Trabajo de Laboratorio Número 1

Sistemas con Microprocesadores 07/09/2022

Instrucciones

- Los problemas de ejercitación propuestos en el presente trabajo práctico pueden ser resueltos en forma individual.
- El planteo de la solución debe realizarse basándose en lo aprendido en las clases teóricas.
- Puede utilizar las clases de consulta para consultar problemas de enunciado y verificar la validez de la solución obtenida.
- Incluya comentarios en sus programas para que otros puedan entenderlo.
- Enfatice claridad, simplicidad y buena estructura. No pierda estos atributos para lograr mayor performance, a menos que así se especifique en el enunciado. Los programas deben ser razonablemente eficientes.
- Emplee rótulos y variables que posean significado, por ej. SUMA o PRUEBA en lugar de X o Y.
- En todos los casos comience por un diagrama de flujo de nivel intermedio (similar a un código en C).
- Posteriormente, se tomará una evaluación con problemas similares a los de este práctico, la cual deberá ser resuelta en forma individual. La fecha de evaluación se encuentra en el sitio web de la materia.

Conceptos involucrados

- Set de instrucciones de ARM.
- Modos de direccionamiento.
- Manejo de lazos.
- Manejo de tablas.

Problemas propuestos

1) Discuta en grupo si las siguientes afirmaciones son verdaderas o falsas. Se recomienda consultar los manuales de ARM, que se encuentran publicados en el sitio web de la cátedra.
   a) ¿En que casos usaría la instrucción **`TST`**? De 2 ejemplos.
   b) ¿Por qué en la instrucción **`BL`** el bit[0] del registro *rm* debe tener el valor 1?
   c) ¿La instrucción **`ADD`** demora 1T en todas sus variantes en su ejecución? Explique.

2) Escriba un programa para inicializar con 0x55 un vector. El tamaño de los datos del vector es de 16 bits y la cantidad de elementos se encuentra en la dirección **base** (también de 16 bits). El vector comienza en la dirección **base+2**.
   a) Resolver el ejercicio planteado como una rutina.
   b) Con directivas al ensamblador.
   c) Explique cuando se debe utilizar cada uno de los dos métodos.

3) *[Recomendado]* Escriba un programa que agregue un bit de paridad a una cadena de caracteres ASCII. La finalización de la cadena esta marcada con el valor 0x00 y el bloque comienza en la dirección **base**. Se debe poner en 1 el bit más significativo de cada caracter si y sólo si esto hace que el número total de unos en ese byte sea par. Por ejemplo:

Dato	**Resultado**
(base) = 0x06	(base) = 0x06
(base + 1) = 0x7A	(base + 1) = 0xFA
(base + 2) = 0x7B	(base + 2) = 0x7B
(base + 3) = 0x7C	(base + 3) = 0xFC
(base + 4) = 0x00	(base + 4) = 0x00

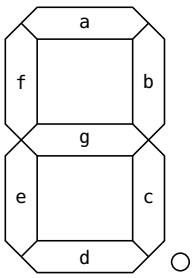
4) *[Profundización]* Codifique en lenguaje ensamblador del Cortex-M4 la rutina Actualizar Hora vista en clase teórica (transparencias 47 y 48) que incrementa el valor de los segundos, minutos y horas. Recuerde que esta función es llamada una vez cada 1ms. Ejecute el código en las placas EDUCIIA y compruebe su funcionamiento.

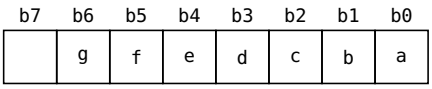
5) Escriba un programa para encontrar el mayor elemento en un bloque de datos. El tamaño de dato es de 8 bits. El resultado debe guardarse en la dirección **base**, la longitud del bloque está en la dirección **base+1** y el bloque comienza a partir de la dirección **base+2**. Por ejemplo:

Datos	**Resultado**
(base + 1) = 0x03	(base) = 0xF2
(base + 2) = 0x3A	
(base + 3) = 0xAA	
(base + 4) = 0xF2	

6) El método más simple para detectar alteraciones en un bloque de memoria consiste en agregar al mismo la suma byte a byte (sin considerar el carry) de todo el contenido del bloque. Este byte agregado se conoce como checksum o suma de comprobación. Escriba un programa que calcule el checksum de un bloque de datos cuya cantidad de elementos está almacenada en el registro **R0** y cuyo puntero al inicio del bloque se encuentra almacenado en las direcciones **base**. Se pide almacenar el resultado obtenido en la dirección **base+4**.

7) *[Recomendado]* Modifique el ejercicio visto en la clase de laboratorio para prender los segmentos correspondientes a un dígito determinado. El valor a mostrar, que deberá estar entre 0 y 9, se encuentra almacenado en el registro R0. Para la solución deberá utilizar una tabla de conversión de BCD a 7 segmentos la cual deberá estar almacenada en memoria no volatil. La asignación de los bits a los correspondientes segmentos del dígito se muestra en la figura que acompaña al enunciado.





8) *[Profundización]* Dado un número N menor que 2^{16}, se plantearán diferentes alternativas para escribir un programa que encuentre su raíz cuadrada entera aproximada, es decir, el mayor número entero n tal que $n^2 <= N$. Su solución debe contener el diagrama de flujo general del algoritmo y la estructura de datos, y diagrama de flujo detallado uno a uno convertible en assembler, más el programa en assembler en los siguientes casos:
   a) Deberá preparar una tabla en memoria a partir de la etiqueta **TABLA** una tabla en la cual estén almacenados en forma creciente los cuadrados de los números n que puedan ser solución del problema planteado. Indique la longitud y ancho de los ítems que tendrá la tabla. Suponga que el número N se encuentra ya cargado en el registro **R1** y la dirección de la Tabla en **R2**. Debe escribir un programa que coloque el resultado en el registro **R0**.
   b) Existe un algoritmo basado en las siguientes identidades:

$$1 = 1^2; 1+3 = 4 = 2^2; 1+3+5 = 9 = 3^2; \ldots; \sum_{i=1}^{n}(2i-1) = n^2$$

   Escriba el programa que calcule la raíz cuadrada de N empleando este método. Tanto N como el resultado deben ubicarse en los mismos registros que para el caso anterior .
   c) Deberá comparar las dos soluciones del problema planteado considerando espacio de memoria ocupado y tiempo de ejecución. Suponga ,por simplicidad, que todas las instrucciones duran 1T salvo los saltos tomados que duran 2T.
      - Evalúe en cada caso cuántos ciclos T lleva calcular la raíz cuadrada aproximada de 145.
      - ¿Qué cantidad de memoria requiere cada una de las soluciones en bytes?