

## Ejercicio 1

Si inicialmente los registros se encuentran en este estado:

a = \$ffffffffffffff  
b = \$ffffffffffffff  
x = \$ffffffffffff

y se ejecuta el siguiente código:

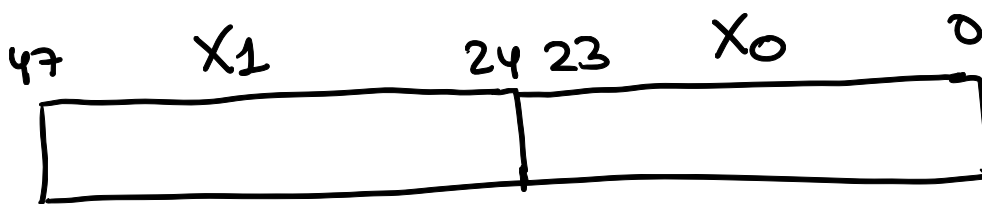
```
move #$3d,x1  
move #$3d,a1  
move #$3d,b
```

Indicar el estado final de los registros

a =  
b =  
x =

La instrucción `move #$3d,x1` lo que hace es usar direccionamiento inmediato y cargar el número 3d (hexa) en x1 /

(ver pág 80 pdf del manual)



Es un `move immediate short into 24 bit register` lo que resulta:

porque transfiere a x1

X1 X0  
f f f f f f ; f f f f f f → 3d 000 ; f f f f f f

Lo que tengo `move #$3d,a1` y como a1 es parte del acumulador A, se transfiere a la parte menos

significativa

A2 A1 A0 A2 A1 A0  
f f f f f f f f f f f f → f f 0000 3d f f f f f f

(ver ejemplo A pág 80 (pdf) del manual)

La última instrucción es `move #3d,b 10`

Cual resulta en:

B2	B1	B0	B2	B1	B0
FF	FFF	FFF	00	3d	000000

→

∴ luego, los estados finales de los registros son:

`a = FF 0000 3d FFFF FF`

`b = 00 3d 0000 000000`

`x = 3d 000 FFFF FF`

## Ejercicio 2

Si inicialmente los registros se encuentran en este estado:

`a = $0000000000000000`  
`b = $0000000000000000`  
`x = $0000000000000000`

y se ejecuta el siguiente código:

```
move #caba00,x1
move x1,a
move x1,b1
```

Indicar el estado final de los registros

`a =`  
`b =`  
`x =`

inicialmente tengo 

	X1	X0
000000	000000	

 y ahora tengo

	X1	X0
00	000000	

 luego con el `move x1,a`

con `a = 00 000000 000000` ahora tengo

A0	A1	A2
FF	caba	00 000000

 → mover de registro a acumulador

es signado frecuentemente. Se pone FF para mantener signo.

y con el move  $x_1, b_1$  tengo  $\overset{B_0}{00} \text{ cab} \overset{B_1}{2} 00 \overset{B_2}{000000}$

Finalmente, tengo

$a = ff \text{ cab} 00 000000$

$b = 00 \text{ cab} 00 000000$

$x = \text{cab} 00 000000$

### Ejercicio 3

Si inicialmente los registros se encuentran en este estado:

$a = \$00a00000000000$

$y = \$00000000000000$

$\boxed{CCR} = \$00$

Condition code register

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

$x =$

$y =$

$r7 =$

$CCR =$

Primero tenemos `move a1, x1` luego

$\overset{A_2}{a} = \overset{A_1}{00} \overset{A_0}{200000} \overset{A_0}{000000} \quad \overset{A_2}{a} = 00 \overset{A_1}{200000} \overset{A_0}{000000}$

$\overset{x_0}{x} = \overset{x_1}{xxxxxx} \overset{x_1}{xxxxxx} \quad \Rightarrow x = \overset{x_1}{xxxxxx} \overset{x_1}{200000}$

$\overset{x_1}{\text{move } a1, x1} \quad \overset{x_1}{x1} \quad \overset{x_1}{x0}$

Luego `move a, y1` lo que pasa es que tendremos mover un número mayor a 1 ( $2000 \rightarrow 1, 2s$ ) y por eso se prende el bit de carry del SR (CARRY: es un bit "sticky" entonces si

no lo volvemos a poner en 0 manualmente no cambia)

con move a, y1 tenemos

x = 2 000000 000000

a = 00 200000 000000

y = 7 fffffff 000000

↳ completa con el número positivo más grande posible para avisar que nos pasamos.

Wepo está move a, r7 y pasa algo parecido a lo de antes, y nos deja r7 = ffff para avisar que nos pasamos (a, x, y no se modifican). Se ve la simulación a continuación:

```
p:$e001 21c700      = move a,y1
trace
x=      $a000000000000 y=      $7ffffff000000
a=      $00a00000000000 b=      $000000000000000
      x1=      $a00000 x0=      $000000 r7=      $0000 n7=      $0000 m7=      $ffff
      y1=      $7ffffff y0=      $000000 r6=      $0000 n6=      $0000 m6=      $ffff
a2=      $00 a1=      $a00000 a0=      $000000 r5=      $0000 n5=      $0000 m5=      $ffff
b2=      $00 b1=      $000000 b0=      $000000 r4=      $0000 n4=      $0000 m4=      $ffff
      r3=      $0000 n3=      $0000 m3=      $ffff
pc=      $e003 sr=      $0340 omr=      $02 r2=      $0000 n2=      $0000 m2=      $ffff
la=      $0000 lc=      $0000 r1=      $0000 n1=      $0000 m1=      $ffff
ssh=      $0000 ssl=      $0000 sp=      $00 r0=      $0000 n0=      $0000 m0=      $ffff
ipr=      $0000 bcr=      $ffff
cyc=      000068 ictr=      000002 cnt1=      000000 cnt2=      000000 cnt3=      000000 cnt4=      000000
p:$e002 21d700      = move a,r7
trace
x=      $a000000000000 y=      $7ffffff000000
a=      $00a00000000000 b=      $000000000000000
      x1=      $a00000 x0=      $000000 r7=      $ffff n7=      $0000 m7=      $ffff
      y1=      $7ffffff y0=      $000000 r6=      $0000 n6=      $0000 m6=      $ffff
```

Finalmente tenemos el move a1, x0 que deja x = 2000000 2000000 (a e y quedan igual) como no cambiamos el sr queda como estaba antes (sr = 0340). La simulación es:

```

p:$e003 218400      = move a1,x0
trace
x=      $a00000a00000 y=      $7ffffff000000
a=      $00a000000000000 b=      $000000000000000
      x1=      $a00000 x0=      $a00000 r7=      $ffff n7=      $0000 m7=      $ffff
      y1=      $7ffffff y0=      $000000 r6=      $0000 n6=      $0000 m6=      $ffff
a2=      $00 a1=      $a00000 a0=      $000000 r5=      $0000 n5=      $0000 m5=      $ffff
b2=      $00 b1=      $000000 b0=      $000000 r4=      $0000 n4=      $0000 m4=      $ffff
      r3=      $0000 n3=      $0000 m3=      $ffff
pc=      $e005 sr=      $0340 omr=      $02 r2=      $0000 n2=      $0000 m2=      $ffff
la=      $0000 lc=      $0000 r1=      $0000 n1=      $0000 m1=      $ffff
ssh=      $0000 ssl=      $0000 sp=      $00 r0=      $0000 n0=      $0000 m0=      $ffff
ipr=      $0000 bcr=      $ffff
cyc=      000102 ictr=      000004 cnt1=      000000 cnt2=      000000 cnt3=      000000 cnt4=      000000

```

Entonces el estado final es:

X = 2 00000 200000

a = 00 200000 000000

Y = 777777 000000

r7 = 7777

CCR = \$40 (SR = 0340)

## Ejercicio 4

Si inicialmente los registros se encuentran en este estado:

```
a      = $00000123800000
b      = $ff000000ffffff
x      = $400000400000
```

y se ejecuta el siguiente código:

```
macr x0,x1,a
rnd  b
mpyr x0,x1,b
```

Indicar el estado final de los registros

```
a      =
b      =
```

valores iniciales

```
display
x=      $400000400000 y=      $000000000000
a=      $00000123800000 b=      $ff000000ffffff
      x1=      $400000 x0=      $400000 r7=      $0000 n7=      $0000 m7=      $ffff
      y1=      $000000 y0=      $000000 r6=      $0000 n6=      $0000 m6=      $ffff
a2=      $00 a1=      $000123 a0=      $800000 r5=      $0000 n5=      $0000 m5=      $ffff
b2=      $ff b1=      $000000 b0=      $ffffff r4=      $0000 n4=      $0000 m4=      $ffff
      r3=      $0000 n3=      $0000 m3=      $ffff
pc=      $e000 sr=      $0300 ovr=      $02 r2=      $0000 n2=      $0000 m2=      $ffff
la=      $0000 lc=      $0000 r1=      $0000 n1=      $0000 m1=      $ffff
ssh=      $0000 ssl=      $0000 sp=      $00 r0=      $0000 n0=      $0000 m0=      $ffff
ipr=      $0000 bcr=      $ffff
cyc=000000 ictr= 000000 cnt1=      000000 cnt2= 000000 cnt3= 000000 cnt4= 000000
p:$e000 000000 = nop
```

Verificamos que pusimos bien el código

```
disassemble
p:$e000 2000a3      = macr x1,x0,a
p:$e001 200019      = rnd b
p:$e002 2000a9      = mpyr x1,x0,b
p:$e003 000000      = nop
p:$e004 000000      = nop
p:$e005 000000      = nop
```

Chequeo código, NOSE PORQUE LO PONE AL REVES EL macr !!!

```
disassemble
p:$e000 2000a3      = macr x1,x0,a
p:$e001 200019      = rnd b
p:$e002 2000a9      = mpyr x1,x0,b
p:$e003 000000      = nop
p:$e004 000000      = nop
```



Ejecutando el macr obtenemos:

```
trace
x=      $400000400000 y=      $000000000000
a=      $00200124000000 b=      $ff000000fffff
      x1=      $400000 x0=      $400000 r7=      $0000 n7=      $0000 m7=      $ffff
      y1=      $000000 y0=      $000000 r6=      $0000 n6=      $0000 m6=      $ffff
a2=      $00 a1=      $200124 a0=      $000000 r5=      $0000 n5=      $0000 m5=      $ffff
b2=      $ff b1=      $000000 b0=      $ffffff r4=      $0000 n4=      $0000 m4=      $ffff
      r3=      $0000 n3=      $0000 m3=      $ffff
pc=      $e002 sr=      $0310 omr=      $02 r2=      $0000 n2=      $0000 m2=      $ffff
la=      $0000 lc=      $0000 r1=      $0000 n1=      $0000 m1=      $ffff
ssh=      $0000 ssl=      $0000 sp=      $00 r0=      $0000 n0=      $0000 m0=      $ffff
ipr=      $0000 bcr=      $ffff
cyc=      000051 ictr=      000001 cnt1=      000000 cnt2=      000000 cnt3=      000000 cnt4=      000000
```

Luego el rnd

```
p:$e001 200019 = rnd b
trace
x=      $400000400000 y=      $000000000000
a=      $00200124000000 b=      $ff000001000000
      x1=      $400000 x0=      $400000 r7=      $0000 n7=      $0000 m7=      $ffff
      y1=      $000000 y0=      $000000 r6=      $0000 n6=      $0000 m6=      $ffff
a2=      $00 a1=      $200124 a0=      $000000 r5=      $0000 n5=      $0000 m5=      $ffff
b2=      $ff b1=      $000001 b0=      $000000 r4=      $0000 n4=      $0000 m4=      $ffff
      r3=      $0000 n3=      $0000 m3=      $ffff
pc=      $e003 sr=      $0338 omr=      $02 r2=      $0000 n2=      $0000 m2=      $ffff
la=      $0000 lc=      $0000 r1=      $0000 n1=      $0000 m1=      $ffff
ssh=      $0000 ssl=      $0000 sp=      $00 r0=      $0000 n0=      $0000 m0=      $ffff
ipr=      $0000 bcr=      $ffff
cyc=      000068 ictr=      000002 cnt1=      000000 cnt2=      000000 cnt3=      000000 cnt4=      000000
```

Ahora el mpyr

```
p:$e002 2000a9 = mpyr x1,x0,b
trace
x=      $400000400000 y=      $000000000000
a=      $00200124000000 b=      $00200000000000
      x1=      $400000 x0=      $400000 r7=      $0000 n7=      $0000 m7=      $ffff
      y1=      $000000 y0=      $000000 r6=      $0000 n6=      $0000 m6=      $ffff
a2=      $00 a1=      $200124 a0=      $000000 r5=      $0000 n5=      $0000 m5=      $ffff
b2=      $00 b1=      $200000 b0=      $000000 r4=      $0000 n4=      $0000 m4=      $ffff
      r3=      $0000 n3=      $0000 m3=      $ffff
pc=      $e004 sr=      $0310 omr=      $02 r2=      $0000 n2=      $0000 m2=      $ffff
la=      $0000 lc=      $0000 r1=      $0000 n1=      $0000 m1=      $ffff
ssh=      $0000 ssl=      $0000 sp=      $00 r0=      $0000 n0=      $0000 m0=      $ffff
ipr=      $0000 bcr=      $ffff
cyc=      000085 ictr=      000003 cnt1=      000000 cnt2=      000000 cnt3=      000000 cnt4=      000000
p:$e003 000000 = nop
0>
```

Ahi se ve el estado final de las variables