Jasdeep Signh

Tln-513, 5th year SET, IPSA

Practical work: IPSA_SCHED

Student: BLOT Samuel

Introduction

In this final assignment the objective is to design an RTOS with four periodic tasks that are :

Task 1:

Print a status message such as "Working !"

Task 2:

Convert a fixed Fahrenheit temperature value to degree Celsius

Task 3:

Multiply two long integer numbers and print the result

Task 4:

Perform binary search on a predefined list of 50 elements for a specific value

We will also be trying to find a WCET for all of these tasks

Part I: WCET

In this part we will be working toward calculating the WCET of each of the tasks we will be working in a C++ to code the tasks an measure them

```cpp
int run_and_measure() {
    vector<long long> elapsedTimes;

    for (int i = 0; i < iterations; ++i) {
        //START MEASURE
        auto start = high_resolution_clock::now();

        // Convert F° to C°
        double fahrenheit = 100;
        double celsius = (5 / 9) * (fahrenheit-32);

        //END MEASURE
        auto elapsed = high_resolution_clock::now() - start;

        elapsedTimes.push_back(duration_cast<microseconds>(elapsed).count());
    }



    // Find the maximum time
    long long maxTime = *max_element(elapsedTimes.begin(), elapsedTimes.end());

    cout << "Maximum time taken by a single run: " << maxTime << " microseconds" << endl;

    return 0;
}

int main() {
    run_and_measure();
    return 0;
}
```

How we did that is that we first defined a function run and measure that will measure our task here the task 2 how many times we want (here 100 000 times). And then we store the value into a vector and use a premade command to find the maximum and conclude the WCET of that:

	Task 1	Task 2	Task 3	Task 4
WCET (in µs)	981	66	659	129
Measures taken	100,000	1,000,000	100,000	1,000,000

WCET of the 4 tasks measured on my Machine

Part II: RTOS

```
/* Priorities at which the tasks are created. */
#define YOUR_TASK1_PRIORITY                ( tskIDLE_PRIORITY + 1 )
#define YOUR_TASK2_PRIORITY                ( tskIDLE_PRIORITY + 2 )
#define YOUR_TASK3_PRIORITY                ( tskIDLE_PRIORITY + 3 )
#define YOUR_TASK4_PRIORITY                ( tskIDLE_PRIORITY + 4 )
/* The rate at which data is sent to the queue.  The times are conve
 * milliseconds to ticks using the pdMS_TO_TICKS() macro. */

#define TASK1_FREQUENCY                    pdMS_TO_TICKS( 4000UL )
#define TASK2_FREQUENCY                    pdMS_TO_TICKS( 200UL )
#define TASK3_FREQUENCY                    pdMS_TO_TICKS( 3000UL )
#define TASK4_FREQUENCY                    pdMS_TO_TICKS( 800UL )
```

I chose to make it so the priority of each Task is the same as their index (i.e. Task1 = Highest priority etc …) and the frequency is higher as their WCET is higher

On my machine I have 1000000 clocks per seconds so with a quick convetion I can get C in ticks which is the same

	Task 1	Task 2	Task 3	Task 4
C (WCET in µs)	981	66	659	129
C (WCET in ticks)	981	66	659	129
TI (in ticks)	4000	200	3000	800
p	0	1	2	3

Task 1: U1=981/4000≈0.245

Task 2: U2=66/200≈0.33

Task 3: U3=659/3000≈0.2197

Task 4: U4=129/800≈0.16125

Now lets sum up to get Utotal and see if it's schedulable:

Utotal=U1+U2+U3+U4

Utotal≈0.245+0.33+0.2197+0.16125

Utotal≈0.95595

So our tasks are schedulable (on my machine at least).