Electrónica Digital 2
Programa de Ingeniería Electrónica
Universidad de Antioquia

Examen: tema 2^A – arquitectura de computadores (15%)
Fecha: miércoles 18 de octubre de 2023
Duración: una hora y cuarenta y cinco minutos (1h:45m)

SOLUCIÓN

EJERCICIOS PARA RESOLVER

1) (24pts) 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) Empleados para pasar argumentos	**e.** Guarda el valor 0x08 en mem[0x08]
b) SUB R0, R15, R15	**f.** Registro SP
c) 0xe2500001	**g.** 0xe4810004
d) Empleados para almacenar direcciones de mem	**b.** R0 = 0x00000000
e) STR R1, [R1], R1 con R1 = 0x08	**a.** Registros R0 a R3
f) Puntero de pila del procesador	**h.** Guarda el valor 0x08 en mem[0x10]
g) STR R0, [R1], #4	**d.** Registros R0 a R15
h) STR R1, [R1, R1] con R1 = 0x08	**c.** SUBS R0, R0, #1

2) (24pts) Afirmaciones para completar o para responder verdadero (V) o falso (F) relacionadas con la arquitectura ARM de 32-bits estudiada en clase:
   a. En un sistema de cómputo con direccionamiento de memoria por bytes y una organización de bytes tipo ***big endian***, al almacenar el valor de 32-bits 0xAABBCCDD en la dirección de memoria 0x04, el byte 0xBB quedará en la dirección **0x05** y el byte 0xCC quedará en la dirección **0x06**.
   b. La instrucción MOV R0, #0xFF10 no compila porque el valor a almacenar no cabe en el registro R0 **F**.
   c. La instrucción STR R1, [R0, #3] genera siempre un problema de acceso a memoria (múltiplo de cuatro) independiente del valor de R0 **F**.
   d. Sí la instrucción ADD R0, R1, PC se encuentra ubicada en la dirección de memoria 0x0000_000C, y el valor de R1 = 0x0000_0008, el valor del registro R0 después de ejecutar la instrucción será **0x0000_001C**.
   e. La instrucción **MVN** R0, #0xFF carga un 0xFFFFFF00 en el registro R0.
   f. Si la dirección de memoria 0x02 tiene el valor de 8-bits 0x80, la instrucción LDRSB R0, [R1, #2], con R1 = 0x00, cargará en el registro R0 el valor de 32-bits **0xFFFF_FF80**.
   g. Es posible codificar la instrucción MOV R0, #61696 de manera correcta en código de máquina **V**.
   h. Dos instrucciones en ensamblador requeridas para realizar la operación NAND R0, R1, R2 serían: **AND R0, R1, R2** y **MVN R0, R0**

3) (22pts) El siguiente programa en ensamblador para el procesador ARM, debería inicializar todas las posiciones del vector **vec** de la siguiente manera: vec[0] = 3, vec[1] = 2, vec[2] = 1, vec[3] = 0; sin embargo, por errores en la programación, la inicialización no es la correcta. Por favor responda las siguientes preguntas:

```
        .text
        .equ    N, 4
        LDR     R1, =vec
        MOV     R0, #N
For1:
        STR     R0, [R1], #4
        SUBS    R0, R0, #1
        BNE     For1
EndFor1:
        B       EndFor1
```

```
.data
vec:    .ds.l N
```

a) (12pts) El valor en cada posición del vector **vec** cuando el programa es ejecutado hasta encontrar por primera vez la instrucción **B** EndFor1. Tenga en cuenta que el vector está ubicado a partir de la dirección 0x0000_0080.

vec[0] = **4** vec[1] = **3** vec[2] = **2** vec[3] = **1**

b) (10pts) ¿Qué línea o líneas de código cambiaría usted para solucionar el problema?

Línea original: **`STR R0, [R1], #4`** Línea corregida: **`SUBS R0, R0, #1`**
Línea original: **`SUBS R0, R0, #1`** Línea corregida: **`STR R0, [R1], #4`**

4) (16pts) La siguiente función (**Func1**) escrita en ensamblador para el procesador ARM modificará los registros R4, R5 y LR dentro del bloque **Loop1** ... **EndLoop1**, por lo que se deberá emplear la pila para guardar una copia de los registros y recuperarlos antes del retorno. Sabiendo que los bloques **Init1 ... Loop1** ... **EndLoop1** no hacen uso de la pila del sistema, indique que instrucciones se deberían agregar (max. 2) para garantizar que los valores de los registros indicados se preserven:

```
   Func1:
1)      CMP     R1, R2
2)      BNE     Init1
3)      MOV     PC, LR
   Init1:
4)      …
   Loop1:
5)      …
   EndLoop1:
6)      MOV     PC, LR
```

Nueva línea de código: PUSH {R4, R5, LR} Entre Init1 y 4)
Nueva línea de código: POP {R4, R5, LR} Entre EndLoop1 y 6)

5) (14pts) Para el siguiente fragmento de código en C, escriba el equivalente código en ensamblador para el procesador ARM, empleando instrucciones condicionales sin saltos (Bxx), asumiendo que **e** y **f** son valores con **signo de 32-bits** que están en los registros R0 y R1, respectivamente:

```
Code:
if (e >= f)
  e = e * f;
else
  e = e >> f;
```

Solución:

1) CMP R0, R1
2) MULGE R0, R0, R1
3) ASRLT R0, R0, R1