PROBLEMAS DE FUNDAMENTOS DE COMPUTADORES (hoja 2.1)

1. Un programa tiene 140 instrucciones de las cuales 70 tardan en ejecutarse en un determinado procesador cuatro ciclos, 35 tardan cinco ciclos, 20 tardan tres ciclos y las 15 restantes tardan siete ciclos. Calcule el CPI promedio para dicho programa. Si el procesador funciona a una frecuencia de 2.0 GHz, determine el tiempo de ejecución del programa.
2. Se dispone de los siguientes datos de dos procesadores y de su rendimiento en la ejecución de una determinada tarea:
 - a. PowerPC que funciona a una frecuencia de 1.8 GHz y obtiene 700 MIPS.
 - b. Pentium 4 que funciona a 1.6 GHz y 850 MIPS.
 Calcule el CPI de cada procesador.
3. Considere los dos procesadores del ejercicio anterior. En la ejecución de un determinado programa los procesadores obtienen un CPI de 5.5 (PowerPC) y 7 (Pentium 4). El compilador genera un código máquina para dicho programa que tiene 9 millones de instrucciones (PowerPC) y 7.2 millones de instrucciones (Pentium) ¿Qué computador ejecutará más rápidamente la tarea?
4. 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	0000 0016h
r1	0000 0054h
r2	FFFF FFFFh
r3	0000 0000h
r4	0000 0004h

Memoria	
00h	0339 3826h
04h	EA00 63AFh
08h	17FA 8912h
0Ch	BC98 3304h
10h	7845 F34Ah
14h	534B 4AAAh

Las instrucciones a ejecutar son:

1. add r3, r0, r1	7. mov r4, #0
2. add r2, r2, #1	8. lsr r2, r0, r4
3. sub r4, r1, r0	9. ldr r0, [r4]
4. sub r4, r0, r1	10. ldr r0, [r4, #-4]
5. mul r4, r0, r1	11. str r2, [r4, r3]
6. or r3, r1, r0	12. str r2, [r0]

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

```

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

6. 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;
}
  
```

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

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

8. 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;
}
```

9. Codifica en ensamblador el siguiente bucle FOR:

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

10. 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;
```

11. Traduce la siguiente sentencia en C a ensamblador, donde **f**, **g** y **h** son enteros contenidos en memoria y **B** es un vector de 10 componentes: $f = g + h + B[4]$.

12. 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.

13. 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.

14. 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
```

15. 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:

$$A = (0, 1, 2, 7, -8, 4, 5, 12, 11, -2, 6, 3) \rightarrow B = (2, 4, 12, 6).$$

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.

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

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

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

17. Traduce el siguiente programa escrito en un lenguaje de alto nivel a lenguaje ensamblador. La orden `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;
```

18. Escribe un programa para el ensamblador del ARM que llame a una subrutina, `swap(int *a, int *b)`, encargada de intercambiar el contenido de dos posiciones de memoria. La subrutina recibirá como parámetros de entrada las posiciones de memoria correspondiente a a y b y deberá preservar el contenido de todos los registros que se empleen para realizar la operación.

Cuestión: ¿Qué registros debemos utilizar dentro de la función para evitar tener que salvar y restaurar registros durante el proceso?

19. Escribe un programa para el ensamblador del ARM que cuente el número de 0's de un vector de longitud arbitraria. Emplea para ello una subrutina llamada **cuenta0s** que reciba como parámetros de entrada toda la información necesaria para llevar a cabo la tarea.

20. Implementar el algoritmo de ordenación de la burbuja o *bubble sort* en ensamblador. Este sencillo algoritmo ordena los elementos de un vector de menor a mayor por medio de un procedimiento muy sencillo: recorre repetidas veces el vector, intercambiado posiciones sucesivas si $V(i) > V(i+1)$, hasta que no se realiza ningún cambio. Se proporciona como ayuda el siguiente código de alto nivel:

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

```
do
  swapped=0
  for i from 0 to N-2 do:
    if V[i] > V[i+1] then
      swap( V[i], V[i+1] )
      swapped = true
    end if
  end for
while swapped
```

21. Escriba un programa en lenguaje de alto nivel que llame a una función **fact** que calcule el factorial de un número **n** usando un bucle. Traduzca el programa al lenguaje ensamblador del ARM.

22. Escriba un programa en lenguaje de alto nivel que calcule el factorial de un número **n** usando recursividad, sabiendo que $\text{fact}(n) = n * \text{fact}(n-1)$. Traduzca el programa al lenguaje ensamblador del ARM.