Problemas de Arquitectura del Repertorio de Instrucciones - ARM

1. Indica cuál es el resultado de ejecutar las siguientes instrucciones, dando el contenido final de los registros y posiciones de memoria para cada instrucción. Se supone que para cada instrucción a ejecutar el contenido de los registros y posiciones de memoria es el siguiente:

Registros	
r0	0x00000016
r1	0x00000054
r2	0xFFFFFFFF
r3	0x00000000
r4	0x00000004

Memoria	
0x00000000	0x03393826
0x00000004	0xEA0063AF
0x00000008	0x17FA8912
0x0000000C	0xBC983304
0x00000010	0x7845F34A
0x00000014	0x534B4AAA

```
1. add  r3, r0, r1
2. add  r2, r2, #1
3. sub  r4, r1, r0
4. sub  r4, r0, r1
5. mul  r4, r0, r1
6. or   r3, r1, r0
```

```
1. mov  r4, #0
2. lsr  r2, r0, r4
3. ldr  r0, [r4]
4. ldr  r0, [r4,#-4]
5. str  r2, [r4,r3]
6. str  r2, [r0]
```

RESPUESTA 1.

2. Codifica en ensamblador la siguiente condición IF-THEN:

```
if (x >= y) {
      x = x+2;
      y = y-2;
}
```

RESPUESTA 2.

3. Codifica en ensamblador la siguiente condición IF-THEN-ELSE:

```
if (x >= y) {
      x = x+2;
      y = y+2;
}
else {
      x = x-2;
      y = y-2;
}
```

RESPUESTA 3.

4. Codifica en ensamblador el siguiente bucle REPEAT-UNTIL:

```
a = 81;
b = 18;
do {
      a = a-b;
} while (a > 0);
```

RESPUESTA 4.

5. Codifica en ensamblador el siguiente bucle WHILE-DO:

```
n = 5;
fant = 1;
f = 1;
i = 2;
```

```
while (i <= n) {
      faux = f;
      f = f + fant;
      fant = faux;
      i = i+1;
}
```

RESPUESTA 5.

6. Codifica en ensamblador el siguiente bucle FOR:

```
for (i=2; i<=n; i++) {
      f=f+f;
}
```

RESPUESTA 6.

7. El siguiente programa calcula el máximo común divisor de dos números a y b según el algoritmo de restas de Euclides.Traducirlo a ensamblador del ARM:

```
int a=5, b=15, mcd;
While (a!=b){
      if (a>b)
            a=a-b;
      else
            b=b-a;
      }
mcd=a;
```

RESPUESTA 7.

8. Tenemos un vector de 10 componentes almacenado en la posición de memoria etiquetada como V. Diseña un programa en ensamblador que sume uno a cada una de sus componentes.

RESPUESTA 8.

9. Tenemos un vector de 6 componentes almacenado en la posición de memoria etiquetada como V. Diseña un programa en ensamblador que cuente el número de valores mayores que 0 que contiene.

RESPUESTA 9.

10. Implementar un programa en ensamblador que calcule la sucesión de *Fibonacci* y la almacene en un vector V de longitud arbitraria N. Esta sucesión infinita de números naturales queda definida como V(0)=0, V(1)=1, V(n)=V(n-1)+V(n-2). Se proporciona como ayuda el siguiente código de alto nivel:

    Nota: definir una constante N que almacene el número de elementos del vector V.

```
for i from 0 to N-2 do:
      V[i+2] = V[i] + V[i+1]
end for
```

RESPUESTA 10.

11. Dado un vector A de 12 componentes se desea generar otro vector B, tal que B sólo contiene las componentes de A que son números pares mayores que cero. Ejemplo:

    A = (0,1,**2**,7,-8,**4**,5,**12**,11,-2,**6**,3) à B = (2,4,12,6).

    Escriba un programa en lenguaje de alto nivel que implemente el procesamiento descrito y calcule el número de componentes del vector B. Traduzca el programa al lenguaje ensamblador del ARM.

RESPUESTA 11.

12. Dados dos vectores A y B de 10 componentes cada uno se desea construir otro vector C tal que:

$$C(i) = |A(i) + B(9-i)|, \quad i = 0,...,9.$$

Escriba un programa en lenguaje de alto nivel que construya el vector C. Traduzca el programa al lenguaje ensamblador del ARM.

RESPUESTA 12.

13. Traduce el siguiente programa escrito en un lenguaje de alto nivel a lenguaje ensamblador. La órden `swap(a, b)` intercambia los valores de las variables a y b.

```
int a=13, b=16;
While (a>10){
      a=a-1;
      b=b+2;
}
if (a<b)
      swap (a, b);
else
      b= a-1;
```

RESPUESTA 13.