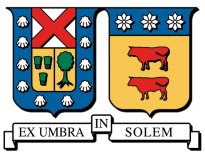
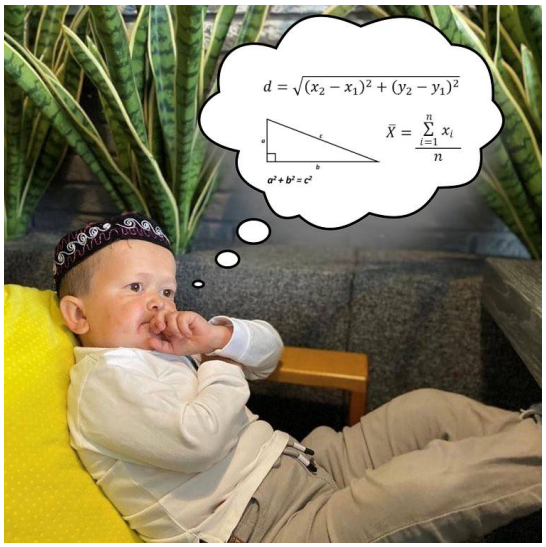


Universidad Técnica Federico Santa María

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORES

INFORME TAREA 4
"ARM ASSEMBLY: EJERCICIOS MATEMÁTICOS MEDIANTE QTARMSIM"



Felipe Guicharrousse, 202073555-7

Gabriel Vergara, 202073616-2

Julio 2022

1. Resumen

En esta informe, se presenta el trabajo y desarrollo de la tarea n°4, en el cual se resuelve una serie de ejercicios matemáticos (figura 15) mediante el uso del lenguaje ensamblador ARM Assembly, utilizando la plataforma QtARMSim. El desarrollo en código de los distintos ejercicios (modos) a resolver podemos observarlos en la figura 2, 3, 4 y 5 y los resultados de cada modo en la sección de resultados del informe. Finalmente, se realiza un análisis y conclusión del trabajo.

2. Introducción

El objetivo de esta tarea es resolver una serie de ejercicios matemáticos utilizando el lenguaje de ensamblador ARM Assembly y todas sus funciones mediante el uso de la plataforma QtARMSim, para simular el funcionamiento de un procesador con dicho lenguaje.

Para este trabajo se presentó el problema de una serie de ejercicios matemáticos que al no recordar como solucionarlos a mano, se decide utilizar una máquina ARM que se tiene en el escritorio para el desarrollo y programación de las soluciones a dichos ejercicios. Los ejercicios a solucionar podemos encontrarlos en la figura 15 en anexos.

3. Desarrollo

Para el desarrollo de esta tarea, se comienza codificando en lenguaje ARM Assembly en la plataforma QtARMSim. En primer lugar, se obtiene el modo del ejercicio a realizar usando la operación ldr, para luego crear condicionales que permitan saber que modo se utilizará y ejecutar solo dicho modo, esto con las operaciones cmp (compare) y beq (branch equal) para poder dirigirse al ejercicio a realizar, esto se puede observar en la figura 1:

```
17          mov r0, #0
18          ldr r0, =modo
19          ldr r0, [r0,#0]
20
21          cmp r0,#1
22          beq modo_1
23
24          cmp r0,#2
25          beq modo_2
26
27          cmp r0,#3
28          beq modo_3
29
30          cmp r0,#4
31          beq modo_4
32
33          b end
34
```

Figura 1: Código para condicionales y realizar el modo que se indique

Ahora, se hará enfoque en cada modo mostrando y explicando a grandes rasgos el desarrollo y solución de estos.

Para el modo 1 se solicita obtener la hipotenusa (c) de un triángulo dado el sus dos catetos (a y b), por lo que para obtener dicho valor se utiliza la ecuación de Pitágoras ($a^2 + b^2 = c^2$). En base a dicha ecuación, mediante el uso de operaciones y funciones que nos ofrece el programa, se obtiene la siguiente solución para el modo 1:

```
37 modo_1:
38          mov     r0, #0
39          ldr     r7, =arreglo
40          ldr     r0, [r7, #0]
41          bl      qfp_int2float
42          mov     r1, r0
43          bl      qfp_fmu
44          mov     r4, r0
45          ldr     r0, [r7, #4]
46          bl      qfp_int2float
47          mov     r1, r0
48          bl      qfp_fmul
49          mov     r1, r4
50          bl      qfp_fadd
51          bl      qfp_fsqrt
52          bl      qfp_float2int
53          bl      qfp_int2float
54          str     r0, [r7, #8]
55
```

Figura 2: Código para el desarrollo del modo 1

Explicando el código mostrado en la figura 2 primero guardamos en un registro el arreglo con los valores de los catetos, luego se saca el primer cateto y lo transformamos a float con la función *bl qfp_int2float* ya que para usar este tipo de operaciones el valor debe encontrarse en tipo float y el arreglo inicialmente se encuentra en tipo entero. Lo siguiente que se hace es multiplicar el cateto por el mismo valor de este para obtener el cuadrado del cateto, esto utilizando dos registros para la multiplicación correspondiente y otro registro para guardar dicho resultado, para el otro cateto se realiza el mismo procedimiento. Finalmente, para obtener el valor de la hipotenusa se utilizaron las funciones (qfp_fadd, qfp_fsqrt) y para dejar el resultado en entero y truncado, se utilizaron las funciones ($qfp\_float2int$ y $qfp\_int2float$). El resultado lo almacenamos en la dirección de memoria de un registro con la operación **str**, esto para mostrar el resultado final por pantalla.

Para el modo 2 se solicita obtener el cateto, lo cual es muy parecido al modo 1, pero esta vez de input se coloca un cateto y la hipotenusa. Cabe recalcar el orden de este. Nuevamente, se usará la fórmula de Pitágoras, pero esta vez dejaremos despejado un cateto: $a^2 = c^2 - b^2$. Por consiguiente, usando una metodología muy parecida al modo 1, se tiene que:

```
73 modo_2:
74          ldr     r7, =arreglo
75          ldr     r0, [r7, #0]
76          bl      qfp_int2float
77          mov     r1, r0
78          bl      qfp_fmul
79          mov     r4, r0
80          ldr     r0, [r7, #4]
81          bl      qfp_int2float
82          mov     r1, r0
83          bl      qfp_fmul
84          mov     r1, r4
85          bl      qfp_fsub
86          bl      qfp_fsqrt
87          bl      qfp_float2int
88          bl      qfp_int2float
89          str     r0, [r7, #8]
```

Figura 3: Código para el desarrollo del modo 2

Como se ve en la imagen 3, primero almacenamos el arreglo en un registro y desde este mismo se comienza a

extraer el primer dato, el cual es cateto, y mediante la función *bl qfp_int2float* explicada anteriormente, se le realiza el Cast a Float para las futuras funciones. Guardamos el dato en otro registro para multiplicar el mismo número, y obtener el cuadrado de este. Para el caso de la hipotenusa se procede exactamente lo mismo que se explicó anteriormente, por lo que luego se hace la resta de los dos cuadrados, teniendo sumo cuidado en que el primer valor sea la hipotenusa y el segundo el cateto, para que nos del valor correcto. Calculamos la raíz cuadrada con la función *qfp_fsqrt*. Luego truncamos nuestro número de la misma forma que en el modo 1 porque al pasar un Float a Integer este se queda solamente con la parte entera, eliminando los decimales. Luego se regresa el número a Float por un tema del Print de pantalla. Finalmente se guarda el resultado en la dirección de memoria dada.

Para el modo 3 se debe obtener la distancia dado dos puntos con sus respectivas coordenadas x e y, por lo que para obtener dicho valor, se utiliza la ecuación $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, siendo d el resultado de la distancia, x_1 e y_1 las coordenadas de un punto y x_2 e y_2 las coordenadas del otro punto. En base a dicha ecuación, mediante el uso de operaciones y funciones que nos ofrece el programa, se obtiene la siguiente solución para el modo 3:

```
108 modo_3:
109         mov     r0, #0
110         ldr     r6, =arreglo
111         ldr     r1, [r6,#8]
112         ldr     r2, [r6, #0]
113         sub     r0, r1, r2
114         mov     r7, r0
115         mul     r0,r7,r0
116         mov     r4, r0
117         ldr     r1, [r6,#12]
118         ldr     r2, [r6, #4]
119         sub     r0, r1, r2
120         mov     r7, r0
121         mul     r0,r7,r0
122         mov     r5, r0
123         add     r0, r4,r5
124         bl      qfp_int2float
125         bl      qfp_fsqrt
126         bl      qfp_float2int
127         bl      qfp_int2float
128         str     r0,[r6,#16]
```

Figura 4: Código para el desarrollo del modo 3

En el desarrollo del modo 3 (figura 4) se comienza con lo mismo, se guarda el arreglo en un registro, luego se extrae los números necesarios, sabiendo las posiciones de estos, se sabe que cada elemento del arreglo están separados por 4 bytes, por lo cual si se quiere tomar el elemento de la posición 0 se tiene que mover 0 $bytes$ extrayendo así el valor de x_1 y para obtener el valor x_2 se tiene que extraer el valor del registro con la distancia de 8 $bytes$, ya que se encuentra en la segunda posición (tercer elemento del arreglo, recordar que las posiciones comienzan en el 0). Se resta estos valores y se elevan al cuadrado. Luego, se hace exactamente lo mismo con el eje y, teniendo en cuenta que y_1 se ubica a 4 $bytes$ de distancia al estar en la posicion 1, el y_2 está a 12 $bytes$ al ser el último elemento. Después, se suma estos dos cuadrados calculados y pasamos estos valores a Float, ya que, anteriormente se usó solamente funciones que eran soportadas por Integer, mientras que las función de raíz cuadrada no es soportada por esta. Finalmente, el resultado lo guardamos en una dirección de memoria.

Para el modo 4 se debe calcular el promedio de un conjunto **n** de números enteros, para desarrollar este ejercicio se utiliza la ecuación mostrada en la figura 17 en anexos. En base a dicha ecuación, mediante el uso de operaciones y funciones que nos ofrece el programa, se obtiene la siguiente solución para el modo 4:

```
147 modo_4:
148         mov     r0, #0
149         ldr     r7, =largo
150         ldr     r7, [r7, #0]
151         sub     r7, r7, #1
152         ldr     r6, =arreglo
153         mov     r5, #4
154         mul     r5, r7, r5
155         ldr     r0, [r6, r5]
156
157 loop_modo_4:
158         cmp     r7, #0
159         beq     solucion_modo_4
160         mov     r5, #4
161         sub     r7, r7, #1
162         mul     r5, r7, r5
163         ldr     r1, [r6, r5]
164         add     r0, r0, r1
165         b       loop_modo_4
166
167 solucion_modo_4:
168         bl      qfp_int2float
169         mov     r3, r0
170         ldr     r0, =largo
171         ldr     r0, [r0, #0]
172         bl      qfp_int2float
173         mov     r1, r0
174         mov     r0, r3
175         bl      qfp_fdiv
176         bl      qfp_float2int
177         bl      qfp_int2float
178         mov     r7, r0
```

Figura 5: Código para el desarrollo del modo 4

Para el código anterior 12 se guarda el largo en un registro, ya que este va a servir para sumar todos los valores, porque como se sabe el largo del arreglo es variable, por lo cual se usara el largo del arreglo como guía de extracción de datos, con lo cual le restamos 1 a este mismo, porque como recordaremos las posiciones van del 0 al $Largo - 1$ así que al tener esta guía se rescata el último valor del arreglo, recordando que hay que multiplicar este valor por 4 porque las posiciones de los arreglos están separados por 4 $bytes$ de distancia. Guardamos este primer valor y entramos al ciclo llamado "Loop_Modo_4" el cual siempre antes de hacer el ciclo va a preguntar si el valor auxiliar que tenemos como guía de extracción es 0, ya que si este es el caso no existen más números que sumar. Si no es el caso, entramos al ciclo y se hace el mismo procedimiento de antes, se resta 1 a la guía para luego hacer el rescate el valor anterior, recordar que este programa extrae los valores del arreglo, comienza desde el último y finaliza con el primero, posteriormente se suma este dato al valor anterior, así repetitivamente. Cuando haya sumado todos los valores del arreglo, se irá a la sección "Solución_Modo_4" , el cual lo único que hace es tomar el valor de la suma y dividirlo. Obviamente, después se encarga de truncar el valor encontrado usando el método utilizado en los modos anteriores. Finaliza guardando el dato en una dirección de memoria, para posteriormente mostrarlo en pantalla.

Cabe mencionar que el método para mostrar por pantalla los resultados fueron igual para todos los modos, teniendo en cuenta que para cada uno existe un texto diferente y se muestra el valor respectivo de dicha solución, lo cual observaremos en la siguiente sección de resultados.

4. Resultados

En la siguiente sección se presentan los resultados de los datos de ejemplo entregados en el enunciado de la tarea (figura 16), se le agregó un texto al output para saber que se estaba obteniendo al calcular el valor:

- Modo 1:

Figura 6: Modo 1 con sus entradas y salida por pantalla

Figura 7: Modo 1 con sus entradas y salida por pantalla

- Modo 2:

Figura 8: Modo 2 con sus entradas y salida por pantalla

Figura 9: Modo 2 con sus entradas y salida por pantalla

- Modo 3:

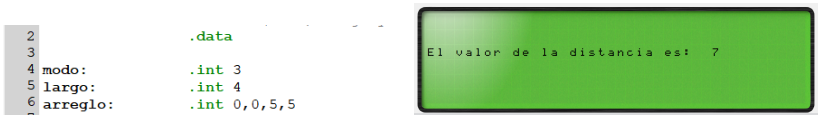


Figura 10: Modo 3 con sus entradas y salida por pantalla

Figura 11: Modo 3 con sus entradas y salida por pantalla

- Modo 4:

Figura 12: Modo 4 con sus entradas y salida por pantalla

Figura 13: Modo 4 con sus entradas y salida por pantalla

Figura 14: Modo 4 con sus entradas y salida por pantalla

5. Análisis

Analizando y discutiendo el desarrollo y resultados de este trabajo, podemos decir que en un principio se tenía el gran problema de no saber muy bien la sintaxis y el funcionamiento general de ARM Assembly, por lo que no se tenía el suficiente conocimiento para utilizar las operaciones, memoria y uso de registros en general para desarrollar los problemas planteados. Gracias al estudio, búsqueda de información, materia pasada en clase y un poco de documentación del lenguaje, se logró entender y poder desarrollar de manera correcta todos los ejercicios planteados en la tarea, desde entender como funciona un registro, guardando información o data importante en este, hasta el uso de operaciones y funciones que permitan obtener los valores esperados y mostrarlos en pantalla mediante la plataforma.

6. Conclusión

En conclusión, se considera que el nivel de finalización de esta tarea es bastante alto, ya que se logra desarrollar y obtener las soluciones para los distintos ejercicios planteados, utilizando los datos de ejemplo y teniendo 100 % de éxito, es decir, todos los outputs esperados. Finalmente, se puede decir que se aprendió y entendió más a fondo el lenguaje de máquina ARM Assembly, entendiendo como usar registros, operaciones permitidas en el ensamblador y uso de memoria en general, así, obteniendo resultados completamente satisfactorios.

7. Anexos

Ejercicio 8.1 — *Dado un triángulo rectángulo de lados a, b, c de los cuales conoce dos, calcule el valor del tercero.*

Ejercicio 13.5 — *Sean a, b dos puntos en el plano cartesiano. Calcule la distancia entre los puntos.*

Ejercicio 17.4 — *Calcule el promedio de un conjunto de n números enteros.*

Figura 15: Serie de ejercicios matemáticos a resolver en ARM Assembly

modo	largo	arreglo	output
1	2	3, 4	5
1	2	3, 5	5
2	2	3, 5	4
2	2	3, 10	9
3	4	0, 0, 5, 5	7
3	4	6, 9, 6, 9	0
4	4	1, 2, 3, 4	5
4	1	14	14
4	5	-10, -6, 4, 5, 7	0

Figura 16: Datos de ejemplo con sus entradas y salidas respectivas

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Figura 17: Fórmula para la obtención del promedio