NOMBRE: G1 DNI:

1) Dados los números A=0xBA1D y B=0x6B17 representados en hexadecimal, calcular el valor de la suma A+B supuesto que tanto ellos como su suma se expresan ocupando 16 bits en binario puro, y que la suma se recoge en un registro de 16 bits que siempre recibe los 16 bits menos significativos del resultado. ¿Hay desbordamiento?¿Los 16 bits de resultado expresan el valor correcto de la suma? Justifique todas sus respuestas (0,5 puntos)

$$\begin{array}{r} BA1D \\ + \quad 6B17 \\ \hline 12534 \end{array}$$

→ Valor que recoge el registro destino.

El valor que recoge el registro destino es menor que los sumandos, por tanto hay desbordamiento

Como hay desbordamiento, los 16 bits del resultado no expresan el valor correcto de la suma

Resultado en hexadecimal: 0x 2534

¿Hay desbordamiento? Rodee con un círculo la respuesta correcta: (SÍ) NO

¿Los 16 bits del resultado expresan el valor correcto de la suma? Rodee con un círculo la respuesta correcta: SÍ (NO)

2) Dados los mismos números A y B del ejercicio 1, calcular el valor de la suma A+B supuesto que representan números expresados en complemento a 2 con 16 bits y su suma ha de expresarse de esa misma forma, siendo recogida en un registro que siempre recibe los 16 bits menos significativos del resultado. ¿Hay desbordamiento?¿Los 16 bits de resultado expresan el valor correcto de la suma?(0,5 puntos)

El valor que recoge el registro destino es el mismo pues al realizar la suma en binario da igual que los números sean interpretados como valores en binario puro o complemento a 2.

Como A es negativo y B es positivo, no puede haber desbordamiento.

Como no hay desbordamiento, los 16 bits del resultado expresan el valor correcto de la suma.

Resultado en hexadecimal: 0x 2534

¿Hay desbordamiento? Rodee con un círculo la respuesta correcta: SÍ (NO)

¿Los 16 bits del resultado expresan el valor correcto de la suma? Rodee con un círculo la respuesta correcta: (SÍ) NO

3) Dados los números A=2,75 y B=-3,50x10^2 , expresados en base 10, obtenga la representación de ambos según el formato de punto flotante IEEE-754 de simple precisión y exprésela en hexadecimal.(1 punto)

A = 2,75

→ Positivo ⇒ Bit 31 = "0"

→ $2{,}75_{10} = 10{,}011_2 = 1{,}011 \times 2^1$

→ Exponente = 1 + 127 = 128_{10} = 1000 0000$_2$

Fracción = 0110 0000 0000 0000 0000 000

S	Exponente	Fracción
0	1000 0000	0110 0000 0000 0000 0000 000

4	0	3	0	0	0	0	0

B = $-3{,}50 \times 10^2$

→ Negativo ⇒ Bit 31 = "1"

→ $3{,}50 \times 10^2 = 350 = 1010\,1111\,0_2 = 1{,}0101\,1110 \times 2^8$

→ Exponente = 8 + 127 = 135_{10} = 1000 0111$_2$

Fracción = 0101 1110 0000 0000 0000 000

S	Exponente	Fracción
1	1000 0111	0101 1110 0000 0000 0000 000

C	3	A	F	0	0	0	0

A = 0x 40 30 00 00

B = 0x C3 AF 00 00

4) A partir de las representaciones binarias del ejercicio anterior, calcule la suma de A y B y exprese el resultado según el formato de punto flotante IEEE-754 de DOBLE precisión en hexadecimal. ¿Se produce "overflow"?¿Se realiza algún cambio en la fase de redondeo?(1 punto)

A = $1{,}011 \times 2^1$; B = $-1{,}0101\,1110 \times 2^8$

A+B = A − (−B) = − [(−B) − A]

Alineamos mantisas y restamos:

$$\begin{array}{rrl} (-B) & = & 1{,}0101\,1110\,000 \times 2^8 \\ -\ A & = & 0{,}0000\,0010\,11 \times 2^8 \\ \hline (-B)-A & = & 1{,}0101\,1011\,01 \times 2^8 \end{array}$$

Por tanto: A+B = − [(−B) − A] = $-1{,}0101\,1011^{01}_{\ 1} \times 2^8$

Exponente = 8 + 1023 = 1031_{10} = 100 0000 0111$_2$

Fracción = 0101 1011 0100 y 10 grupos más de "0000"

S	Exponente	Fracción
1	100 0000 0111	0101 1011 0100 0000 0000 ... 0000

C	0	7	5	B	4	0	...	0

A+B = 0x C075 B400 0000 0000

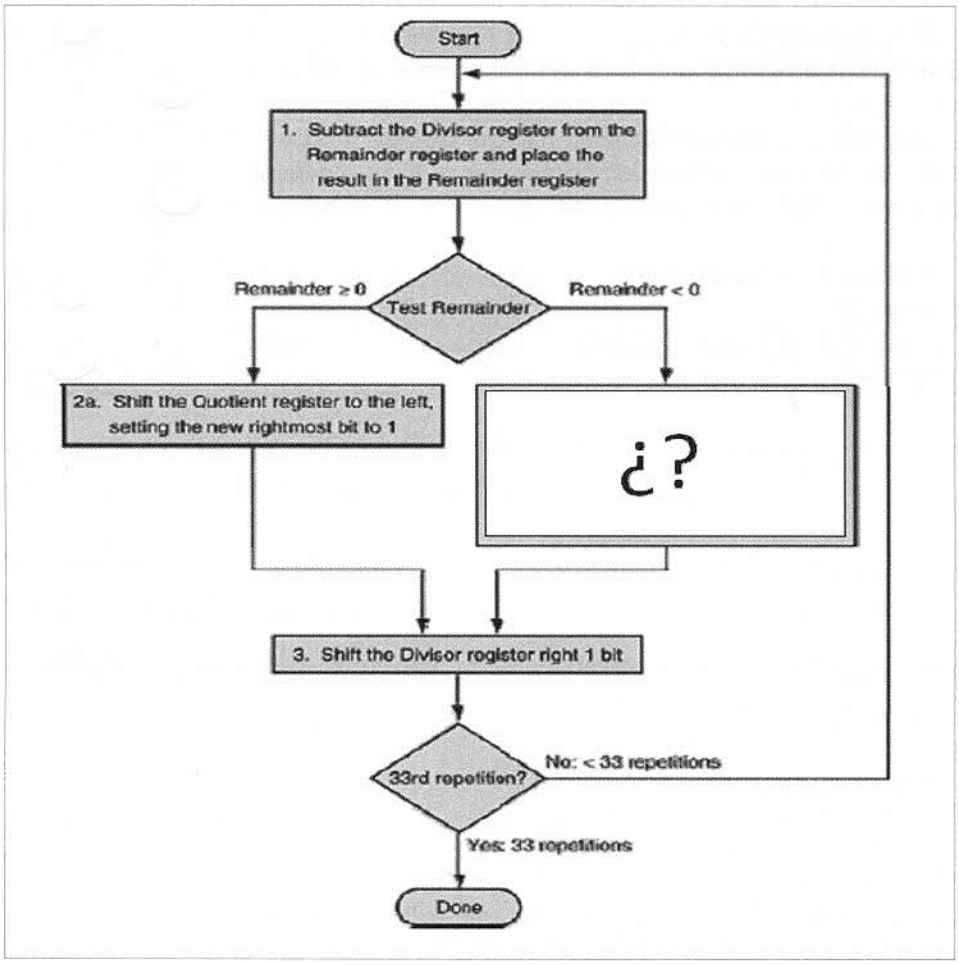
El exponente 8 pertenece al rango representable en precisión doble: [−1022, 1023], por tanto no hay "overflow"

En el cálculo no se agota el número de cifras decimales que se pueden representar, por tanto no se hace cambio alguno en la fase de redondeo.

¿Se produce "overflow"? Rodee con un círculo la respuesta correcta: SÍ (NO)

¿Se realiza algún cambio en la fase de redondeo? Rodee con un círculo la respuesta correcta: SÍ (NO)

5) Dado el siguiente diagrama de realización de una operación de división entera, utilizando el algoritmo y conjunto de registros comentados en las sesiones de teoría ,indique lo que hay que hacer en la caja marcada con interrogante. (1 punto)



Se restaura el valor original en el registro "Resto" mirando a dicho registro el valor del registro "Divisor" y colocando el resultado de esa resta en el registro resto. Y además, se desplaza el registro cociente hacia la izquierda, estableciendo en "0" el nuevo bit menos significativo del cociente.

Para cada una de las afirmaciones siguientes, indique si es verdadera o falsa rodeando con un círculo "F" si la considera falsa o "V" si la considera verdadera: (1 punto)

El número 1,3 se puede representar exactamente utilizando el formato de punto flotante IEEE-754 en precisión doble	V	(F)
Un grupo de 32 bits en cero representa el cero en formato de punto flotante IEEE-754	(V)	F
Si comparamos dos números representados en formato de punto flotante de IEEE-754 de precisión simple positivos, siempre es mayor el que contiene un número mayor en los 8 bits de exponente tomados como un número en binario puro.	(V)	F
Un producto de enteros realizado usando la instrucción `mult` nunca produce desbordamiento	V	(F)
Una división de enteros realizada utilizando la instrucción `div` nunca produce desbordamiento	V	(F)

NOMBRE:		DNI:	

1

Exprese en hexadecimal cuál es el contenido final del registro $s0 después de ejecutarse la siguiente porción de código:

```
addi $s0, $zero, -1
sll $t0,$s0,20
sub $s0,$zero,$s0
add $s0, $s0, $t0
```

$s0 = 0x FFF00001

Justificación:

La primera instrucción addi deja $s0=0xFFFFFFFF. La segunda desplaza 20 posiciones desplaza $s0 hacia la izquierda, insertando ceros, con lo que resulta $t0=0xFFF00000. La tercera deja $s0=0x00000001, y le suma a eso el contenido de $t0, con lo que resulta finalmente $s0=0xFFF00001

2

Exprese en hexadecimal cuál es el contenido final del registro $s0 después de ejecutarse la siguiente porción de código:

```
  la $t0 dato
  lw $s0, 0($t0)
  ori $s1, $s0,0xF00F
  sub $s0, $s1, $s0

      .data
dato: .word 0x1AB3C016
```

$s0 = 0x 00003009

Justificación:

"la" carga en $t0 la dirección que corresponde a la etiqueta "dato". "lw" deja $s0=0x1AB3C016. "ori" hace una operación OR bit a bit con el inmediato que se indica, con lo que resulta $s0=0x1AB3F01F y finalmente la instrucción "sub" le resta a eso el valor inicial almacenado en "dato", con lo que resulta $s0=0x1AB33019

3

Exprese en hexadecimal cuál es el contenido final del registro $s0 después de ejecutarse la siguiente porción de código:

```
  la $t0 v
  lw $t1,8($t0)
  lw $t2,12($t0)
  add $t1,$t1,$t2
  slti $t2,$t1,15
  add $s0,$t1,$t2

      .data
  v:  .word 3,18,3,7,5
```

$s0 = 0x 0000000B

Justificación:

"la" carga en $t0 la dirección de inicio de almacenamiento del array "v". La primera "lw" carga en $t1 el valor de v[2]=3; la segunda carga en $t2 el valor v[3]=7. La siguiente "add" deja en $t1 3+7=10. La tercera compara 3 con 15; como es menor, deja $t2=1. Por tanto, la última suma 1 a $t1, y deja el resultado en $t1. El valor de $t1 es 10+1, que en hexadecimal será 0x0B; por tanto resulta $t1=0x0000000B

4

Exprese en hexadecimal cuál es el contenido final del registro $s0 después de ejecutarse la siguiente porción de código:

```
        addi $s1,$zero,0x00AA
        addi $t1,$zero,1
        addi $t2,$zero,0x100
bucle:  or  $s1,$s1,$t1
        sll $t1,$t1,1
        addi $t2,$t2,-1
        beq  $t2,$zero,final
        j bucle
final:  sub $s0,$s1,$zero
```

$s0 = 0x FFFFFFFF

Justificación:

Las tres primeras instrucciones dejan $s1=0x000000AA, $t1=0x00000001 y $t2=0x00000100=32. Desde "bucle" hasta "final" está construido un lazo que en cada vuelta hace una operación OR de $t1 con $s1, y deja el resultado en $s1, luego desplaza una posición bit hacia la izquierda el contenido de $t1; y ese lazo da tantas vueltas como sea el valor inicial de $t2, que es 32. Por tanto, lo que hace en cada vuelta es poner a "1" un bit de $s1, comenzando por el bit 0 y terminando por el 31. Con ello, al final $s1 estará con todos sus bits en 1, y de ese modo, la última instrucción "sub" dejará $s0=0xFFFFFFFF

5

Exprese en base 10 cuál es el contenido final del registro doble $f0 después de ejecutarse la siguiente porción de código:

```
  la $t0 v
  l.d $f0,8($t0)
  s.d $f0,16($t0)
  l.d $f2,0($t0)
  sub.d $f0,$f2,$f0

      .data
   v: .double 5.9, 8.3, -10.5
```

$f0 (en base 10) = -2.4

Justificación:

"la" carga en $t0 la dirección inicial de almacenamiento de un array (al que podemos llamar "v"). La primera "l.d" carga en $f0 el valor v[1], esto es, 8.3. "s.d" almacena ese valor en v[2]. La siguiente instrucción "l.d" carga el valor v[0]en $f2, con lo cual queda $f2=5.9. Finalmente, "sub.d" calcula $f0 - $f2 y deja el resultado en $f0, con lo cual su valor final será -2.4.

ETIQUETAS	INSTRUCCIONES	COMENTARIOS
	.data	
x:	.word 6,21,-2,33,12,27,5,1	A partir de la posición etiquetada como "x" se almacenan los valores 6,21,-2,33,12,27,5,1,19,10 como valores enteros ocupando 4 bytes cada uno
	.word 19,10	
cad1:	.asciz ", "	A partir de la posición etiquetada como "cad1" se almacena lo que haga falta para imprimir los valores separados por comas
	.text	
__start		Etiqueta de punto de inicio de ejecución
	addi $t2, $zero, 1	Poner $t2 en 1
bucle:		Dirección de destino "bucle"
	beq $t2, $zero, final	Si $t2 es cero, ir a "final", y si no lo es continuar en la siguiente instrucción
	add $t0, $zero,$zero	Poner cero en $t0
	add $t2, $zero, $zero	Poner cero en $t2
lazo:		Dirección de destino "lazo"
	la $t3 x	Cargar en $s0 el valor de x[i] y en $s1 el valor de x[i+1] siendo i el valor contenido en $t0
	sll $t4, $t0, 2	
	add $t4, $t4, $t3	
	lw $s0, 0($t4)	
	lw $s1, 4($t4)	
	slt $t6, $s0,$s1	Si el valor de x[i] ($s0)es mayor que el de x[i+1] ($s1)salta a la posición "continuar", y si no lo es, continuar en la siguiente instrucción
	beq $t6, $zero, continuar	
	addi $t2,$zero,1	Poner "1" en $t2 e intercambiar los valores de los elementos x[i] y x[i+1]
	sw $s0, 4($t4)	
	sw $s1, 0($t4)	
continuar:		Dirección de destino "continuar"
	addi $t0, $t0, 1	Sumar 1 a $t0. Si el resultado es menor que 9, saltar a la posición "lazo", y si no, continuar en la siguiente instrucción.
	slti $t5, $t0, 9	
	bne $t5, $zero, lazo	
	j bucle	Saltar incondicionalmente a la posición "bucle"
final:		Dirección de destino "final"

ETIQUETAS	INSTRUCCIONES	COMENTARIOS
	add $t0, $zero, $zero	Escribir en pantalla el contenido final del array "v", de modo que los valores aparezcan todos en una línea, separados por comas. Nota: print-int tiene el número de servicio 1, y se le pasa el valor en $a0; print-string tiene el valor de servicio 4 y se le pasa la dirección de la cadena en $a0
lazo2:	la $t3 x	
	sll $t4, $t0, 2	
	add $t4, $t4, $t3	
	lw $a0, 0($t4)	
	addi $v0, $zero, 1	
	syscall	
	la $a0 cad1	
	addi $v0, $zero, 4	
	syscall	
	addi $t0, $t0, 1	
	slti $t5, $t0, 10	
	bne $t5, $zero, lazo	
	addi $v0, $zero, 10	Final del programa
	syscall	