2	A2	M1	M2	W1							
N		F1	F2	D1	R1	A1	A2	M1	M2	W1						
N+1			F1	F2	D1	R1	A1	A2	M1	M2	W1					
N+2						F1	F2	D1	R1	A1	A2	M1	M2	W1		
N+3							F1	F2	D1	R1	A1	A2	M1	M2	W1	
T				F1	F2	--										
T+1					F1	--										

Predicción estática efectiva/Salto efectivo: **2** ciclos de penalización

Branch	F1	F2	D1(1	R1	A1(2	A2	M1	M2	W1							
N		F1	F2	D1	R1	--										
N+1			F1	F2	D1	--										
N+2																
T				F1	F2	D1	R1	A1	A2	M1	M2	W1				
T+1					F1	F2	D1	R1	A1	A2	M1	M2	W1			
T+2						F1	F2	D1	R1	A1	A2	M1	M2	W1		

NO ESTÁ EN EL EXAMEN. Para reflexionar.

a.2. ¿a qué tipo de salto es equiparable el salto incondicional?:
Predicción estática efectiva, salto efectivo. Es decir, tiene **2** ciclos de penalización.
Breve justificación:
Se conoce que es un salto incondicional en la etapa D1 y se conoce la dirección de salto. En ese momento hay dos instrucciones en el pipeline que hay que eliminar

a.3. Penalización de la estrategia predicción estática Efectiva y CPI total teniendo en cuenta los saltos

Si salto no efectivo (70%), 2 ciclos de penalización. Si salto efectivo (30%), 2 ciclos de penalización.

PenalizSaltos: 0.1 * (0.7*2 + 0.3*2) = 0.1*2.0 = 0.2

CPI_prefEf: 1.4 + 0.2 = 1.6

a.4. Penalización de la estrategia de predicción estática No Efectiva y CPI total teniendo en cuenta los saltos

Si salto no efectivo (70%), 0 ciclos de penalización. Si salto efectivo (30%), 4 ciclos de penalización.

PenalizSaltos: 0.1 * (0.7*0 + 0.3*4) = 0.1*1.2 = 0.12

CPI_prefNoEf: 1.4 + 0.12 = 1.52

Problemas de Repaso: Temas 1 y 2

NO ESTÁ EN EL EXAMEN. Para reflexionar. ¿como afectaría al resultado que el 2% de las instrucciones fuesen saltos incondicionales?

a.5. Penalización de la estrategia predicción estática Efectiva y CPI total teniendo en cuenta los saltos condicionales e incondicionales

Si salto no efectivo (70%), 2 ciclos de penalización. Si salto efectivo (30%), 2 ciclos de penalización.

PenalizSaltos: 0.1 * (0.7*2 + 0.3*2) + 0.02*2 = 0.1*2.0 + 0.04 = 0.2+.04 = 0.24

CPI_prefEf: 1.4 + 0.24 = 1.64

a.6. Penalización de la estrategia de predicción estática No Efectiva y CPI total teniendo en cuenta los saltos condicionales e incondifcionales

Si salto no efectivo (70%), 0 ciclos de penalización. Si salto efectivo (30%), 4 ciclos de penalización.

PenalizSaltos: 0.1 * (0.7*0 + 0.3*4) + 0.02*2 = 0.1*1.2 + 0.04 = 0.12 + .04 = 0.16

CPI_prefNoEf: 1.4 + 0.16 = 1.56

Ejercicio b1. Analice la penalización en el fallo y arranque en frio del BTB. Utilice las tablas para justificar la penalización en ciclos.

Existen 4 casos a considerar:

	Hasta conocer condición	Cuando conoce condición
1) No está en BTB y Salto Efectivo	Avanzar pipeline	Actualizar BTB. eliminar instrucciones incorrectas. Comenzar Instrucción correcta
2) No está en BTB y Salto NO Efectivo	Avanzar pipeline	Actualizar BTB. Seguir computando. No hay perdida de ciclos
3) En BTB y acierta predicción (sea efectiva o no efectiva)	Captura siguiente Instrucc. de BTB	Seguir computando. No hay perdida de ciclos
4) En BTB y NO acierta predicción (sea efectiva o no efectiva)	Captura (errónea-mente) siguiente Instrucc. de BTB	Actualizar BTB. eliminar instrucciones incorrectas. Comenzar Instrucción correcta

El enunciado sugiere que el 90% corresponde a los casos 3 y 2. Los casos 1 y 4 no pierden tiempo y representan el 10%. En la tabla se muestra la pérdida de tiempo.

@ Se acepta como correcto, suponer que el 90% son los casos 3. Y el 10% son los casos 1, 2 y 4. Calculándolo así 30% de los saltos son efectivos (caso 1 y 4), el 70% no efectivo (caso 2)

Problemas de Repaso: Temas 1 y 2

Penalización (por predicción incorrecta) y arranque en frio de salto Efectivo del salto condicional. Penaliza **4** ciclos

Ayuda: en el ciclo A1 se conoce efectivamente la condición y se puede actualizar el BTB

Branch	F1	F2	D1[i]	R1	A1[ii]	A2	M1	M2	W1							
N		F1	F2	D1	R1	--										
N+1			F1	F2	D1	--										
N+2				F1	F2	--										
N+3					F1	--										
T						F1	F2	D1	R1	A1	A2	M1	M2	W1		
T+1							F1	F2	D1	R1	A1	A2	M1	M2	W1	

i) Calcula dirección correcta de destino. ii) resuelve condición, salta y actualiza BTB

Explicación del caso 2). No pedido en el examen.

Penalización arranque en frio de salto condicional NO Efectivo. Penaliza **0** ciclos

	T1	T2	T3													
Branch	F1	F2	D1[i]	R1	A1[*	A2	M1	M2	W1							
N		F1	F2	D1	R1	A1	A2	M1	M2	W1						
N+1			F1	F2	D1	R1	A1	A2	M1	M2	W1					
N+2				F1	F2	D1	R1	A1	A2	M1	M2	W1				
N+3					F1	F2	D1	R1	A1	A2	M1	M2	W1			
N+3						F1	F2	D1	R1	A1	A2	M1	M2	W1		

* En el ciclo A1 se conoce efectivamente la condición y al ser NO efectiva, se actualiza el BTB, pero no es necesario eliminar instrucciones del pipeline

b2. Penalización debida de los saltos usando BTB y CPI total teniendo en cuenta los saltos

Para saltos condicionales, el BTB acierta el 90% de las veces con 0 ciclos de penalización. Y el 10% penaliza 4 ciclos.

PenalizSaltosBTB: 0.1 * (0.9*0 + 0.1*4) = 0.1*0.4 = 0.04

CPI_BTB: 1.4 + 0.04 = 1.44

Problemas de Repaso: Temas 1 y 2

NO SE PIDE EN EXAMEN. ¿Que pasaría con los saltos incondicionales?
Penalización y arranque en frio del salto in condicional. Penaliza **2** ciclos
* en el ciclo D1 se conoce que se trata de un salto y la dirección efectiva y puede actualizar el BTB

Branch	F1	F2	D1[1]	R1	A1[2]	A2	M1	M2	W1							
T		F1	F2	D1	R1	--										
T+1			F1	F2	D1	--										
T+2																
C				F1	F2	D1	R1	A1	A2	M1	M2	W1				
C+1					F1	F2	D1	R1	A1	A2	M1	M2	W1			

B3. Penalización debida de los saltos usando BTB y CPI total teniendo en cuenta los saltos

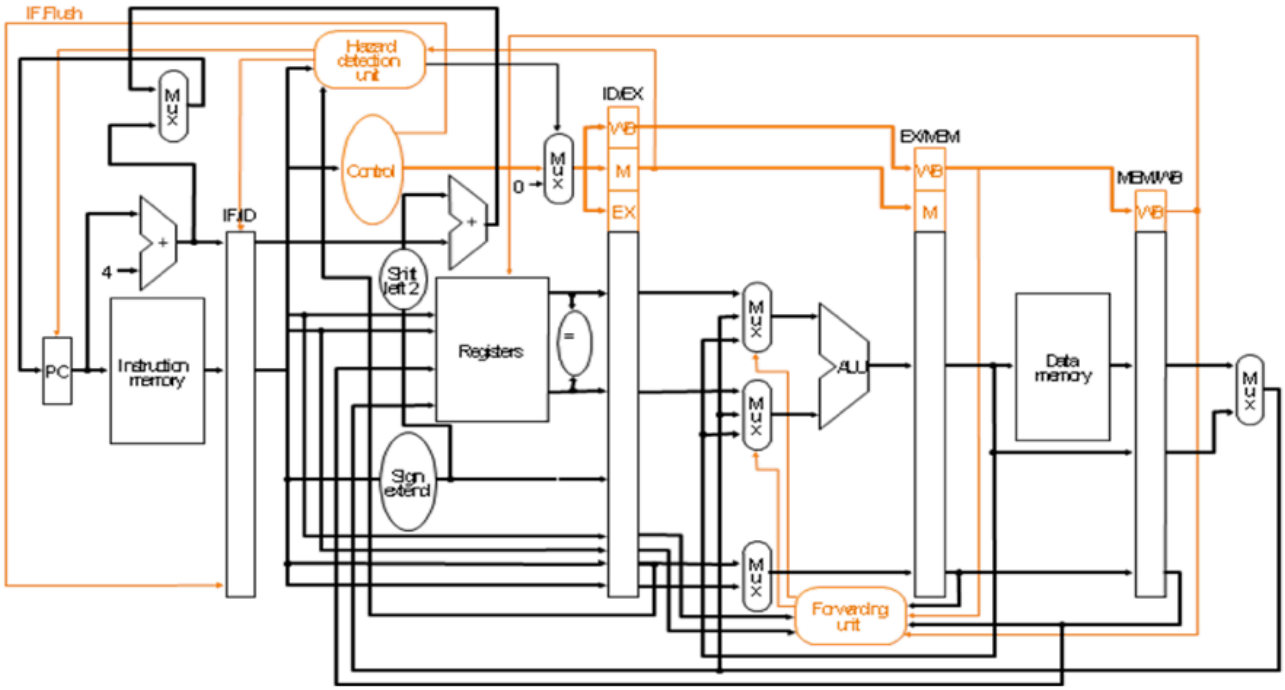
Para saltos condicionales, el BTB acierta el 90% de las veces con 0 ciclos de penalización. Y el 10% penaliza 4 ciclos.
Para los incondicionales el 1% de las veces penaliza 2 ciclos

PenalizSaltosBTB: 0.1 * (0.9*0 + 0.1*4) + 0.02*(0.01*2) = 0.1*0.4 + 0.0004 = 0.04+.0004 = 0.0404

CPI_BTB: 1.4 + 0.0404 = 1.4404

Problemas de Repaso: Temas 1 y 2

P4.- El microprocesador MIPS, estudiado en teoría, posee una unidad de adelantamiento de datos (forwarding unit) que solo permite adelantar los datos de las etapas MEM y WB a la etapa de ejecución (EX):



NOTA: Los registros de las diferentes etapas se llaman PC, IF/ID, ID/EX, EX/MEM y MEM/W respectivamente. El multiplexor de la entrada superior de la ALU se denomina AnticiparA y el de la entrada inferior de la ALU se denomina AnticiparB. En estos multiplexores de tres entradas la entrada superior se corresponde con la selección 00, la del centro con la 01 y la inferior con 10.

La lógica (pseudocódigo) correspondiente a la unidad de adelantamientos es:

Adelantamientos desde la etapa MEM	***Adelantamientos desde la etapa WB***
if (EX/MEM.EscrReg **and** EX/MEM.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd = ID/EX.Reg.Rs) **then** AnticiparA = 10 **else** AnticiparA = 00 **if** (EX/MEM.EscrReg **and** EX/MEM.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd = ID/EX.Reg.Rt) **then** AnticiparB = 10 **else** AnticiparB = 00	**if** (MEM/WB.EscrReg **and** MEM/WB.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd ≠ ID/EX.Reg.Rs **and** MEM/WB.Reg.Rd = ID/EX.Reg.Rs) **then** AnticiparA = 01 **else** AnticiparA = 00 **if** (MEM/WB.EscrReg **and** MEM/WB.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd ≠ ID/EX.Reg.Rt **and** MEM/WB.Reg..Rd = ID/EX.Reg.Rt) **then** AnticiparB = 01 **else** AnticiparB = 00

Tenga en cuenta que, como se representa en la figura, para mejorar las penalizaciones en los saltos se ha cambiado el diseño inicial, introduciendo en la etapa ID, un comparador que permite evaluar la condición de saltos, un sumador para calcular el destino de salto y un multiplexor para actualizar el PC caso de que sea necesario al final de esta etapa.

Problemas de Repaso: Temas 1 y 2

Suponga que se ejecuta la siguiente secuencia de código:

```
I1: Ld  r5, r1, 8       ; r5  <= Mem(r1+8)
I2: Add r1, r1, r3      ; r1 <= r1+r3
I3: Mul  r1, r5, r1     ; r1 <=r5*r1
I4: sub r2, r1, r4      ;r2 <= r1-r4
```

e) Indique los riesgos que presenta la secuencia de código anterior:

RAW : I3 con I1 por R5
RAW : I3 con I2 por R1
RAW : I4 con I3 por R1
No es correcto marcar RAW de I4 con I2 por R1

f) Complete el cronograma indicando los adelantamientos que se hayan podido realizar.

	1	2	3	4	5	6	7	8
LD R5, R1,8	IF	ID	EX	M				
ADD R1,R1,R3		IF	ID	EX				
MUL R1,R5,R1			IF	ID				
SUB R2,R1,R4				IF				

Sol:

	1	2	3	4	5	6	7	8
LD R5, R1,8	IF	ID	EX	M	W			
ADD R1,R1,R3		IF	ID	EX	M	W		
MUL R1,R5,R1			IF	ID	EX	M	W	
SUB R2,R1,R4				IF	ID	EX	M	W

Se puede resolver todos los riegos RAW para que no tengan penalización.

- En el ciclo 4 no se necesitan adelantamientos
- Adelantamiento desde MEM a EX en el ciclo 5 para llevar el valor leído en memoria desde Reg MEM/W a la entrada primera de la ALU para la ejecución de MUL
- Adelantamiento desde EX a EX en el ciclo 5 para llevar el valor que se calcula en la ALU para R1 (ADD) desde el reg EX/MEM a la segunda entrada de la ALU para la ejecución de MUL
- Adelantamiento desde EX a EX en el ciclo 6 para llevar el valor que se calcula en la ALU por (MUL) para R1 desde el reg EX/MEM a la primera entrada de la ALU para la ejecución de SUB.

g) En el ciclo 4 (Instrucción ADD en la etapa EX) indique los valores de

Sol:
ID/EX.Reg.Rs = r1
ID/EX.Reg.Rt = r3
ID/EX.Reg.Rd = r1
EX/MEM.Reg.Rd = r5
MEM/W.Reg.Rd =x
Multiplexor AnticiparA= 00
Multiplexor AnticiparB= 00

h) En el ciclo 5 indique los valores de

Sol:
ID/EX.Reg.Rs = r5
ID/EX.Reg.Rt = r1
ID/EX.Reg.Rd = r1
EX/MEM.Reg.Rd = r1
MEM/W.Reg.Rd =r5
Multiplexor AnticiparA= 01
Multiplexor AnticiparB= 10

i) En el ciclo 6 indique los valores de

Sol:
ID/EX.Reg.Rs = r1
ID/EX.Reg.Rt = r4
ID/EX.Reg.Rd = r2
EX/MEM.Reg.Rd = r1
MEM/W.Reg.Rd =r1
Multiplexor AnticiparA= 10
Multiplexor AnticiparB= 00

Problemas de Repaso: Temas 1 y 2

P5.- A la hora de desarrollar un sistema basado en procesador MIPS segmentado, como el estudiado en clase (5 etapas: IF, ID, EX, M y WB), se están considerando 2 tipos de arquitectura: Von Neumann y Harvard.

Para evaluar ambas arquitecturas se ejecuta el siguiente programa:

```
I1: ld r1, 100(r0)
I2: ld r2, 104(r0)
I3: add r4, r2, r1
I4: ld r3, 108(r0)
I5: add r5, r4, r3
I6: store r5, 112(r0)
```

Se pide completar los primeros 17 ciclos de los cronogramas adjuntos. Considere que las memorias son ideales (acceso en un ciclo), **no hay adelantamiento de datos**, y **no es posible escribir y leer** un mismo **registro en un ciclo** de reloj.

Von Neumann

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I1:	IF*	ID	EX	M*	WB												
I2:		IF*	ID	EX	M*	WB											
I3:			IF*	(ID)	(ID)	(ID)	ID	EX	M	WB							
I4:				*	*	IF*	(IF)	(ID)	EX	M	W						
I5:								IF*	ID	(ID)	(ID)	ID	EX	M	W		
I6:									IF*	(IF)	(IF)	(IF)	ID	ID	ID	ID	EX

Harvard

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I1:	IF	ID	EX	M	WB												
I2:		IF	ID	EX	M	WB											
I3:			IF	(ID)	(ID)	(ID)	ID	EX	M	WB							
I4:				IF	(IF)	(IF)	(IF)	(ID)	EX	M	WB						
I5:								IF	ID	(ID)	(ID)	ID	EX	M	WB		
I6:									IF	(IF)	(IF)	(IF)	ID	(ID)	(ID)	ID	EX

Problemas de Repaso: Temas 1 y 2

P6.- Considerar el siguiente procesador RISC con un CPI de 1,2 ciclos (teniendo en cuenta fallos de caché, TLB y riesgos de datos insalvables por adelantamiento) segmentado en 9 etapas:

- **F1** Captura de instrucción 1. Envía la dirección de la instrucción a la I-caché.
- **F2** Captura de instrucción 2. Lee la instrucción de la I-caché.
- **D1** Decodificación de la instrucción. Calcula la dirección destino en saltos.
- **R1** Lectura de operandos desde registros.
- **A1** Comienza la operación en la ALU.
- **A2** Termina la operación con la ALU. Resuelve la condición de saltos y actualiza BTB
- **M1** Captura de datos 1. Envía la dirección efectiva a la D-caché.
- **M2** Captura de datos 2. Lee/escribe el dato de/en la D-caché.
- **W1** Escribe resultado en un registro.

Un código de benchmarking utilizado posee un porcentaje de saltos condicionales del 10%, de los cuales el 40% son efectivos. No se consideran los saltos incondicionales. Suponiendo que no existen otros riesgos que los de control, se pide:

a) Determinar el CPI real si la predicción es estática i) efectiva o ii) no efectiva. En ambos casos, cuando se detecta la instrucción de salto, **el procesador no se detiene** hasta poder ejecutar la predicción elegida.

b) Determinar el CPI real si se mejora el caso anterior con una estructura BTB (Branch Target Buffer). Donde los saltos condicionales aciertan la predicción del BTB el 90% (el 10% restante corresponde a fallos de predicción y arranque en frio (primera ocurrencia) del salto).

Ejercicio a1. Para Justificar su respuesta, utilice las 4 tablas adjuntas que analizan la penalización en ciclos para los casos de predicción estática efectiva y no efectiva

Predicción estática No efectiva/Salto No efectivo: **0** ciclos de penalización

Branch	F1	F2	D1(1	R1	A1(2	A2	M1	M2	W1							
N		F1	F2	D1	R1	A1	A2	M1	M2	W1						
N+1			F1	F2	D1	R1	A1	A2	M1	M2	W1					
N+2				F1	F2	D1	R1	A1	A2	M1	M2	W1				
N+3					F1	F2	D1	R1	A1	A2	M1	M2	W1			
N+3						F1	F2	D1	R1	A1	A2	M1	M2	W1		

1) Calcula dirección de destino. 2) resuelve condición

Problemas de Repaso: Temas 1 y 2

Predicción estática No efectiva/Salto efectivo: **5** ciclos de penalización

Branch	F1	F2	D1(1	R1	A1	A2(2	M1	M2	W1							
N		F1	F2	D1	R1	A1	--									
N+1			F1	F2	D1	R1	--									
N+2				F1	F2	D1	--									
N+3					F1	F2	--									
						F1	--									
T							F1	F2	D1	R1	A1	A2	M1	M2	W1	
T+1								F1	F2	D1	R1	A1	A2	M1	M2	W1

Predicción estática efectiva/Salto No efectivo: **3** ciclos de penalización

Branch	F1	F2	D1(1	R1	A1	A2(2	M1	M2	W1							
N		F1	F2	D1	R1	A1	A2	M1	M2	W1						
N+1			F1	F2	D1	R1	A1	A2	M1	M2	W1					
N+2							F1	F2	D1	R1	A1	A2	M1	M2	W1	
N+3								F1	F2	D1	R1	A1	A2	M1	M2	W1
T				F1	F2	D1	--									
T+1					F1	F2	--									
						F1	--									

Predicción estática efectiva/Salto efectivo: **2** ciclos de penalización

Branch	F1	F2	D1(1	R1	A1	A2(2	M1	M2	W1							
N		F1	F2	D1	R1	--										
N+1			F1	F2	D1	--										
N+2																
T				F1	F2	D1	R1	A1	A2	M1	M2	W1				
T+1					F1	F2	D1	R1	A1	A2	M1	M2	W1			
T+2						F1	F2	D1	R1	A1	A2	M1	M2	W1		

a.3. Penalización de la estrategia de predicción estática Efectiva y CPI total teniendo en cuenta los saltos

Si salto no efectivo (60%), 3 ciclos de penalización. Si salto efectivo (40%), 2 ciclos de penalización.

PenalizSaltos: 0.1 * (0.6*3 + 0.4*2) = 0.1*2.4 = 0.24

CPI_prefEf: 1.2 + 0.24 = 1.44

a.4. Penalización de la estrategia predicción estática NO Efectiva y CPI total teniendo en cuenta los saltos

Si salto no efectivo (60%), 0 ciclos de penalización. Si salto efectivo (40%), 5 ciclos de penalización.

PenalizSaltos: 0.1 * (0.6*0 + 0.4*5) = 0.1*2.0 = 0.2

CPI_prefNoEf: 1.2 + 0.2 = 1.4

Problemas de Repaso: Temas 1 y 2

Ejercicio b1. Analice la penalización en el fallo y arranque en frio del BTB. Utilice las tablas para justificar la penalización en ciclos.

Penalización y arranque en frio de salto Efectivo en salto condicional. Penaliza **5** ciclos
Ayuda: en A2 se conoce efectivamente la condición y se puede actualizar el BTB

Branch	F1	F2	D1(1	R1	A1	A2(2	M1	M2	W1							
N		F1	F2	D1	R1	A1	--									
N+1			F1	F2	D1	R1	--									
N+2				F1	F2	D1	--									
N+3					F1	F2	--									
N+4						F1	--									
T							F1	F2	D1	R1	A1	A2	M1	M2	W1	
T+1								F1	F2	D1	R1	A1	A2	M1	M2	W1

Penalización y arranque en frio de salto no Efectivo en salto condicional. Penaliza **0** ciclos
Ayuda: en A2 se conoce efectivamente la condición y se puede actualizar el BTB

Branch	F1	F2	D1(1	R1	A1	A2(2	M1	M2	W1							
N		F1	F2	D1	R1	A1	A2	M1	M2	W1						
N+1			F1	F2	D1	R1	A1	A2	M1	M2	W1					
N+2				F1	F2	D1	R1	A1	A2	M1	M2	W1				
N+3					F1	F2	D1	R1	A1	A2	M1	M2	W1			
N+4						F1	F2	D1	R1	A1	A2	M1	M2	W1		

b2. Penalización debida de los saltos usando BTB y CPI total teniendo en cuenta los saltos

Para saltos condicionales, el BTB acierta el 90% de las veces con 0 ciclos de penalización. Y el 10% penaliza 5 ciclos solo en los casos de salto efectivos (Los no efectivos al mantener la inercia del pipeline no penalizan ni siquiera en el arranque en frio)

PenalizSaltosBTB: 0.1 * (0.9*0 + 0.1*(0,6*0+0,4*5) = 0.1*0.2 = 0.02

CPI_BTB: 1.2 + 0.02= 1.22