# Guia 1 – Data ALU – Grupo 2

**Ejercicio 1**

Si inicialmente los registros se encuentran en este estado:

a = $FF FFFFFF FFFFFF

b = $FF FFFFFF FFFFFF

x = $FFFFFF FFFFFF

y se ejecuta el siguiente código:

move #$3d,x1

move #$3d,a1

move #$3d,b

Indicar el estado final de los registros

a = $FF 00003D FFFFFF

b = $00 3D0000 000000

x = $3D0000 FFFFFF

La explicación de lo que ocurre está en la pag. 111 del Technical Training. Depende el destino lo toma como una fracción signada (X1, B) o un entero no signado (A1), y con eso cambia si hace zero padding a la izquierda o a la derecha.

**Ejercicio 2**

Si inicialmente los registros se encuentran en este estado:

a = $00 000000 000000

b = $00 000000 000000

x = $000000 000000

y se ejecuta el siguiente código:

move #$caba00,x1

move x1,a

move x1,b1

Indicar el estado final de los registros

a = $FF CABA00 000000

b = $00 CABA00 000000

x = $CABA00 000000

En este caso al mover de registro a acumulador es siempre signado fraccionario. Al ser 1 el MSBit se trata de un numero negativo, por lo que al escribir el acumulador A entero, A2 es $FF para mantener el signo.

**Ejercicio 3**

Si inicialmente los registros se encuentran en este estado:

a = $00 A00000 000000

y = $000000 000000

ccr = $00

y se ejecuta el siguiente código:

move a1,x1

move a,y1

move a,r7

move a1,x0

Indicar el estado final de los registros

x = $A00000 A00000

y = $7FFFFF 000000

r7 = $FFFF

ccr = $40

En el CCR, hay L=1 (limiting) cuando pasamos del accA entero a cualquier otro registro. Si bien en la última operación no hay limiting, el bit L debe ser forzado a volver a 0, sino queda latcheado este valor, denominándose un bit “sticky”. Luego para ver el valor que queda en Y1 tener en cuenta que A=$00A000... es 1.25, pero el mayor número que puede representar Y es 0.1111..., por lo tanto $7FFFFF. Lo mismo sucede con R7, pero en este caso se lo toma como un entero no signado así que su valor máximo es $FFFF, menor al $A0000000 de accA.

**Ejercicio 4**

Si inicialmente los registros se encuentran en este estado:

a = $00 000123 800000

b = $FF 000000 FFFFFF

x = $400000 400000

y se ejecuta el siguiente código:

macr x0,x1,a

rnd b

mpyr x0,x1,b

Indicar el estado final de los registros

a = $00 200124 000000

b = $00 200000 000000

En la primera instrucción el resultado de multiplicar y acumular es $002001238 y luego de redondear es $00200124. Luego el RND B redondea el $FFFFFF de B0 a un 1 en el LSBit de B1. Finalmente, la instrucción mpyr pisa lo que había en B y el resultado de la multiplicación es $00200000000000.

**Ejercicio 5**

Si inicialmente los registros se encuentran en este estado:

a = $00 000000 000000

sr = $0300

y se ejecuta el siguiente código:

move $400000,x0

add x0,a

add x0,a

1) indicar el estado final de los registros y justificar este resultado

sr = $0320

En este caso caso, estamos en No Scaling (S0=S1=0). Entonces, la parte signada entera de los acumuladores viene dada por los bits 55 a 47. Luego del segundo add, el último de estos bits tiene un valor distinto, por lo que E=1, resultando CCR=$20. Para el cálculo de U nos importan los bits 46 y 47 de accA, los cuales son distintos en ambos casos, por lo tanto U=0.

2) repetir considerando que inicialmente sr = $0700

sr = $0700

En este caso, estamos en Scale Down (S0=1 y S1=0). Entonces, la parte signada entera de los acumuladores viene dada por los bits 55 a 48. Siendo estos bits siempre iguales, E=0. Para el cálculo de U nos importan los bits 47 y 48 de accA, que al ejecutar el segundo add pasan a ser distintos, resultando U=0 al final del programa.

**Ejercicio 6**

Si inicialmente los registros se encuentran en este estado:

a = $00 000000 000000

x = $0C0000 600000

r0 = $0000

y se ejecuta el siguiente código:

add x1,a

rep #$a

norm r0,a

add x0,a

Indicar el estado final de los registros e indicar los cambios que se producen en el CCR a lo largo de la ejecución:

a = $00 C00000 000000

r0 = $FFFD = -3 (3 shifts a la izquierda)

En la primera instrucción tenemos U=1 porque los bits 46 y 47 son iguales a 0.

Durante la normalización U=1 hasta que luego de 3 shifts a la izquierda llega un 1 al bit 46, provocando U=0 y terminando la normalización.

Finalmente, la última instrucción deja los bits 46 y 47 en 1, provocando U=1 por ser iguales estos dos bits y E=1 por ser el bit 47 distinto a los más significativos de accA.

**Ejercicio 7**

Si inicialmente los registros se encuentran en este estado:

r0 = $0000 m0 = $FFFF

r4 = $0000 m4 = $FFFF

sr = $0800

Se tiene el siguiente mapa de memoria:

X:$0000 $10FEDC

X:$0001 $210FED

X:$0002 $4210FE

X:$0003 $84210F

X:$0004 $D84210

X:$0005 $FB8421

y se ejecuta el siguiente código:

move x:(r0)+,a

rep #6

move a,y:(r4)+ x:(r0)+,a

jlc OK

bset #0,y:$100

OK bclr #6,sr

Indicar el estado final de la memoria Y.

¿Qué significado tiene la memoria Y:$100?

Y:$0000 $21FDB8

Y:$0001 $421FDA

Y:$0002 $7FFFFF

Y:$0003 $800000

Y:$0004 $B08420

Y:$0005 $F70842

…

Y:$0100 $000001

Como SR=$0800, S1=1 y S0=0, estamos trabajando en Scale Up, lo que significa que la parte fraccionaria de A comienza en el bit 45. Por lo tanto, los bits transferidos de accA a la memoria Y serán los bits 46 a 23 inclusive, es decir, 23 bits de A1 y el MSBit de A0.

En el caso de la tercera iteración, A1 tiene el valor $4210FE, el cual es interpretado como un valor positivo mayor a 1, y satura al máximo valor positivo $7FFFFF.

En la cuarta iteración, en A se había cargado un valor negativo, por lo que satura al mínimo valor negativo, -1.0 = $800000.

En ambos casos hay limitación, por lo que luego de ejecutar estas instrucciones se tiene L=1 y la instrucción JLC no salta, se ejecuta el BSET y la memoria Y:$100 indica que ocurrió limitación en los movimientos de memoria.