

Examen: tema 2^B – arquitectura de computadores (15%)

Fecha: viernes 26 de abril de 2024

Duración: una hora y cuarenta y cinco minutos (1h:45m)

NOMBRE: SOLUCIÓN

EJERCICIOS PARA RESOLVER

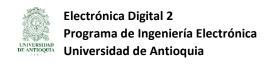
1) (21pts) Relacione el valor de la columna izquierda con el equivalente de la derecha, de acuerdo con sus conocimientos sobre la arquitectura ARM de 32-bits estudiada en clase:

- a) Arquitectura.
- b) Registros, Caché, RAM, Disco.
- c) Contador de Programa.
- d) ANDEQ R0, R0, R0, LSL #0.
- e) R0-R3.
- f) Carácterística de Arq. Harvard.
- g) MOV PC, LR.

- **g.** SUB PC, LR, #0.
- a. Referido a las instrucciones de la CPU y acceso a operandos.
- e. Pasar argumentos a las funciones.
- **f.** Buses independientes para acceder a instrucciones y datos.
- **d.** 0x0000.
- c. Dirección de la siguiente instrucción a ejecutar.
- **b.** Jerarquía de memoria.
- 2) (21pts) Afirmaciones para completar relacionadas con la arquitectura ARM de 32-bits estudiada en clase:
 - a. La arquitectura **VON NEUMANN** no permite el acceso simultáneo a memoria de instrucciones y datos.
 - b. Las instrucciones de carga y almacenamiento en memoria son típicas de la filosofía de diseño de instrucciones **RISC**.
 - c. La instrucción LDR R8, =0xABCD, es en realidad un LDR R8, [REGISTRO, #CTE], donde REGISTRO corresponde a **PC**.
 - d. Cuando un procesador con dos niveles de memoria caché, L1 y L2, solicita leer la memoria, accederá al nivel L2 siempre que el dato no se encuentre disponible en el nivel L1.
 - e. Si la instrucción BL F1 está en la dirección 0x8 y la primera instrucción de la función F1 está en la dirección 0x100, el valor inmediato que se codifica en el campo imm24 de la instrucción BL F1 es **0x3C**.
 - f. La instrucción MOV RO, #0xFFFFFF00 se puede reemplazar por la instrucción MVN RO, #0xFF.
 - g. El valor inmediato de la instrucción MOV RO, #0xC0000002 se puede reemplazar por el valor 0x0B siempre que se le indique a la CPU rotar éste último 2 posiciones hacia la derecha.
- 3) (12pts) Para los dos esquemas de memoria mostrados en la siguiente figura, Little-Endian y Big-Endian, indique como quedaría almacenado el valor de 64-bits 0x123C2E1B_3E2C10FA en la dirección de memoria 0x20 en cada uno de ellos:

Little-Endian				Dirección	Big-Endian			
FA	10	2C	3E	0000_0020	12	3C	2E	1B
Byte 20 Byte 21 Byte 22 Byte 23					Byte 20 Byte 21 Byte 22 Byte 23			
1B	2E	3C	12	0000_0024	3E	2C	10	FA
Byte 24 Byte 25 Byte 26 Byte 27					Byte 24 Byte 25 Byte 26 Byte 27			

4) (16 pts) Una función en ensamblador para ARM denominada £2 recibe seis parámetros, los cuatro primeros en los registros RO-R3 y los últimos dos en el *stack:* el quinto en la dirección [SP, #4] y el sexto en la dirección [SP]. El retorno se realiza en el registro RO. Esta función (£2) es llamada dentro de otra función (£1), la cual deberá pasarle los seis argumentos. En el siguiente fragmento de código ya se han cargado los primeros cuatro argumentos en los registros RO y R3, pero hace falta cargar los argumentos cinco y seis en el *stack*, los cuales se encuentran en los registros R5 y R9, respectivamente. Indique las instrucciones que se



Examen: tema 2^B – arquitectura de computadores (15%)

Fecha: viernes 26 de abril de 2024

Duración: una hora y cuarenta y cinco minutos (1h:45m)

encargan de llevar los argumentos cinco y seis al *stack*, junto con la apropiada manipulación del registro SP. **Restricción**: sólo pueden emplearse las instrucciones ADD, LDR y STR.

```
F1:

...
BL F2
...

Líneas antes de la instrucción BL F2
STR R5, [SP, #-4]!
STR R9, [SP, #-4]!

Líneas luego de la instrucción BL F2
ADD SP, SP, #8
STR R9, [SP, #-4]!
```

5) (30pts) Dado el prototipo de una función en lenguaje C: int32_t findValue(int32_t array[], uint32_t size, int32_t value), escriba el cuerpo de dicha función empleando lenguaje ensamblador para el procesador ARM estudiado en clase, que retorne el índice de la primera posición del vector array de tamaño size que sea igual a value. Si ninguna posición dentro del vector array es igual a value, entonces se deberá retornar -1. Para los parámetros de entrada, emplee los registros RO, R1 y R2, para recibir array, size y value, respectivamente. Para el retorno de la función, utilice el registro RO. Emplee el stack para guardar en memoria el valor de los registros preservados que vaya a emplear en la función, si es requerido.

```
findValue:
       MOV R3, #0
                                        // Inicializa index a 0
Loop:
       CMP R3, R1
                                        // Si index >= size, no encontrado
       BHS EndLoop
       LDR R12, [R0, R3, LSL #2]
                                        // Cargar array[index] en R12
       CMP R12, R2
                                        // Compara array[index] con value
       BEQ EndFindValue
                                        // Si iguales, retornar index
       ADD R3, R3, #1
                                        // Si no, intentar sgte. index
           qool
EndLoop:
       MOV R3, \#-1
                                        // Value no encontrado
EndFindValue:
                                        // Retornar index o -1 en R0
       MOV RO, R3
       MOV PC, LR
```