Trabajo de Laboratorio Número 2

Sistemas con Microprocesadores 19/9/2022

Instrucciones

- Los problemas de ejercitación propuestos en el presente trabajo de laboratorio pueden ser resueltos en forma individual o grupal.
- El planteo de la solución debe realizarse basándose en lo aprendido en las clases teóricas.
- Puede utilizar las clases de consulta para resolver problemas de interpretación de los enunciados y para verificar la validez de las soluciones obtenidas.
- 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 escrito en lenguaje C).
- Recuerde que el procesador utilizado tiene un ordenamiento de byte del tipo Little-endian por lo cual la parte menos significativa de un número se ubica en la dirección de memoria mas baja.
- Posteriormente, se tomará una evaluación con problemas similares a los de este trabajo, la cual deberá ser resuelta en forma individual. La fecha de evaluación se encuentra en el sitio web de la materia.
- Las subrutinas transparentes deben cumplir el estándar EABI definido por ARM.

Conceptos involucrados

- Programación avanzada de ARM.
- Subrutinas.
- Paso de Parámetros.

Problemas propuestos

1) Discuta en grupo si las siguientes afirmaciones son verdaderas o falsas
   a) ¿Qúe requisitos debe cumplir una subrutina para ser transparente?
   b) ¿Cuáles son los beneficios de la programación modular?
   c) ¿Cómo se prueba en programación modular?
   d) ¿Cuándo se justifica el empleo de programación modular?
   e) ¿Es posible realizar una programación modular sin utilizar subrutinas transparentes?

2) Escriba una subrutina transparente que realice la suma de un número de 64 bits y uno de 32 bits. Esta subrutina recibe la dirección del número de 64 bits en **R0** y el número de 32 bits en **R1**. El resultado, de 64 bits, se almacena en el mismo lugar donde se recibió el primer operando. Escriba un programa principal para probar el funcionamiento de la misma. Por ejemplo:

Parámetros	R0 = 0x1008.0000	R1 = 0xA056.0102
Pre condiciones	M[R0] = 0x8100.0304	M[R0+4] = 0x0020.0605
Resultado	M[R0] = 0x2156.0406	M[R0+4] = 0x0020.0606

3) *[Recomendado]* Extraiga de la solución desarrollada por usted para el ejercicio 4 del laboratorio 1 el código necesario para implementar una subrutina transparente que realice el incremento de los segundos representados como dos dígitos BCD almacenados en dos direcciones consecutivas de memoria. La misma recibe el valor numérico 1 en el registro **R0** y la dirección de memoria donde está almacenado el dígito menos significativo en el registro **R1**. La subrutina devuelve en el registro **R0** el valor 1 si ocurre un desbordamiento de los segundos y se debe efectuar un incremento en los minutos, o 0 en cualquier otro caso.

4) *[Recomendado]* Reimplemente la solución desarrollada por usted para el ejercicio 4 del laboratorio 1 usando dos llamadas sucesivas a la subrutina del ejercicio anterior para incrementar los segundos y minutos.

5) Modifique la subrutina transparente que realiza el incremento de los segundos desarrollado en el ejercicio anterior para que pueda incrementar o decrementar el valor del par BCD almacenado en memoria. Para el caso de decremento se utilizará el valor -1 en el registro **R0**.

6) *[Profundización]* Escriba la subrutina que ejecute el gestor de eventos correspondiente a las teclas del reloj. Ésta debe identificar la subrutina correspondiente a la tecla presionada (gestor del evento) y saltar a la misma. Para ello emplea el código de la tecla presionada almacenado en el registro **R0** como un índice en una tabla de saltos que comienza en el lugar **base**. Cada entrada en la tabla de saltos contiene la dirección de la primera instrucción de la subrutina correspondiente (punto de entrada). El programa debe transferir control a la dirección que corresponde al índice. Por ejemplo, si el índice fuera 2, el programa saltaría a la dirección que está almacenada en la entrada 2 de la tabla. Como es lógico cada entrada tiene 4 bytes. Por ejemplo:

R0=2	**Datos**	**Resultado**
	(base) = 0x1A00.1D05	(PC) = 0x1A00.5FC4
	(base + 4) = 0x1A00.2321	
	(base + 8) = 0x01A0.5FC4	
	(base + 12) = 0x01A0.7C3A	

7) *[Profundización]* Escriba una subrutina que invierta una cadena de caracteres ASCII se encuentra almacenada en memoria. Esta subrutina recibe el puntero al inicio de la cadena en **R0** y un puntero al final en **R1**.
   a) Diseñe una subrutina **cambiar** que invierta el primer y el último caracter de la cadena. Esta subrutina recibe el puntero al primer elemento en R0 y al último en R1.
   b) Diseñe ahora una subrutina recursiva **invertir** que utiliza la rutina anterior para intercambiar los elementos del extremo de la cadena y a si misma para invertir el resto.