Ejercicio 1

```
Si inicialmente los registros se encuentran en este estado:
a     = $ffffffffffff
b     = $ffffffffffff
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     =     FF 00003D FFFFFF
b     =     00 3D0000 000000
x     =     3D0000 FFFFFF
```

Ejercicio 2

```
Si inicialmente los registros se encuentran en este estado:
a     = $000000000000
b     = $000000000000
x     = $000000000000

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
```

Ejercicio 3

```
Si inicialmente los registros se encuentran en este estado:
a     = $00a00000000000
y     = $000000000000
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
```

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 = 00 200124 000000

b = 00 200000 000000

Ejercicio 5

Si inicialmente los registros se encuentran en este estado:

```
a     = $000000000000000
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 = 0330 El MSB (MR) no se ve modificado por las cuentas de aritmética, pero el LSB (CCR) se pone en $20 ya que se enciende el bit E, indicando que la parte entera signada del resultado está en uso.

2) repetir considerando que inicialmente sr = $0700

   sr = 0700 En este caso el SR no se ve modificado ya que en MR los bits de Scaling se encuentran en 01 (Scale Down). Esto implica que el punto fraccionario ahora se encuentra entre los bits 48 y 47 del registro A, y por ende el valor A = 00 800000 000000 no es el entero 1, es 0.5.

Ejercicio 6

Si inicialmente los registros se encuentran en este estado:

```
a     = $000000000000000
x     = $0c0000600000
r0    = $0000
```

y se ejecuta el siguiente código:

```
add   x1,a    El bit Unnormalised se setea. CCR = $10
rep   #$a     No es operacion aritmética, no hay cambios. CCR = $10
norm  r0,a    Luego de las dos primeras ejecuciones el resultado sigue unnormalised (CCR = $10). A partir de la tercera y hasta
              el final (décima), el resultado pasa a estar normalised (CCR = $00)
add   x0,a    El resultado está unnormalised y se usa la parte entera (E = 1 y U = 1). CCR = $30
```

Indicar el estado final de los registros

a = 00 C00000 000000

r0 = FFFD

Además indicar los cambios que se producen en el CCR a lo largo de la ejecución

Se indican al lado de cada instrucción

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
```

Y:$0000 $21FDB8
Y:$0001 $421FDA
Y:$0002 $7FFFFF El valor que contenía A es positivo ya que antes se le transfirió $4210FE, lo cual hace que en A haya quedado $00 421FE 00000. Debido al Scaling Mode en Scale Up, eso se interpreta como:
0000 0000 01.00 0010 0001 1111 1110 0000 ...
Cuyo valor fraccionario representado es mayor al máximo positivo admitido por el destino de 24 bits (0.111 11...). Hay limit positivo.
Y:$0003 $800000 Hay limit negativo ya que en A quedó $FF 84210F 000000. Debido al Scaling Mode esto se interpreta como:
1111 1111 10.00 0100 0000 1111 0000 ... igual a...
-0000 0000 01.11 1011 1111 0001 0000 ...
Cuyo valor es menor al mínimo negativo admitido por el destino de 24 bits (-1.000 00...).
Y:$0004 $B08420
Y:$0005 $F70842

La memoria Y:$100 estará seteada si hubo algun limit en la transferencia de datos. Como efectivamente existió tal limit, en esa posición hay un 1.

Indicar el estado final de la memoria Y.
¿Qué significado tiene la memoria Y:$100?

Ejercicio 8

Escribir la subrutina vect_max.
Compara elemento a elemento los vectores A y B, guarda el valor con modulo mayor en B. Recibe la dirección de inicio de los vectores en r0 y r4, y la cantidad de elementos en n0.

Utilizar la instrucción LOOP, y optimizar la cantidad de instrucciones dentro del bucle a la minima posible. Hint: considerar las instrucciones Tcc (transfer condicional).

Escribir un main de prueba y simular el resultado.