Lab 8 Report

Brandon Shumin

ECE 6680

Introduction:

The purpose of this lab was to perform RMA on a theoretical Inertial Navigation System (INS) by using the provided theorem for RMA and calculating crucial theorem values in a table before coding. The crucial values needed were the total max blocking time (in milliseconds) for each priority of task, the period (in milliseconds), and the run time for each priority of task (in milliseconds) which was provided to us. These values are then used to determine if each given task passes the RMA with resource blocking theorem at various values of k and l in C code.

Data Tables:

The data shown in Figure 1 is the formatted version of the data provided in the lab manual for the RMA calculations. These tasks were sorted by priority based on the period time with 0 being highest priority and 7 being the lowest.

Each resource, result table usage, I/O channel usage, and disk usage, were then separately analyzed to obtain the max blocking times (in milliseconds) for each priority of task. These calculations can be seen in Figure 2.

Finally, the total max blocking for each priority was determined by summing the max blocking for each resource at each priority level as shown in Figure 3.

				Resources		
Feature	Priority	Period (ms)	Run Time (ms)	Result Table Usage (ms)	I/O channel usage (ms)	Disk usage (ms)
Compute attitude data	0(Highest)	10.56	1.30	0.20	0.00	2.00
Compute velocity data	1	40.96	4.70	0.20	0.00	3.00
Compose attitude message	2	61.44	9.00	0.00	3.00	0.00
Display data	3	100.00	23.00	0.30	0.00	0.00
Compose navigation message	4	165.00	38.30	0.00	6.00	0.00
Run-time Built-In Test (BIT)	5	285.00	10.00	0.00	0.00	1.00
Compute position data	6	350.00	3.00	0.20	0.00	3.00
Compose test message	7(Lowest)	700.00	2.00	0.00	2.00	0.00

Figure 1: Formatted data provided from lab manual sorted by priority

Result Table Usage Blocking			
Priority	Direct	Pushthrough	Max Blocking
0(Highest)	0.30	0.00	0.30
1	0.30	0.30	0.30
2	0.00	0.30	0.30
3	0.20	0.20	0.20
4	0.00	0.20	0.20
5	0.00	0.20	0.20
6	0.00	0.00	0.00
7(Lowest)	0.00	0.00	0.00

I/O channel Usage Blocking			
Priority	Direct	Pushthrough	Max Blocking
0(Highest)	0.00	0.00	0.00
1	0.00	0.00	0.00
2	6.00	6.00	6.00
3	0.00	6.00	6.00
4	2.00	2.00	2.00
5	0.00	2.00	2.00
6	0.00	2.00	2.00
7(Lowest)	0.00	0.00	0.00

Disk Usage Blocking			
Priority	Direct	Pushthrough	Max Blocking
0(Highest)	3.00	0.00	3.00
1	3.00	3.00	3.00
2	0.00	3.00	3.00
3	0.00	3.00	3.00
4	0.00	3.00	3.00
5	3.00	3.00	3.00
6	0.00	0.00	0.00
7(Lowest)	0.00	0.00	0.00

Figure 2: Max blocking for each resource at each priority level

Priority	Result Table Max Blocking	I/O channel Max Blocking	Disk Max Blocking	Total Max Blocking
0(Highest)	0.30	0.00	3.00	3.30
1	0.30	0.00	3.00	3.30
2	0.30	6.00	3.00	9.30
3	0.20	6.00	3.00	9.20
4	0.20	2.00	3.00	5.20
5	0.20	2.00	3.00	5.20
6	0.00	2.00	0.00	2.00
7(Lowest)	0.00	0.00	0.00	0.00

Figure 3: Total Max Blocking for each priority level

Code:

When ran using the obtained period, runtime, and max blocking values from the above data tables, the code to perform RMA, shown below in Figure 4, outputs the results shown in Figure 5. These results show that the RMA theorem is satisfied for the (k, l) values shown in Figure 5. Since all tasks are schedulable, the whole system is schedulable as well.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// compile command: gcc -o schedule lab8.c -Wall -g -lm

// number of tasks
#define TASKS 8
//Overhead value 156 ns
#define OH .153  // convert microseconds to milliseconds


int main(void){
  int i,j, k, l;
  //period for each priority (ms)
  float period[TASKS]   = {10.56, 40.96, 61.44, 100.00, 165.00, 285.00, 350.00, 700.00};
  //runTime for each priority (ms)
  float runTime[TASKS]  = { 1.30,  4.70, 9.00,  23.00,  38.30,  10.00,   3.00,   2.00};
  //total Max Blocking for each Priority (ms)
  float maxBlock[TASKS] = { 3.30,  3.30,  9.30,   9.20,   5.20,   5.20,   2.00,   0.00};

  // summation segment of the RMA theorem
  float sum = 0;
  // l*Tk value for the theorem comparison
  float comp = 0;
  // implement RMA theorem from slides
  for(i=1;i<=TASKS;i++){
    for(k=1;k<=i;k++){
      for(l=1;l<=floor(period[i-1]/period[k-1]);l++) {
        sum=0;
        for(j=1;j<=i-1;j++){
          sum += (runTime[j-1] + OH)*ceil(l*period[k-1]/period[j-1]);
        }
        // add run time and total maxBlocking
        sum += runTime[i-1] + maxBlock[i-1];
        // set comparison value
        comp = l*period[k-1];

        //print statement if passes comparison
        if(sum <= comp){
          printf("THEOREM PASSES FOR %d | (%d,%d)\n", i, k, l);
        }

      }
    }
  }
  return 1;
}
```

Figure 4: Code to perform RMA Theorem Calculation

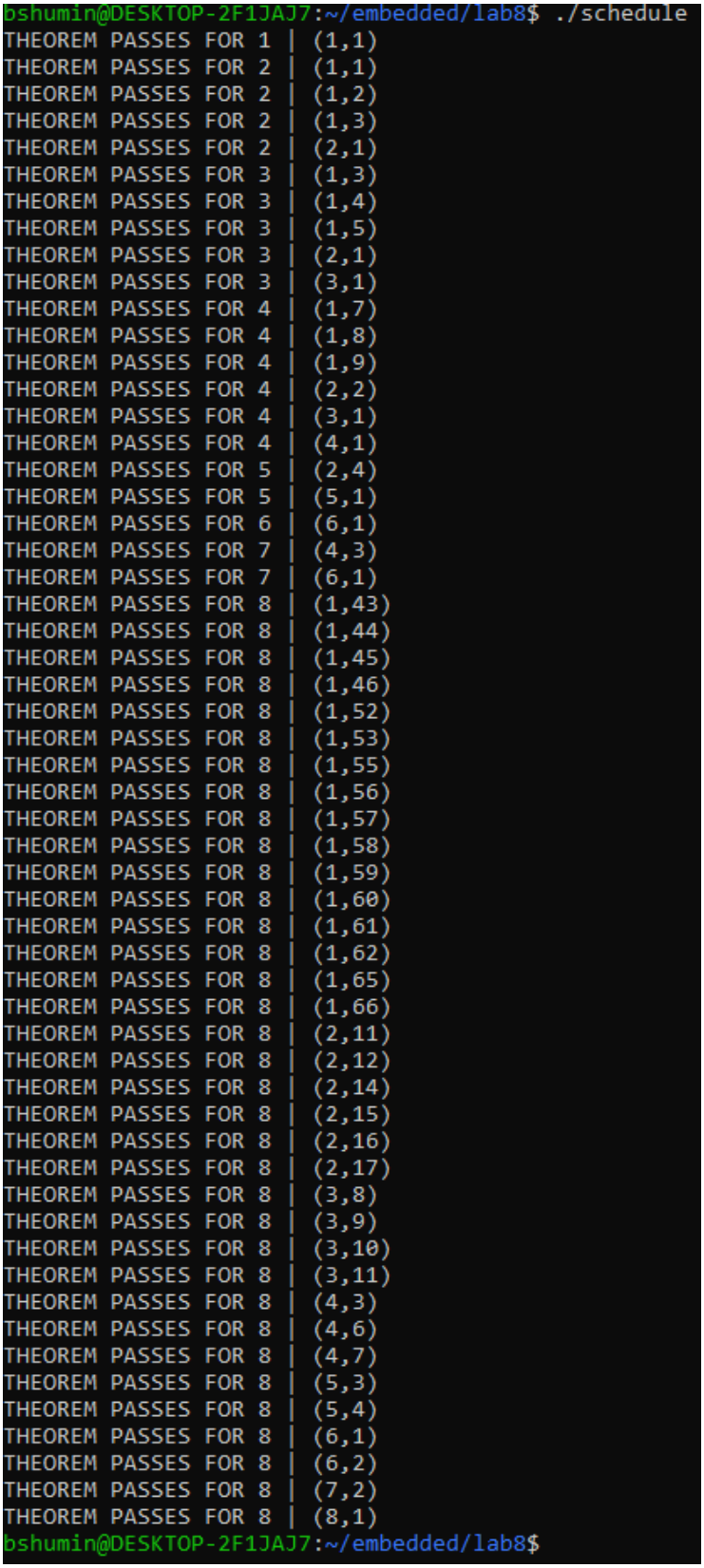

Figure 5: Output for the program