Tarea #3

Microprocesadores

Tiempo de respuesta regresión y tabla

Jorge Isaac Aguirre Morgado

Introducción

Para cualquier sistema siempre queremos que sea lo más eficiente posible, queremos usar la menor cantidad de recursos y que la respuesta sea lo más rápida posible. Para este trabajo comparamos el tiempo de respuesta de una tabla de búsqueda y de una regresión polinomial. Por pura teoría esperamos que la búsqueda de tabla sea más rápida que la regresión, sin embargo, la tabla nos cuesta más espacio de memoria, veremos cual termina siendo lo más eficiente

Desarrollo

Primeramente, usamos un código en Python que nos generó la tabla de búsqueda para nuestro archivo principal. Para esta tabla fue necesaria nuestra ecuación obtenida en la anterior tarea.

```
import sympy as sp
import numpy as np

x = sp.Symbol('x')

poly = 140.2359 + (-0.1164) * x + (0.00001) * x**2

# Convertir la expresión simbólica en una función de NumPy para un
cálculo rápido
f = sp.lambdify(x, poly, "numpy")

# El tamaño de la tabla es 2^12 = 4096 (para un ADC de 12 bits)
lut_size = 2**12

# Calcular todos los valores de la tabla de búsqueda
# np.arange para generar todos los valores de entrada del ADC (0 a
4095)
values = f(np.arange(lut_size))
values = np.clip(values, 0, 100)

# Escribir en el archivo de cabecera C
with open("lookuptable.h", "w") as file:
    file.write("#ifndef LOOKUPTABLE_H\n")
    file.write("#define LOOKUPTABLE_H\n\n")
```

```
    file.write("#define LUT_SIZE {}\n\n".format(lut_size))
    file.write("static const uint32_t percentage_values[LUT_SIZE] =
{\n")

    # Formatear el array con 16 valores por línea
    for i in range(0, lut_size, 16):
        row = ", ".join(str(int(v)) for v in values[i:i+16])
        file.write("  {},\n".format(row))

    file.write("};\n\n")
    file.write("#endif // LOOKUPTABLE_H\n")

print("Archivo lookuptable.h generado exitosamente.")
```

Al correr este código se nos genera un nuevo archivo .h con los valores de la tabla

```
#ifndef LOOKUPTABLE_H
#define LOOKUPTABLE_H

#define LUT_SIZE 4096

static const uint32_t percentage_values[LUT_SIZE] = {…
};

#endif // LOOKUPTABLE_H
```

La table es un arreglo con el valor calculado dependiendo de la posición de la tabla

Una vez que tenemos nuestro archivo .h ya podemos usar nuestro código principal en la ESP32

```
#include "esp_timer.h"
#include "lookuptable.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/adc.h"
#include <stdio.h>
#include <stdlib.h>

int regression_func(int adc_value) {
    float a = -0.1164;
    float b = 0.00001;
    float c = 140.2359;

    float adc_float = (float)adc_value;
```

```
    float humidity_float = c + (a * adc_float) + (b * adc_float *
adc_float);

    if (humidity_float > 100.0) return 100;
    if (humidity_float < 0.0) return 0;
    return (int)humidity_float;
}

void app_main(void) {
    int result;

    // Configuramos ADC1, canal 4 (GPIO32)
    adc1_config_width(ADC_WIDTH_BIT_12); // 12 bits
    adc1_config_channel_atten(ADC1_CHANNEL_4, ADC_ATTEN_DB_11); //
GPIO32

    while (1) {
        // Leemos el ADC del pin 32
        int adc_val = adc1_get_raw(ADC1_CHANNEL_4);

        // Medimos tiempo de la función de regresión
        int64_t start = esp_timer_get_time();
        result = regression_func(adc_val);
        int64_t end = esp_timer_get_time();
        printf("ADC = %d | Resultado de Regresion = %d (Humedad),
tiempo = %lld us\n", adc_val, result, (end - start));

        // Medimos tiempo de la tabla
        start = esp_timer_get_time();
        result = percentage_values[adc_val]; // usando la tabla
        end = esp_timer_get_time();
        printf("ADC = %d | Resultado de la Tabla = %d (Humedad),
tiempo = %lld us\n", adc_val, result, (end - start));

        // Esperamos 1 segundo antes de la siguiente iteración
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
```

Con este código nos imprimirá el valor de ADC leído, la respuesta de la tabla, el tiempo que le tomó buscarlo, la respuesta de la regresión y el tiempo que le tomó calcularlo

```
ADC = 685 | Resultado de Regresion = 65 (Humedad), tiempo = 1 us
ADC = 685 | Resultado de la Tabla = 65 (Humedad), tiempo = 4 us
ADC = 639 | Resultado de Regresion = 69 (Humedad), tiempo = 1 us
ADC = 639 | Resultado de la Tabla = 69 (Humedad), tiempo = 0 us
ADC = 592 | Resultado de Regresion = 74 (Humedad), tiempo = 1 us
ADC = 592 | Resultado de la Tabla = 74 (Humedad), tiempo = 4 us
ADC = 544 | Resultado de Regresion = 79 (Humedad), tiempo = 1 us
ADC = 544 | Resultado de la Tabla = 79 (Humedad), tiempo = 4 us
ADC = 493 | Resultado de Regresion = 85 (Humedad), tiempo = 1 us
ADC = 493 | Resultado de la Tabla = 85 (Humedad), tiempo = 4 us
ADC = 457 | Resultado de Regresion = 89 (Humedad), tiempo = 1 us
ADC = 457 | Resultado de la Tabla = 89 (Humedad), tiempo = 4 us
ADC = 419 | Resultado de Regresion = 93 (Humedad), tiempo = 5 us
ADC = 419 | Resultado de la Tabla = 93 (Humedad), tiempo = 5 us
ADC = 622 | Resultado de Regresion = 71 (Humedad), tiempo = 1 us
ADC = 622 | Resultado de la Tabla = 71 (Humedad), tiempo = 4 us
```

Podemos notar que ambos métodos nos da los mismos resultados, sin embargo, aparentemente la regresión pareciera que fue más rápida que la tabla, realmente es una regresión muy sencilla y probablemente por eso fue efectuada más rápida la operación y nuestro bucle para encontrar el valor podría llegar a ser más tardado.