Pregunta	si	no
1. El siguiente fragmento de código produce 1 ciclo de atasco cuando se ejecuta en un cauce segmentado de 5 etapas (IF-ID-EX-MEM-WB) sin caminos de adelantamiento. ¿Es eso cierto?	☐	●

```
add $s3, $s2, $s1
sw  $s3, 100($s0)
```

no, pues no existe ninguna dependencia RAW entre las instrucciones. Se trata de una falsa dependencia (Write after Write – WAW) que es automáticamente resuelta en cauces escalares segmentados.

2. Un procesador tiene un cauce segmentado de 5 etapas, un CPI de 3.2 y funciona a una frecuencia de 2.8 GHz. ¿Es cierto que el rendimiento de dicho procesador es de 875 MIPS?	●	☐

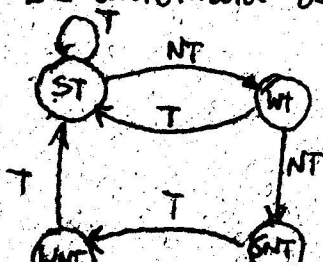
$$MIPS = \frac{f_{ck}}{CPI \cdot 10^6} = \frac{2{,}8 \cdot 10^9}{3{,}2 \cdot 10^6} = 875 \text{ MIPS}$$

3. Un procesador dispone de un sistema de memoria virtual mixto formado por una tabla de páginas y un TLB. El tiempo de acceso a la memoria es de 70 ns, mientras que el TLB tiene un tiempo de acceso de 5 ns y una tasa de fallo del 10%. El tiempo medio para realizar una traducción de dirección es de 12 ns. ¿Es eso cierto?	●	☐

$$t_{traducción} = t_{hit\text{-}TLB} + p_{miss\text{-}TLB} \cdot t_{MEM} = 5\,ns + 0{,}1 \cdot 70\,ns = 12\,ns$$

4. Una instrucción de salto tiene la siguiente secuencia de "tomado" (T) y "no tomado" (N): T T T N N N T T T N N N. Si el procesador dispone de un predictor de saltos de 2 bits cuyo estado inicial es el "tomado fuerte", ¿es cierto que, por la secuencia anterior de resultados del salto, el predictor en cuestión tiene una tasa de acierto del 70%?	☐	●

El autómata del predictor es:



NT = not taken
T = Taken

ST = Strong Taken
WT = Weak Taken
SNT = Strong Not Taken

Estado	ST	ST	ST	ST	WT	SNT	SNT	WNT	ST	ST
Predicción	T	T	T	T	T	N	N	N	T	T
Resultado	T	T	T	N	N	N	T	T	T	N

WT SNT

el número representado es negativo (C=1) entonces no puede ser 21.8675

PREGUNTA 23. Un hipotético procesador monociclo tiene un espacio de direccionamiento virtual de 4096 palabras, un tamaño de página de 128 palabras y utiliza una política de remplazo de tipo LRU. La memoria principal puede contener 4 páginas y el sistema no admite solapamientos de páginas. Inicialmente la memoria principal está vacía y la CPU empieza a ejecutar una secuencia ordenada de 10 instrucciones. La dirección de palabra de cada instrucción y la dirección de palabra de cada dato se muestran (en representación decimal) en la Tabla 1. El símbolo "-" denota que la instrucción en cuestión no accede a memoria. Complete la Tabla 1 indicando para todos los accesos a memoria si producen un fallo o un acierto. Calcule la tasa de acierto del sistema e indique también en qué orden se cargan las páginas en la memoria principal y cuáles páginas son remplazadas.

Tabla 1:

#	Dirección instrucción		Dirección operando		Instr. (hit/miss)	Dato (hit/miss)
1	PÁG. #0	15	PÁG. 2	270	miss	miss
2	PÁG. #0	16	PÁG. 2	271	hit	hit
3	PÁG. #0	17	—	-	hit	—
4	PÁG. #1	130	PÁG. 3	402	miss	miss
5	PÁG. #1	131	PÁG. 3	405	hit	hit
6	PÁG. #1	132	PÁG. 4	525	hit	miss
7	PÁG. #1	133	—	-	hit	—
8	PÁG. #0	39	PÁG. 4	518	hit	hit
9	PÁG. #0	40	PÁG. 6	769	hit	miss
10	PÁG. #0	41	PÁG. 3	470	hit	miss

aquí se lleva la memoria

ahora la pág. más antigua es la #3

remplaza #3

remplaza #1

hay que cargar la página #4 remplazando la más antigua (página #2 a la que se accedió en el ciclo 2)

Cada página contiene 128 palabras por lo que el mapeo entre direcciones físicas y número de página es

Dirección de memoria	# página
0 – 127	0
128 – 255	1
256 – 383	2
384 – 511	3
512 – 639	4
640 – 767	5
768 –	6

Actualizo la tabla (1) anotando el número de página al lado de cada dirección física y observo que el programa accede a las páginas 0, 2, 1, 3, 4 y 6

En memoria sólo residen 4 páginas por lo que se remplazará una página sólo cuando la memoria esté ll...

1 → 3 remplaza 1

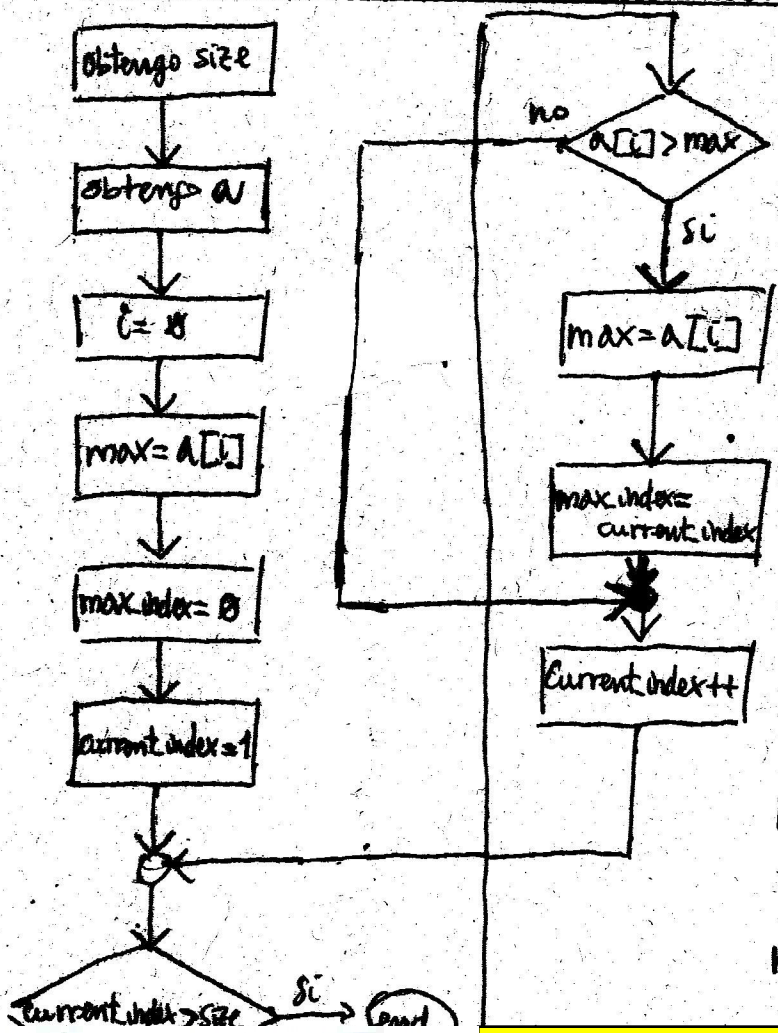
3 → 6 remplaza 3

tasa de acierto = 11 aciertos / 18 accesos = 0.61

PREGUNTA 2.2. Considere el siguiente programa en Java:

```
int max = a[0];
for (i = 1; i < a.size; i++){
  if (a[i] > max){
    max = a[i];
  }
}
```

el programa calcula el valor más grande en un array de enteros. Dibuje el diagrama de flujo relativo al código anterior y a continuación traduzca el programa Java en un programa en lenguaje ensamblador MIPS. Finalmente, modifique el programa ensamblador añadiendo exactamente dos instrucciones para determinar el índice del array relativo al valor máximo. Utilice el registro $s3 para almacenar el índice e indique con una flecha las posiciones en el código en la que hay que insertar las dos nuevas instrucciones.

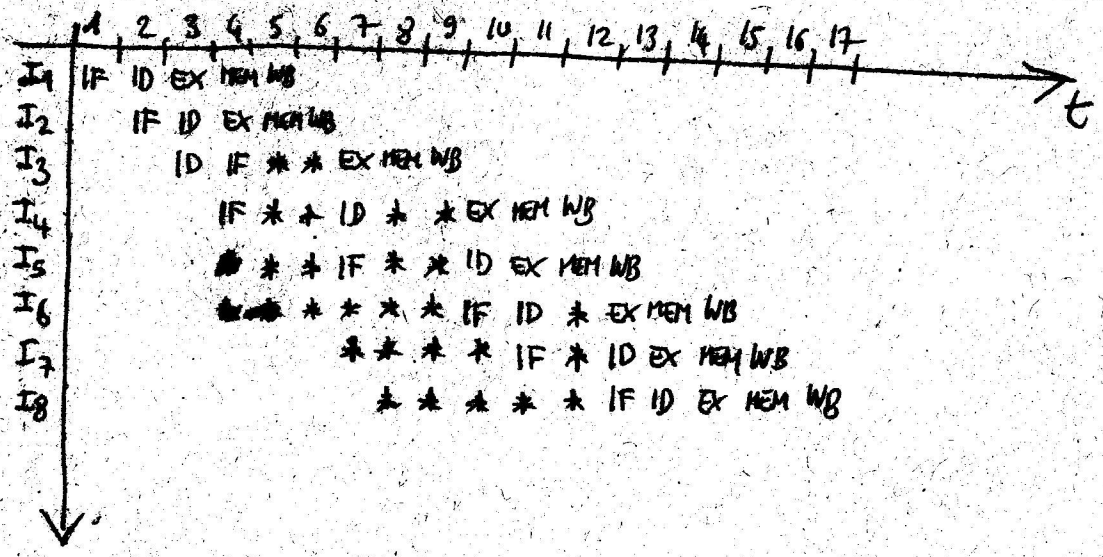



```
main:
    la   $t1, size
    lw   $s0, 0($t1)      # $s0 = size
    la   $t1, a           # $t1 = a[0]
    lw   $s1, 0($t1)      # $s1 = max = a[0]
    addi $s2, $zero, 1    # $s2 = current_index = 1
    addi $s3, $zero, 0    # $s3 = max_index = 0
loop:
    bge  $s2, $s0, endloop
    addi $t1, $t1, 4      # $t1 = i++
    lw   $t2, 0($t1)      # $t2 = a[i]
    bgt  $t2, $s1, newmax
    j nomax
newmax:
    move $s1, $t2         # $s1 = max
    move $s3, $s2         # max_index = current_index
nomax:
    addi $s2, $s2, 1      # current_index++
```

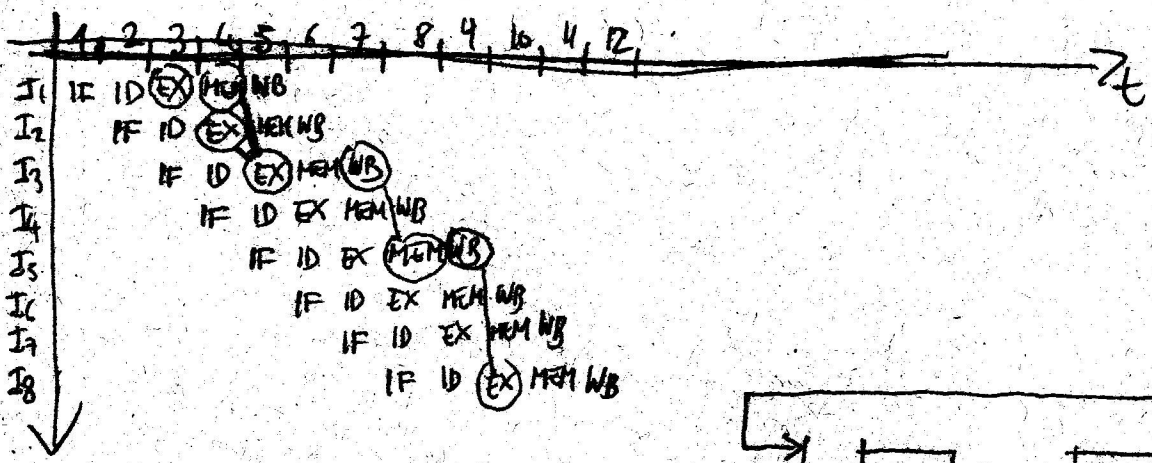
PREGUNTA 2.3 (2 puntos). El siguiente fragmento de código se ejecuta en un procesador MIPS segmentado de 5 etapas (IF, ID, EX, MEM, WB):

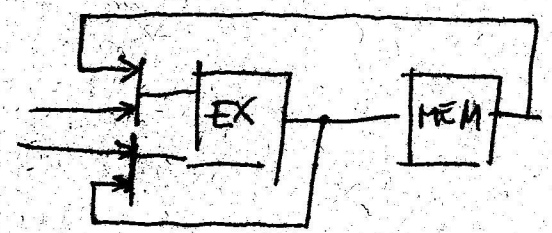
```
I1 lw  $s1, 0($s0)
I2 lw  $s2, 4($s0)
I3 add $s3, $s1, $s2
I4 add $s4, $s1, $s3
I5 lw  $s5, 8($s0)
I6 add $s6, $s3, $s4
I7 sw  $s6, 12($s0)
I8 beq $s4, $s5, label
...
label:
...
```

1. ¿Cuántos ciclos se necesitarán para ejecutar el código anterior teniendo en cuenta únicamente los riesgos generados por dependencias de datos y resolviéndolos mediante atascos del cauce?
2. Si para resolver los riesgos utilizamos la técnica de adelantamiento de datos (bypass), ¿cuántos ciclos se necesitarían para ejecutar el código anterior?
3. ¿Cuál es la mejora del rendimiento (speedup) que se obtiene por introducir bypass en el cauce? Dibujar, para cada uno de los casos anteriores, el diagrama de tiempos con la evolución de la ejecución del fragmento de código a lo largo del cauce segmentado.



Sin bypass y produciendo atascos el fragmento de código tarda 17 ciclos





Con bypass la ejecución tarda 12 ciclos y se cumplen todas las dependencias de datos

$$Speedup = \frac{17}{12} \approx 1.41$$

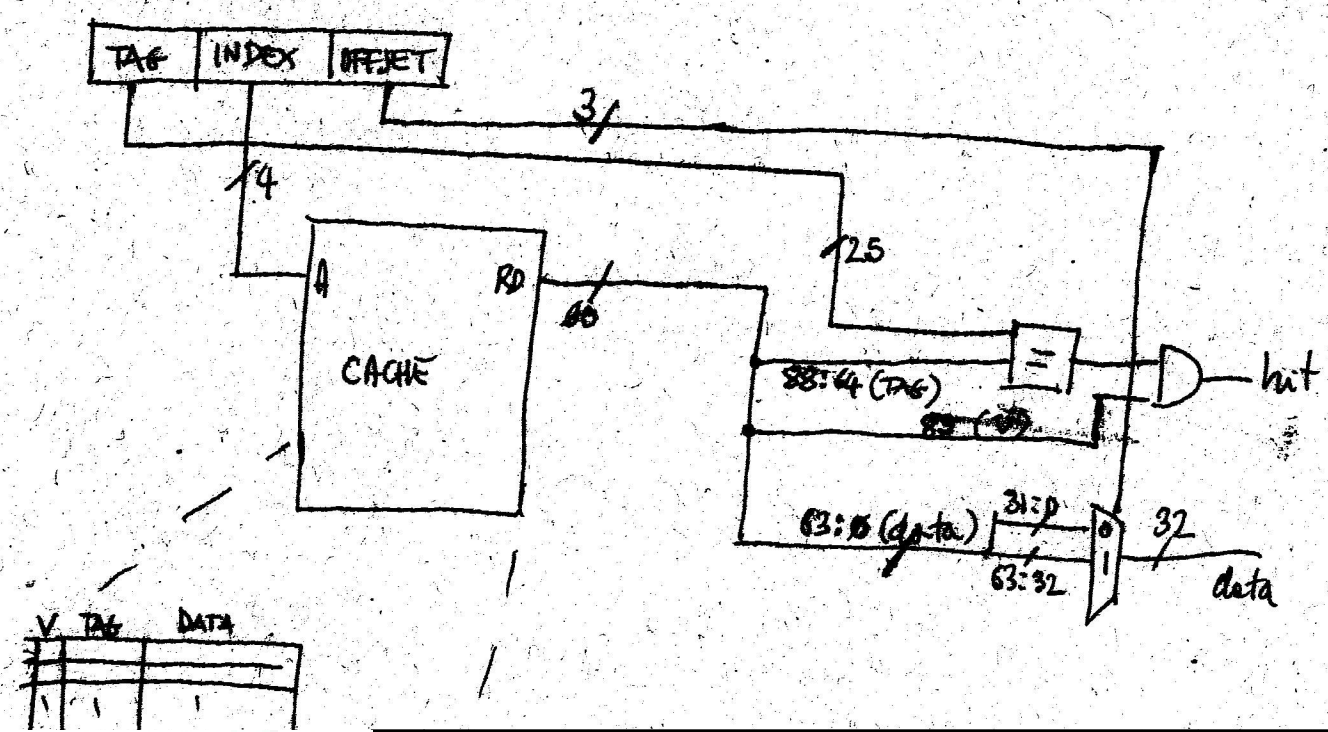
PROBLEMA 3.1. Asuma un procesador MIPS de 32-bits con una cache de instrucciones de mapeo directo. Asuma que la cache tenga 16 filas y que el tamaño de un bloque es de 8 bytes.

1. Dibuje la implementación hardware del circuito necesario para acceder a la cache en lectura.
2. Explique cuál es el significado de los campos de etiqueta, de índice y de offset y calcule su tamaño para la cache en cuestión.
3. Explique, paso a paso qué es lo que pasa y cómo se actualiza el contenido de la cache al ejecutarse el programa que se muestra a continuación:

```
        addi $t0,$0,0
        addi $t1,$0,0
loop:   add  $t1,$t1,$t0
        addi $t0,$t0,1
        slti $t2,$t0,10
        bne  $t2,$0, loop
```

Asuma que la primera instrucción se halle a la dirección **0x00400000** y que la cache esté inicialmente vacía. Finalmente, calcule la tasa de acierto de la memoria cache cuando se ejecuta el programa anterior.

(1)



(2) Ver teoría

(3) Una fila contiene 8 bytes (dos instrucciones).

Al principio PC = 0x400000 y la caché está vacía por lo que hay un miss y se cargan dos instrucciones

PC = 0x400004 => hit

PC = 0x400008 => miss y se cargan dos instrucciones

PC = 0x40000C => hit

PC = 0x400010 => miss y se cargan dos instrucciones.

PC = 0x400014 => hit

A continuación empieza un bucle de 9 iteraciones entre las direcciones 0x400008 y 0x400014. Todas las instrucciones se hallan en la caché por lo que habrá hit durante las 9 iteraciones siguientes.

La secuencia de hit (H) y miss (M) es:

M, H, M, H, M, H, H x 36 veces.

número accesos = 42 número aciertos (hit) = 39

por tanto

$$\text{hit rate} = \frac{\#\text{hit}}{\#\text{accesos}} = \frac{39}{42} = 0{,}928$$