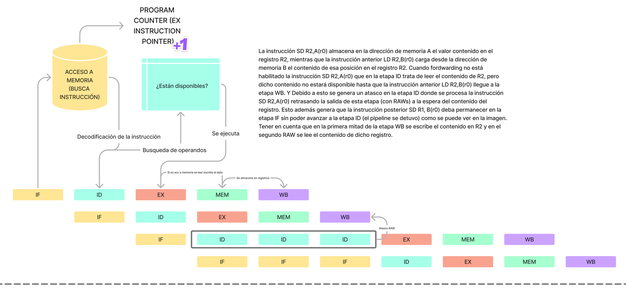
TP4



La instrucción SD R2,A(r0) almacena en la dirección de memoria A el valor contenido en el registro R2, mientras que la instrucción anterior LD R2,B(r0) carga desde la dirección de memoria B el contenido de esa posición en el registro R2. Cuando forwarding no está habilitado la instrucción SD R2,A(r0) que en la etapa ID trata de leer el contenido de R2, pero dicho contenido no estará disponible hasta que la instrucción anterior LD R2,B(r0) llegue a la etapa WB. Y Debido a esto se genera un atasco en la etapa ID donde se procesa la instrucción SD R2,A(r0) retrasando la salida de esta etapa (con RAWs) a la espera del contenido del registro. Esto además genera que la instrucción posterior SD R1, B(r0) deba permanecer en la etapa IF sin poder avanzar a la etapa ID (el pipeline se detuvo) como se puede ver en la imagen. Tener en cuenta que en la primera mitad de la etapa WB se escribe el contenido en R2 y en el segundo RAW se lee el contenido de dicho registro.

a) Se presentan atasco por dependencia de datos de tipo RAW causado por la instrucción BNEZ R2, loop al
procesarse en la etapa ID. Esta instrucción necesita del contenido del registro R2 que está siendo utilizado por la
instrucción DADDI R2,R2,-1 en la etapa EX sin salir aún de esta.
El atasco de tipo Branch Taken Stalls (BTS), ocurre como consecuencia de la ejecución incorrecta de la instrucción siguiente a una instrucción condicional. Esto se debe a que la condición a evaluar tarda algunos ciclos en ser ejecutada, mientras que durante esos ciclos siguen entrando nuevas instrucciones al pipeline. Luego de evaluada la condición si la instrucción posterior a ésta que se ejecutó no es la que debía ser ejecutada, su ejecución se trunca y se ejecuta la que está en el lugar de memoria indicada por la etiqueta en la instrucción condicional.
Se ejecutan 12 instrucciones en 21 ciclos dando un CPI de 1.750

b) Las instrucciones que generan los atascos RAW son: la instrucción DSLL R1,R1, que trata de leer el contenido del registro R1, mientras que la instrucción LD R1,A(r0) todavía no copio el contenido de la dirección de memoria A en R1 y permanece aún en la etapa WB (RAW durante 1 ciclo). Y la instrucción BNEZ R2,loop que trata de leer el contenido del registro R2, mientras que DADDI R2,R2,-1 está buscando copiar el resultado de la operación en dicho registro, permaneciendo en la etapa MEM y posteriormente en la etapa WB (RAW durante 2 ciclos).
Con forwarding deshabilitado, los atascos por Branch Taken Stalls duran 2 ciclos en cada vuelta del lazo loop, mientras que con dicha opción habilitada se reducen a 1 ciclo por vuelta de lazo. Esta diferencia tiene su causa en la instrucción condicional que es la que está generando los atascos RAW; entonces al disminuir la cantidad de RAWs producidos por esta, también disminuyen los ciclos de espera de la instrucción siguiente, que además se dejara de ejecutar si la condicional así se lo indica al procesador.

2.083 CPI 12 instrucciones 25 ciclos (sin forwarding) vs 12 instrucciones en 21 ciclos (con forwarding)
c)

```
    .data
A: .word 1
B: .word 3
    .code
ld r2, B(r0)
ld r1, A(r0)
loop: daddi r2, r2, -1
dsll r1, r1, 1
bnez r2, loop
halt
```

El programa busca en TABLA un elemento igual al contenido en la dirección de memoria NUM. En este caso dicha coincidencia se produce cuando el contenido del registro R4 es igual al contenido del registro R2 (R4=R2), razón por la cual luego de evaluada esta condición y de resultar verdadera se salta a la posición de memoria indicada por la etiqueta “listo”. Cuando hay coincidencia la línea de programa en listo suma al registro R10 un 1, caso contrario el contenido del registro R10 queda en 0. Este es el resultado y queda almacenado en el registro R10. El registro R3 se utiliza como índice para recorrer la TABLA. El contenido del registro R3 se incrementa de a 8 porque cada elemento de tabla es del tamaño word, es decir de 64 bits (8 bytes).
Habilitando la opción Branch Target Buffer (BTB) logramos reducir los atascos Branch Taken stalls a la mitad. Tener en cuenta que esta opción es útil cuando aumenta la cantidad de iteraciones de un lazo. Como vemos también esta opción no actúa sobre los atascos por dependencia de datos (RAW en este caso) que no se modifican.

	CON FORWARDING	CON BTB
CICLOS	71	75
CPI	1.651	1.744
RAWs	16	24
BTS	8	4

El Delay Slot consiste en ejecutar siempre la siguiente instrucción a un salto. El NOP se utiliza para brindarle más tiempo a la siguiente instrucción

88 ciclos
59 instrucciones
1.492 CPI
vs
61 ciclos
57 instrucciones
1.070 CPI

TP5

1 b) 4 atascos RAW: f2, f3 y f4 los registros
c) Los atascos estructurales son provocados por conflictos por los recursos. En el contexto del MIPS sucede cuando DOS instrucciones intentan acceder a la etapa MEM simultáneamente.
d) El atasco WAR se produce porque hay una instrucción de escritura (mul.d en este caso) que intenta escribir en un registro de destino antes de que se complete una instrucción de lectura previa (l.d en este caso) que utiliza ese mismo registro como fuente: mul.d f1, f2, f1 depende de los valores de f2 y f1, que se están leyendo en las instrucciones l.d anteriores.
e) Cuando se coloca un NOP (instrucción de no operación) antes de la instrucción add.d, se introduce un ciclo adicional de instrucción, lo que permite que la instrucción l.d anterior se complete antes de que se ejecute la instrucción add.d. Esto resuelve el peligro de dependencia RAW (Read-After-Write) entre la lectura de f1 y la escritura de f3. La introducción del NOP permite que la instrucción l.d tenga tiempo suficiente para escribir en el registro f1 antes de que se lea en la instrucción add.d.
Al resolver el peligro de dependencia RAW, también se crea un espacio temporal que evita el peligro de dependencia WAR

Instrucción	Descripción
mtc1 r1, f1	Copia los 64 bits del registro entero r1 al registro f1 de punto
mfc1 r1, f1	Copia los 64 bits del registro f1 de punto flotante al registro r1 entero
cvt.d.l f1, f2	Convierte a punto flotante el valor entero copiado al registro f2, dejándolo en f1
cvt.l.d f1, f2	Convierte a entero el valor en punto flotante contenido en f2, dejándolo en f1