Plancha 4
Arquitectura ARM

Arquitectura del Computador
Licenciatura en Ciencias de la Computación

1) Suponga que un arreglo, `array`, contiene 10 palabras de 4 bytes. Un compilador asocia las variables `x` y `y` con los registros `r0` y `r1`, respectivamente, y que la dirección inicial del arreglo está almacenada en el registro `r2`. Traduzca las siguientes declaraciones en C a instrucciones en ensamblador ARM:

a) `x = array[7] + y;`

b) `array[10] = array[8] + y;`

En cada uno de los casos realizar el código completo para poder compilar y depurar.

2) Suponga que el registro `r3` contiene el valor `0x8000`. ¿Qué contendrá el registro después de ejecutar las siguientes instrucciones?

a) `STR r6, [r3, #12]`

b) `STRB r7, [r3], #4`

c) `LDRH r5, [r3], #8`

d) `LDR r12, [r3, #12]!`

3) Suponiendo que se tiene una arquitectura ARM *little-endian*,

a) ¿qué contendría el registro `r4` después de ejecutar las siguientes instrucciones?

```
STR r6, [r3]
LDRB r4, [r3]
```

El registro `r6` contiene el valor `0xBEEFFACE` y el registro `r3` contiene el valor `0x8000`.

b) ¿Qué pasaría si fuera *big-endian*?

4) Indicar el contenido del registro r7 luego de ejecutadas las siguientes instrucciones:

a) `MOV r7, #0x8C, 4`

b) `MOV r7, #0x42, 30`

c) `MVN r7, #2`

d) `MVN r7, #0x8C, 4`

5) Usando el esquema de rotación, calcule la instrucción y la rotación necesarias para cargar los siguientes valores inmediatos en el registro `r2`:

a) `0xA400`

b) `0x3D8`

c) `0x17400`

d) `0x1980`

e) `0xffffff00`

f) `-1`

6) Indique si las siguientes constantes pueden cargarse en un registro sin crear un *literal pool* y usando solo una instrucción:

a) `0xC0000034`

b) `0x12340000`

c) `0x77777777`

d) `0xFFFFFFFF`

e) `0xFFFFFFFE`

7) Indicar el inconveniente en cada una de las siguientes instrucciones ARM e indicar otra alternativa válida:

a) `ADD r3, r7, #1023`

b) `SUB r11, r12, r3, LSL #32`

8) Desde la versión ARMv6T2 de la arquitectura ARM, puede usarse la siguiente instrucción para cargar un valor inmediato de 32 bits o la dirección de 32 bits de una etiqueta en un registro:

```
ldr r1, =0x12abcdef
ldr r1, =etiqueta
```

Ahora bien, si todas las instrucciones tienen 32 bits de longitud (o 16 en el perfil THUMB), ¿cómo explica que en una sola instrucción pueda usarse un valor inmediato de ese tamaño?

9) El algoritmo del campesino ruso se puede implementar en lenguaje C de la siguiente manera:

```
unsigned int campesino_ruso(unsigned int i, unsigned int j)
{
        unsigned int res = 0;

        while (j > 1) {
                if (j & 1) {
                        res += i;
                        j -= 1;
```

```
                } else {
                        i *= 2;
                        j /= 2;
                }
        }
        return res + i;
}
```

Escriba esta función para arquitectura ARM, utilizando solo una instrucción de salto en su implementación.

Pista: Aprovechar las capacidades de ejecución condicional de las instrucciones de la arquitectura ARM.

10) Escriba una función en Assembler ARM que dado un argumento i entre 0 y 31 en `r0`, calcule y retorne 2^i. Para calcular este valor sólo puede usar instrucciones de copia (`MOV`).

11) Escriba una función que, dados 4 valores en punto flotante que representan una matriz 2×2, devuelva su determinante.

12) Dado el siguiente código en C que suma los elementos de dos arreglos de igual longitud con datos de tipo entero:

```
#include <stdio.h>

int a[]={1, 2, 3, 4};
int b[]={2, 3, 4, 5};

int suma(int a[], int b[], int L){
    int i, result=0;
    for(i=0; i<L; i++){
        result = result + a[i] + b[i];
    }
    return result;
}

int main(){
    int L=4;
    printf("La suma es: %d\n", suma(a, b, L));
    return 0;
}
```

Escribir un código equivalente en Assembler ARM.

13) Dado el siguiente código en C que suma las componentes de dos vectores de tipo `float` y, además, realiza la suma de los dos arreglos:

```
    #include <stdio.h>

    float a[5] = {1.0, 2.0, 3.0, 4.0, 5.0};
    float b[5] = {2.0, 4.0, 6.0, 8.0, 10.0};
    float c[5];

    int main()
```

```
{
    float suma=0;
    for(int i=0; i<5; i++){
        printf("%f\n", c[i] = a[i] + b[i]);
        suma = suma + c[i];
    }
    printf("La suma de las componentes es: %f\n", suma);
    return 0;
}
```

Escribir un código equivalente en Assembler ARM.