PRÁCTICA 2: TIPOS DE DATOS Y MODOS DE DIRECCIONAMIENTO

OBJETIVOS Y COMPETENCIAS A ADQUIRIR:

- Comprensión de los modos de direccionamiento de datos
- Construcción de programas iterativos mediante el uso de instrucciones de salto condicionales

I PARTE: EJERCICIO 1

A) Diseña, codifica e implementa un programa en ensamblador de ARM para el LPC2105 que escriba en una tabla en memoria RAM una secuencia de medias palabras (16 bits) con los números: 0, 1, 2, …, 1023. Obsérvese el resultado en la ventana de memoria (vista en modo short). Obsérvese igualmente cómo se ensamblan los modos de direccionamiento.

Trabajo previo:

- Calcula el tamaño de la tabla en bytes
- Busca una directiva que permita reservar una porción de memoria

B) La secuencia anterior ocupa en memoria 2 KiB. Supón que esos 2 KiB se dividen en cuatro bloques de 512 bytes. Diseña, codifica e implementa un programa en ensamblador de ARM que rote los cuatro bloques así: El bloque 1 pasa a posición 4, el 4 a 3, el 3 a 2 y el 2 a 1. La secuencia seguirá ocupando la misma región de memoria. Obsérvese el resultado en la ventana de memoria del depurador.

Trabajo previo

Piensa si es necesario reservar espacio extra para resolver el apartado 2.

II PARTE: EJERCICIO 2

Tenemos un mensaje codificado en memoria que contiene los siguientes bytes:

`0x11, 0x12, 0x04, 0x03, 0x04, 0x0C, 0x10, 0x04, 0x05, 0x04, 0x0D, 0x12, 0xFF.`

El byte `0xFF` del mensaje codificado indica fin de mensaje.

Cada número en hexadecimal (salvo el `0xFF`) indica la entrada en la siguiente tabla de traducción a ASCII:

```
                                    ;cod ascii    entrada
        trad        DCB     0x43    ;C            00
                    DCB     0x44    ;D            01
                    DCB     0x45    ;E            02
                    DCB     0x2B    ;+            03
                    DCB     0x20    ;Espacio      04
                    DCB     0x3D    ;=            05
                    DCB     0xFF, 0xFF  ;no usadas  06,07
                    DCB     0x36    ;6            08
                    DCB     0x37    ;7            09
                    DCB     0x38    ;8            0A
                    DCB     0x39    ;9            0B
                    DCB     0x41    ;A            0C
                    DCB     0x42    ;B            0D
                    DCB     0xFF, 0xFF  ;no usadas  0E,0F
                    DCB     0x30    ;0            10
                    DCB     0x31    ;1            11
                    DCB     0x32    ;2            12
                    DCB     0x33    ;3            13
                    DCB     0x34    ;4            14
                    DCB     0x35    ;5            15
```

Esta tabla puede escribirse también de forma más compacta, aunque menos clara, aprovechando que el ensamblador genera directamente códigos ASCII a partir del carácter entrecomillado:

```
trad  DCB   "CDE+ =", 0xFF, 0xFF, "6789AB", 0xFF, 0xFF, "012345"
```

Diseña, codifica e implementa un programa en ensamblador de ARM para el LPC2105 que efectúe la traducción y copie a la memoria RAM del controlador el mensaje traducido. Obsérvese el resultado en la ventana de memoria (vista en modo ASCII).