Pregunta	si	no
1. Considere un formato IEEE 754 normalizado sobre 8 bits con un bit de signo, 4 bits de exponente (codificado en exceso 7) y 3 bits de mantisa. ¿Es cierto que **0 0110 101** representa el número **0?0011011$_2$**? (redondee, si es posible, el resultado al número par más cercano)	☐	■

GRS G= guard bit, R= round bit, S= sticky bit

$0.0011011 = 1.101100 \times 2^{-3}$

GRS = 100 es el caso límite en el que podemos decidir si redondear o no el resultado
El exponente va convertido en exceso 7: -3+7= 4 = 0100_2 por tanto:
0.0011011 = 0 0100 101

2. Asuma el formato IEEE 754 de la pregunta anterior, ¿es cierto que **0 1010 000** representa **16.0$_{10}$**?	☐	■

$0\ 1010\ 000 = +\ 1.000 \times 2^{(10-7)} = 1 \times 2^3 = 8$

3. En una máquina de tipo *little-endian* la siguiente palabra de 32 bits se halla a partir de la dirección **1000**: **`CE 12 AB 91`**. Si este valor representa un número entero natural, ¿es cierto que su valor decimal sería 2.427.130.573?	☐	■

`CE 12 AB 91 (Little endian) = 91 AB 12 CE = 2.443.907.790`

4. A=**`11010011`** y B=**`00111110`** representan dos números binarios con signo representados en complemento a 2. ¿Es cierto que el $\|A\| > \|B\|$?	☐	■
5. El numero binario natural **`10 1111 0111 1010`** equivale a **`2F7A`** en hexadecimal. ¿Es eso cierto?	■	☐

2 F 7 A
10 1111 0111 1010

PREGUNTA 2.1. Asuma un computador que ejecuta un programa formado por un 50% de multiplicaciones de punto flotante, un 20% de divisiones de punto flotante y un 30% de otras instrucciones. Su jefe de proyecto pretende optimizar el computador de manera que pueda ser 1.4 veces más rápido. Ud. es capaz de mejorar la división de un factor 3 y la multiplicación de un factor 8. ¿Puede cumplir con las especificaciones de su jefe realizando sólo una de las dos mejoras? ¿Cuál de las dos?
Si pudiese implementar ambas mejoras del procesador, ¿cuál sería la mejora total del procesador (speedup) respecto a la máquina original?

Sea T_a el tiempo de ejecución y T_m el tiempo de ejecución mejorado
$T_m = (0.5/8)T_a + 0.2T_a + 0.3T_a$ → la mejora de la multiplicación lleva a $T_m = (0.5/8 + 0.5)\ T_a = 0.5625\ T_a$
Speedup = $T_a/T_m = 1/0.5625 = 1.78$ (cumplo con las especificaciones que exigen un speedup de 1.4)

$T_m = 0.5T_a + (0.2/3)T_a + 0.3T_a$ → la mejora de la división lleva a $T_m = (0.2/3 + 0.8)T_a = 0.867T_a$ Speedup = $T_a/T_m = 1/0.867 = 1.15$ (no cumplo con las especificaciones)

Mejora total: $T_m = (0.5/8)T_a + (0.2/3)T_a + 0.3T_a = 0.429\ T_a$ → Speedup = $T_a/T_m = 1/0.429 = 2.33$

PREGUNTA 2.2. Escriba el código ensamblador MIPS equivalente al siguiente fragmento de programa en lenguaje C:

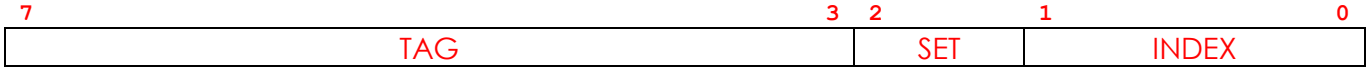
```
int fact (int n) {
  if (n < 1) return;
  else return n * fact(n - 1);
}
```

Asuma que el argumento `n` se halle en el registro `$a0` y que el resultado se guarde en el registro `$v0`.

Solución en la transparencia 44 del capítulo 2

PREGUNTA 2.3. Asuma que un procesador genera la secuencia de direcciones de 8 bits que se muestra en la tabla siguiente. La cache del procesador utiliza bloques de 4 bytes, es asociativa por conjuntos de 2 vías, tiene una capacidad de 4 bloques y utiliza una política de remplazo de tipo LRU.
Asuma la cache inicialmente vacía y determine los campos TAG, SET, INDEX y rellene la tabla.

Cache de 2 vías → 2 conjuntos → basta 1 bit para direccionar los conjuntos.
La cache tiene bloques de 4 bytes → hacen falta $\log_2 4$=2 bits de offset (campo index)
El número de bits del campo de etiqueta es 8-1-2=5
Formato dirección:



tiempo	0	1	2	3	4	5	6	7
dirección	**10001101**	**10110010**	**10111111**	**10001100**	**10011100**	**11101001**	**11111110**	**11101001**
TAG	**10001**	**10110**	**10111**	**10001**	**10011**	**11101**	**11111**	**11101**
SET	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**
INDEX	**01**	**10**	**11**	**00**	**00**	**01**	**10**	**01**

En la Figura siguiente marque para cada acceso el TAG, el bit LRU y el bit de HIT.

LRU bit = 1 si el bloque es el menos utilizado recientemente, Hit bit = 1 si hay cache hit

	Access 0			Access 1	
	Block 0	Block 1		Block 0	Block 1
Set 0			**Set 0**	**10110,0,0**	
Set 1	**10001,0,0**		**Set 1**	**10001,0,0**	

	Access 2			Access 3	
	Block 0	Block 1		Block 0	Block 1
Set 0	**10110,0,0**		**Set 0**	**10110,0,0**	
Set 1	**10001,1,0**	**10111,0,0**	**Set 1**	**10001,0,1**	**10111,1,0**

	Access 4			Access 5	
	Block 0	Block 1		Block 0	Block 1
Set 0	**10110,0,0**		**Set 0**	**10110,1,0**	**11101,0,0**
Set 1	**10001,1,0**	**10011,0,0**	**Set 1**	**10001,1,0**	**10011,0,0**

	Access 6			Access 7	
	Block 0	Block 1		Block 0	Block 1
Set 0	**10110,1,0**	**11101,0,0**	**Set 0**	**11101,0,0**	**11101,1,0**
Set 1	**11111,0,0**	**10011,1,0**	**Set 1**	**11111,0,0**	**10011,1,0**

PROBLEMA 3.1 (3 puntos). Considere la ruta de datos (data path) que se muestra en la Figura 1.

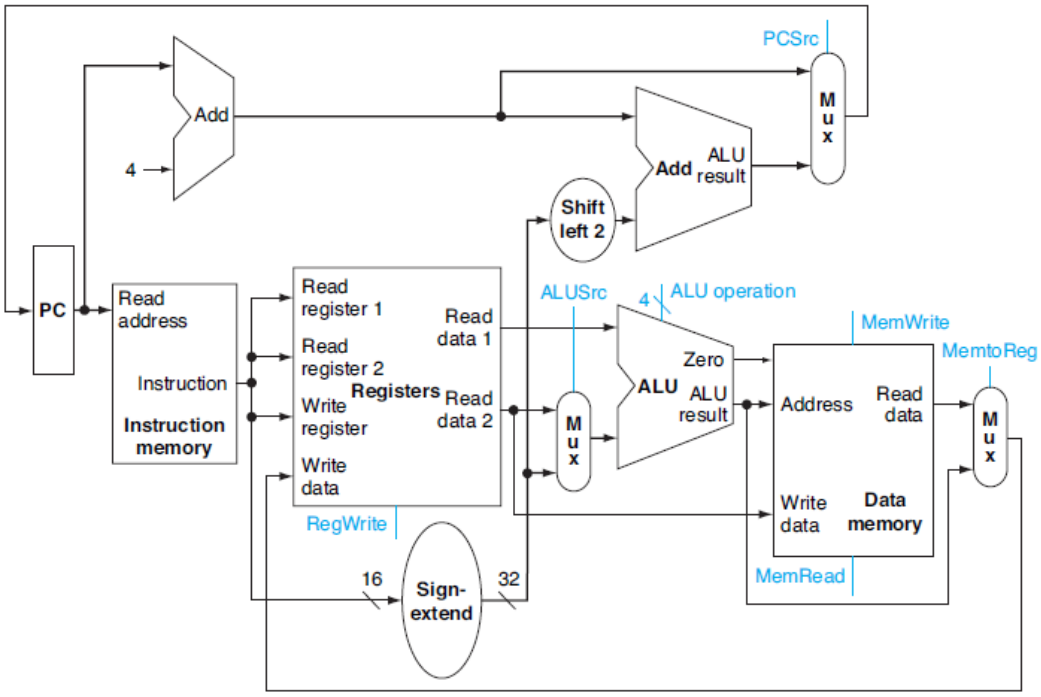


Figura 1. Ruta de datos del Problema 3.

Asuma las latencias siguientes para los bloques lógicos que forman la ruta de datos:

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-ext	Shift-left 2
200 ps	70 ps	20 ps	90 ps	90 ps	250 ps	15 ps	10 ps

¿Cuál es el tiempo de ciclo si hay que soportar instrucciones aritméticas, saltos condicionales e instrucciones de carga y almacenamiento?

T_{ALU}=Imem + Regs + Mux + ALU + Mux = 200 + 90 + 20 +90 +20 = 420 ps
$T_{L/S}$ = Imem + Regs + Mux + ALU + Dmem + Mux = 200 + 90 + 20 + 90 + 250 +20 = 670 ps
T_{Branch}= Imem + Sign-ext +Shift-left-2 + ALu + Mux + Regs = 200 + 15 +10 +90 +20 + 90 = 425 ps

Sigue que:

T_{cycle} = max { T_{ALU}, $T_{L/S}$, T_{Branch} } = $T_{L/S}$ = 670 ps

Ahora asuma que no existen atascos del cauce (*pipeline stalls*) y que las instrucciones se distribuyen como sigue:

Add	Addi	Not	Beq	Lw	Sw
20%	20%	0%	25%	25%	10%

1. Qué fracción del tiempo de ejecución utiliza la memoria de datos.

La memoria de datos es utilizada por Lw y Sw, por tanto, la fracción de tiempo es: 25% + 10% = 35%

2. Si pudiéramos mejorar en un 10% la latencia de uno de los componentes de la ruta de datos, ¿qué componente debería ser y cuál sería la mejora (*speedup*) producida por este cambio?

El tiempo de ciclo es determinado por la instrucción más lenta (Load). Entre los bloques del camino crítico, Dmem es el que tarda más, por consiguiente, reduciendo un 10% la latencia de Dmem (Dmem = 250 x (10/100) = 225 ps) el tiempo de ciclo se vuelve t_{cycle} = 645 ps → Speedup = 670/645 = 1.039