1 **(1,5 puntos)** *Describa las fases de ejecución de la instrucción de dos palabras* **ST .R1,/dir***, indicando las acciones que se realizan en cada una de ellas. ¿Qué registros del procesador se utilizan en cada una de las fases? ¿Cuántas veces se accede a memoria en cada fase? Justifique su respuesta.*

SOLUCIÓN

En primer lugar está la fase de Fetch, que realiza un acceso a memoria a la dirección donde apunta el contador de programa e incrementa éste. La´primera palabra de la instrucción, que viene de memoria pasándo por el registro de datos se mete en el registro de instrucción.

Se realiza la decodificación.

A continuación se realiza otro acceso de lectura a la memoria poniendo de nuevo en el registro AR el contador de programa (e incrementándo éste) para obrener la dirección de almacenamiento. Ésta se pone en el registro de direcciones de memoria para poder completar la ejecución.

La fase de ejecución consiste en realizar un acceso de escritura a la dirección de memoria que acabamos de obtener poniendo en el registro de datos el contenido del registro .R1.

2 **(3,5 puntos)** *En un computador con palabras y direcciones de 32 bits, y direccionamiento a nivel de byte, que opera con aritmética entera en complemento a 2, se ejecuta el fragmento de código que se muestra a continuación.*

a) *Rellene la tabla que se proporciona en la hoja adjunta, con los valores sucesivos que van tomando los registros y las posiciones de memoria afectadas por la ejecución de cada una de las instrucciones, incluyendo los biestables Z (cero) y C (acarreo). Considere para ello lo siguiente:*

- *Los registros* ***R1, R2, R4 y R6*** *tienen el valor 100, 200, 2 y 1 respectivamente.*
- *A partir de la dirección de memoria 100 están almacenados los valores H'00000400, H'FFFFFFFE y H'00000000*
- *El tamaño de las instrucciones es de una palabra.*

```
LD .R3, #0[.R1++]
XOR .R6, .R6, .R6
SHL .R5, .R3, #1
BNC $4
SUB .R6, .R6, #1
ST .R6, #0[.R2++]
ST .R3, #0[.R2++]
SUB .R4, .R4, #1
BNZ $-36
```

SOLUCIÓN

Solución en hoja aparte

3 (5 puntos) *Un computador cuenta con dos formatos de representación de números en coma flotante:*

Formato 1. Formato estándar IEEE754 de 32 bits.

Formato 2. 16 bits. El bit superior representa el signo del número, los 7 siguientes el exponente y los 8 últimos la mantisa. Este formato sigue las convenciones del formato estándar IEEE754 en todo excepto en los tamaños, es decir, usa el mismo tipo de representaciones especiales, representación del exponente, bit implícito, situación de la coma, representaciones normalizadas y no normalizadas, así como bits de guarda y redondeo.

a) *Determine el rango y resolución del formato 2*

b) *Dadas las cadenas* $A = H'C4950000$ *y* $B = H'80300000$*, que son números representados en el formato 1, determine su valor decimal y represéntelos en el formato 2*

c) *Represente en el formato 2 el número decimal* $C = +0,8$

d) *Realice la operación* $A+C$ *en el formato 2, dejando el resultado en el formato de partida y determinando su valor decimal. Utilice redondeo al más proximo.*

e) *Determine la resolución con que está representado el número A en el formato 2. ¿Cuántos bits habría que añadir a la mantisa para que dicha resolución se dividiera por* 10*?*

SOLUCIÓN

a) Rango y resolución del formato.

Números normalizados.

Exponente: El estándar IEEE754 representa los exponentes en exceso a $2^{n-1}-1$. En este caso sería exceso a 63. Como se reserva el exponente mínimo (-63) para la representación del cero y los números no normalizados, y el exponente máximo ($+64$) para la representación del infinito y las indeterminaciones, el rango de exponente para la representación de números queda: $[-62, 63]$.

La mantisas normalizadas en este formato se representan en signo-magnitud, con bit implícito y la coma situada a la derecha del bit implícito:

Mantisa: $\pm \begin{cases} \texttt{1,00000000} & \rightarrow & 1 \\ \texttt{1,11111111} & \rightarrow & 2-2^{-8} \end{cases}$

El rango para números normalizados es: $\pm\left[1 \cdot 2^{-62}, (2-2^{-8}) \cdot 2^{63}\right]$

Números no normalizados.

Exponente: Siguiendo el estándar IEEE754, aunque los números no normalizados se representan con el exponente todo a ceros (valor -63), todos ellos tienen como exponente el más pequeño de los normalizados, en este caso -62. De este modo hay continuidad en la representación.

Mantisas: $\pm \begin{cases} \texttt{0,00000000} & \rightarrow & 0 \\ \texttt{0,00000001} & \rightarrow & 2^{-8} \\ \texttt{0,11111111} & \rightarrow & 1-2^{-8} \end{cases}$

El rango para números no normalizados es: $\pm\left[2^{-8} \cdot 2^{-62}, (1-2^{-8}) \cdot 2^{-62}\right] \cup 0$

La resolución depende del exponente y es: $2^{-8} \cdot 2^{E}$

b) Valor decimal y representación en el formato 2.

$A = \texttt{H'C4950000} = \texttt{1 10001001 00101010000000000000000} = \texttt{-1,00101010000000000000000} \cdot 2^{10}$

$A = \texttt{-10010101000} = \texttt{-1192}$

En el formato 2:

$A = \texttt{-1,00101010} \cdot 2^{10} = \texttt{1 1001001 00101010} = \texttt{H'C92A}$

$B = \texttt{H'80300000} = \texttt{1 00000000 01100000000000000000000}$

El exponente es todo ceros, luego estamos ante una representación no normalizada. Por tanto ponemos el bit implícito a cero y el exponente a -126

$B = \texttt{-0,01100000000000000000000} \cdot 2^{-126} = \texttt{-3} \cdot 2^{-129}$

En el formato 2:

Al ser el expoente tan pequeño, al tratar de representarlo en el formato 2 se produce UNDERFLOW

c) Representación del número.

$C = \texttt{+0,8} = \texttt{+,110011001} = \texttt{+1,10011001} \cdot 2^{-1} = \texttt{0 0111110 10011001} = \texttt{H'3E99}$

d) Suma $A + C$.

Este formato utiliza dos bits de guarda y un bit retenedor para la suma. Los números a sumar son:

$A = \texttt{-1,00101010} \cdot 2^{10}$ y $C = \texttt{+1,10011001} \cdot 2^{-1}$

Se restan los exponentes: $E_A - E_C = 10 - (-1) = 11$. Hay que desplazar la mantisa de C once lugares a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:

$C = \texttt{-0,00000000 001} \cdot 2^{10}$

M_A	$\texttt{1,00101010 000}$	
M_C	$-\ \texttt{0,00000000 001}$	
	$\texttt{1,00101001 111} \cdot 2^{10}$	*Normalizado*
Redondeo	$\texttt{0,00000000 100}$	
	$\texttt{1,00101010 011} \cdot 2^{10}$	

Como los tres bits que se usan como información de redondeo son 111 (no es el caso especial 100), el redondeo se realiza sumando un uno en la posición más alta de estos bits y truncando.

$A + C = \texttt{-1,00101010} \cdot 2^{10} = \texttt{1 1001001 00101010} = \texttt{H'C92A}$

$A + C = \texttt{-10010101000} = \texttt{-1192}$

e) Resolución y número de bits de la mantisa.

La resolución de este sistema de representación es $2^{-8} \cdot 2^E$. Como el número A se representa con exponente 10 su resolución de representación es: $2^{-8} \cdot 2^{10} = 2^2$

Si queremos que la resolución sea la décima parte, si p es el número de bits de la mantisa será:

$2^{-p} \cdot 2^{10} < 2^2/10 \rightarrow 2^{-p} < 2^{-8}/10 \rightarrow -p \cdot log2 < -8 \cdot log2 - 1$

$p > 8 + 1/log2 = 11,32 \rightarrow p = 12$

Hay que añadir a la mantisa un total de 4 bits.

ESTRUCTURA DE COMPUTADORES (GRADO MI)

PRIMER PARCIAL (10 Marzo de 2020)

Solución ejercicio 2

	Registros						Flags		Posiciones de memoria						
	R1	R2	R3	R4	R5	R6	Z	C	100	104	200	204	208	212	
Instrucción	100	200		2		1	0	0	0..0400	F..FE					
LD .R3, #0[.R1++]	104		0..400												
XOR .R6,.R6,.R6						0	1								
SHL .R5,.R3,#1					0..800			0							
BRNC															
ST .R6,#0[.R2++]		204									00..0				
ST .R3,#0[.R2++]		208										0..0400			
SUB .R4,.R4,#1				1			0	1							
BRNZ															
LD .R3, #0[.R1++]	108		F..FFE												
XOR .R6,.R6,.R6						0	1								
SHL .R5,.R3,#1					F..FC			1							
BRNC															
SUB .R6,.R6,#1						F..F	0	0							
ST .R6,#0[.R2++]		212											F..F		
ST .R3,#0[.R2++]		216												F..FE	
SUB .R4,.R4,#1				0			1	1							
BRNZ															