

ASSIGNMENT 5

Contents

SIMULATION ASSIGNMENT

Modify the python code in P_RM.py to use firstfit instead of the current algorithm. Please follow the steps in This document.

-Schedule the following task set on three processors using your modified algorithm.

PROGRAMMING PART

-Create a task "matrixtask" containing the following functionality

-Create a queue and send the content of (double **)c to the queue in matrixtask with before the vTaskDelay() call (hint: place the c variable in a struct). (More information Here).

-Create a reader task which reads the content of the queue in case there is something in the queue.

-In case the queue has some content it should save the data in a local (double **) variable.

-Print out the content of the (double **)c variable in case the content is updated. The data transferred from c should be a 10x10 matrix with the value 390 in each slot.

A screenshot of the execution

SIMULATION ASSIGNMENT

The assignment is to modify a real-time simulator to verify feasibility of a set of tasks.

Modify the python code in P_RM.py to use firstfit instead of the current algorithm.
Please follow the steps in This document.

-Hint: Instead of scheduling the task to the CPU with the lowest utilization chose the first one which has a lower utilization than $U_{rm}(x+1)$ where x is the already scheduled tasks on the CPU

-Hint2: have a look at the def packer(self) function in the file P_RM.py

```
def packer(self):
    # First Fit
    cpus = [[cpu, 0] for cpu in self.processors]
    numCPUs = len(cpus)
    print "CPU num: ", numCPUs
    taskNUM = [0] * numCPUs
    Urm = 0.0
    U = 0.0
    for task in self.task_list:
        #m = cpus[0][1]
        j = 0
        # Find the processor with the lowest Load.
        for i, c in enumerate(cpus):
            Urm = (taskNUM[i]+1.0) * ((pow(2.0, 1/(taskNUM[i]+1.0))) - 1.0)
            U = (c[1] + (task.wcet / task.period))
            print "CPU U = ", c[1]
            print "U after scheduling = ", U
            print "Urm = ", Urm
            if U < Urm:
                j = i
                break
        taskNUM[j] = taskNUM[j] + 1
        print "CPU scheduled = ", j
        print "Tasks = ", taskNUM

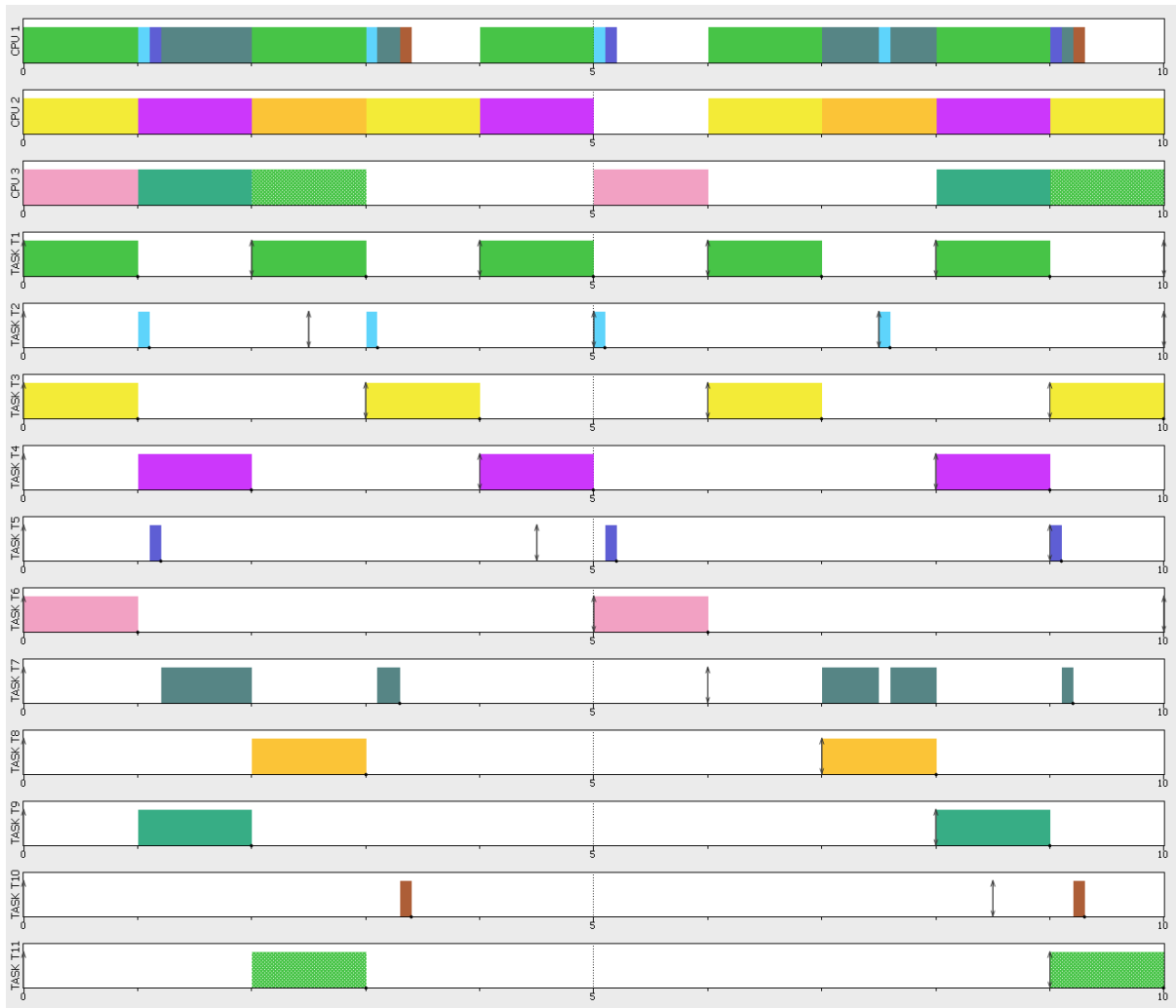
        # Affect it to the task.
        self.affect_task_to_processor(task, cpus[j][0])

        # Update utilization.
        cpus[j][1] += float(task.wcet) / task.period
    return True
```

-Schedule the following task set on three processors using your modified algorithm.

T1(2,1) T2(2.5,0.1) T3(3,1) T4(4,1) T5(4.5,0.1) T6(5,1) T7(6,1) T8(7,1) T9(8,1) T10(8.5,0.1) T11(9,1)

General Scheduler Processors Tasks										
Id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	
1	TASK T1	Periodic	<input checked="" type="checkbox"/> Yes	0.0	2.0	-	2.0	1.0		
2	TASK T2	Periodic	<input checked="" type="checkbox"/> Yes	0.0	2.5	-	2.5	0.1		
3	TASK T3	Periodic	<input checked="" type="checkbox"/> Yes	0.0	3.0	-	3.0	1.0		
4	TASK T4	Periodic	<input checked="" type="checkbox"/> Yes	0.0	4.0	-	4.0	1.0		
5	TASK T5	Periodic	<input checked="" type="checkbox"/> Yes	0.0	4.5	-	4.5	0.1		
6	TASK T6	Periodic	<input checked="" type="checkbox"/> Yes	0.0	5.0	-	5.0	1.0		
7	TASK T7	Periodic	<input checked="" type="checkbox"/> Yes	0.0	6.0	-	6.0	1.0		
8	TASK T8	Periodic	<input checked="" type="checkbox"/> Yes	0.0	7.0	-	7.0	1.0		
9	TASK T9	Periodic	<input checked="" type="checkbox"/> Yes	0.0	8.0	-	8.0	1.0		
10	TASK T10	Periodic	<input checked="" type="checkbox"/> Yes	0.0	8.5	-	8.5	0.1		
11	TASK T11	Periodic	<input checked="" type="checkbox"/> Yes	0.0	9.0	-	9.0	1.0		



PROGRAMMING PART

```
#define SIZE 10
#define ROW SIZE
#define COL SIZE
static boolean matrix_running = FALSE;
static boolean ap_running = FALSE;
static int matrix_period = 0;
int volatile size_2 = 0;
typedef struct Cmatrix_ {
    double **c;
} Cmatrix;
;
QueueHandle_t Qhandle;
```

- Create a task "matrixtask" containing the following functionality
- Create a queue and send the content of (double **)c to the queue in matrixtask with before the vTaskDelay() call (hint: place the c variable in a struct). ([More information Here](#)).

```

static void matrix_task() {
    int i;
    double **a = (double**) pvPortMalloc(ROW * sizeof(double*));
    for (i = 0; i < ROW; i++)
        a[i] = (double*) pvPortMalloc(COL * sizeof(double));
    double **b = (double**) pvPortMalloc(ROW * sizeof(double*));
    for (i = 0; i < ROW; i++)
        b[i] = (double*) pvPortMalloc(COL * sizeof(double));
    double **c = (double**) pvPortMalloc(ROW * sizeof(double*));
    for (i = 0; i < ROW; i++)
        c[i] = (double*) pvPortMalloc(COL * sizeof(double));
    double sum = 0.0;
    int j, k, l;
    for (i = 0; i < SIZE; i++) {
        for (j = 0; j < SIZE; j++) {
            a[i][j] = 1.5;
            b[i][j] = 2.6;
        }
    }

    while (1) {
        matrix_running = TRUE;
        matrix_period = 0;
        long simulationdelay;
        for (simulationdelay = 0; simulationdelay < 1000000000; simulationdelay++) ;
        for (i = 0; i < SIZE; i++) {
            for (j = 0; j < SIZE; j++) {
                c[i][j] = 0.0;
            }
        }
        for (i = 0; i < SIZE; i++) {
            for (j = 0; j < SIZE; j++) {
                sum = 0.0;
                for (k = 0; k < SIZE; k++) {
                    for (l = 0; l < 10; l++) {
                        sum = sum + a[i][k] * b[k][j];
                    }
                }
                c[i][j] = sum;
            }
        }
        vTaskDelay(100);
        matrix_running = FALSE;
        Cmatrix dataSent;
    }
}

```

```

fflush(stdout);
dataSent.c = (double**) pvPortMalloc(ROW * sizeof(double*));
for (i = 0; i < ROW; i++) {
    //printf("sent");
    fflush(stdout);

    dataSent.c[i] = (double*) pvPortMalloc(COL * sizeof(double));
    //printf("sent");
    fflush(stdout);

    for (j = 0; j < SIZE; j++) {
        dataSent.c[i][j] = c[i][j];
    }
}
xQueueSend(Qhandle, &dataSent, (TickType_t ) 0);
size_2 = uxQueueMessagesWaiting(Qhandle);
printf("matrix_period %d _ q size %d __\n",
        matrix_period * portTICK_PERIOD_MS,
        size_2);
fflush(stdout);    } }

```

-Create a reader task which reads the content of the queue in case there is something in the queue.

-In case the queue has some content it should save the data in a local (double **) variable.

-Print out the content of the (double **)c variable in case the content is updated. The data transferred from c should be a 10x10 matrix with the value 390 in each slot.

```

int volatile size_1 = 0;

static void read_task() {
    while (1) {
        if (size_1 >= size_2 && size_1 > 0) {
            //read
            Cmatrix result;
            xQueueReceive(Qhandle, &result, 0);
            printf("read\n");
            fflush(stdout);
            for (int i = 0; i < ROW; i++) {
                for (int j = 0; j < SIZE; j++) {
                    printf("c[%d][%d]= %f ", i, j, result.c[i][j]);

                }
                printf("\n");
                fflush(stdout);
            }

        }

        size_1 = uxQueueMessagesWaiting(Qhandle);

        vTaskDelay(100);
    }
}

xTaskHandle matrix_handle;
xTaskHandle read_task_handle;

```

A screenshot of the execution

The screenshot shows the Eclipse IDE interface with the following components:

- Top Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, AVR, Run, Window, Help.
- Project Explorer:** Shows the project structure with 'RTOSDemo' and a sub-project 'T'.
- Code Editor:** Displays the file 'main.c' with the following code:

```
370 //ap_period++;
371
```
- Console:** Shows the output of the program, which is a series of memory addresses and their corresponding values, organized in a grid-like structure. The output is as follows:

```
RTOSDemo.exe [C/C++ Application]
matrix_period 1163 _q size 1 _
read
c[0][0]= 390.000000 c[0][1]= 390.000000 c[0][2]= 390.000000 c[0][3]= 390.000000 c[0][4]= 390.000000 c[0][5]= 390.000000 c[0][6]= 390.000000 c[0][7]= 390.000000 c[0][8]= 390.000000 c[0][9]= 390.000000
c[1][0]= 390.000000 c[1][1]= 390.000000 c[1][2]= 390.000000 c[1][3]= 390.000000 c[1][4]= 390.000000 c[1][5]= 390.000000 c[1][6]= 390.000000 c[1][7]= 390.000000 c[1][8]= 390.000000 c[1][9]= 390.000000
c[2][0]= 390.000000 c[2][1]= 390.000000 c[2][2]= 390.000000 c[2][3]= 390.000000 c[2][4]= 390.000000 c[2][5]= 390.000000 c[2][6]= 390.000000 c[2][7]= 390.000000 c[2][8]= 390.000000 c[2][9]= 390.000000
c[3][0]= 390.000000 c[3][1]= 390.000000 c[3][2]= 390.000000 c[3][3]= 390.000000 c[3][4]= 390.000000 c[3][5]= 390.000000 c[3][6]= 390.000000 c[3][7]= 390.000000 c[3][8]= 390.000000 c[3][9]= 390.000000
c[4][0]= 390.000000 c[4][1]= 390.000000 c[4][2]= 390.000000 c[4][3]= 390.000000 c[4][4]= 390.000000 c[4][5]= 390.000000 c[4][6]= 390.000000 c[4][7]= 390.000000 c[4][8]= 390.000000 c[4][9]= 390.000000
c[5][0]= 390.000000 c[5][1]= 390.000000 c[5][2]= 390.000000 c[5][3]= 390.000000 c[5][4]= 390.000000 c[5][5]= 390.000000 c[5][6]= 390.000000 c[5][7]= 390.000000 c[5][8]= 390.000000 c[5][9]= 390.000000
c[6][0]= 390.000000 c[6][1]= 390.000000 c[6][2]= 390.000000 c[6][3]= 390.000000 c[6][4]= 390.000000 c[6][5]= 390.000000 c[6][6]= 390.000000 c[6][7]= 390.000000 c[6][8]= 390.000000 c[6][9]= 390.000000
c[7][0]= 390.000000 c[7][1]= 390.000000 c[7][2]= 390.000000 c[7][3]= 390.000000 c[7][4]= 390.000000 c[7][5]= 390.000000 c[7][6]= 390.000000 c[7][7]= 390.000000 c[7][8]= 390.000000 c[7][9]= 390.000000
c[8][0]= 390.000000 c[8][1]= 390.000000 c[8][2]= 390.000000 c[8][3]= 390.000000 c[8][4]= 390.000000 c[8][5]= 390.000000 c[8][6]= 390.000000 c[8][7]= 390.000000 c[8][8]= 390.000000 c[8][9]= 390.000000
c[9][0]= 390.000000 c[9][1]= 390.000000 c[9][2]= 390.000000 c[9][3]= 390.000000 c[9][4]= 390.000000 c[9][5]= 390.000000 c[9][6]= 390.000000 c[9][7]= 390.000000 c[9][8]= 390.000000 c[9][9]= 390.000000
matrix_period 1147 _q size 1 _
read
c[0][0]= 390.000000 c[0][1]= 390.000000 c[0][2]= 390.000000 c[0][3]= 390.000000 c[0][4]= 390.000000 c[0][5]= 390.000000 c[0][6]= 390.000000 c[0][7]= 390.000000 c[0][8]= 390.000000 c[0][9]= 390.000000
c[1][0]= 390.000000 c[1][1]= 390.000000 c[1][2]= 390.000000 c[1][3]= 390.000000 c[1][4]= 390.000000 c[1][5]= 390.000000 c[1][6]= 390.000000 c[1][7]= 390.000000 c[1][8]= 390.000000 c[1][9]= 390.000000
c[2][0]= 390.000000 c[2][1]= 390.000000 c[2][2]= 390.000000 c[2][3]= 390.000000 c[2][4]= 390.000000 c[2][5]= 390.000000 c[2][6]= 390.000000 c[2][7]= 390.000000 c[2][8]= 390.000000 c[2][9]= 390.000000
c[3][0]= 390.000000 c[3][1]= 390.000000 c[3][2]= 390.000000 c[3][3]= 390.000000 c[3][4]= 390.000000 c[3][5]= 390.000000 c[3][6]= 390.000000 c[3][7]= 390.000000 c[3][8]= 390.000000 c[3][9]= 390.000000
c[4][0]= 390.000000 c[4][1]= 390.000000 c[4][2]= 390.000000 c[4][3]= 390.000000 c[4][4]= 390.000000 c[4][5]= 390.000000 c[4][6]= 390.000000 c[4][7]= 390.000000 c[4][8]= 390.000000 c[4][9]= 390.000000
c[5][0]= 390.000000 c[5][1]= 390.000000 c[5][2]= 390.000000 c[5][3]= 390.000000 c[5][4]= 390.000000 c[5][5]= 390.000000 c[5][6]= 390.000000 c[5][7]= 390.000000 c[5][8]= 390.000000 c[5][9]= 390.000000
c[6][0]= 390.000000 c[6][1]= 390.000000 c[6][2]= 390.000000 c[6][3]= 390.000000 c[6][4]= 390.000000 c[6][5]= 390.000000 c[6][6]= 390.000000 c[6][7]= 390.000000 c[6][8]= 390.000000 c[6][9]= 390.000000
c[7][0]= 390.000000 c[7][1]= 390.000000 c[7][2]= 390.000000 c[7][3]= 390.000000 c[7][4]= 390.000000 c[7][5]= 390.000000 c[7][6]= 390.000000 c[7][7]= 390.000000 c[7][8]= 390.000000 c[7][9]= 390.000000
c[8][0]= 390.000000 c[8][1]= 390.000000 c[8][2]= 390.000000 c[8][3]= 390.000000 c[8][4]= 390.000000 c[8][5]= 390.000000 c[8][6]= 390.000000 c[8][7]= 390.000000 c[8][8]= 390.000000 c[8][9]= 390.000000
c[9][0]= 390.000000 c[9][1]= 390.000000 c[9][2]= 390.000000 c[9][3]= 390.000000 c[9][4]= 390.000000 c[9][5]= 390.000000 c[9][6]= 390.000000 c[9][7]= 390.000000 c[9][8]= 390.000000 c[9][9]= 390.000000
matrix_period 1124 _q size 1 _
read
c[0][0]= 390.000000 c[0][1]= 390.000000 c[0][2]= 390.000000 c[0][3]= 390.000000 c[0][4]= 390.000000 c[0][5]= 390.000000 c[0][6]= 390.000000 c[0][7]= 390.000000 c[0][8]= 390.000000 c[0][9]= 390.000000
c[1][0]= 390.000000 c[1][1]= 390.000000 c[1][2]= 390.000000 c[1][3]= 390.000000 c[1][4]= 390.000000 c[1][5]= 390.000000 c[1][6]= 390.000000 c[1][7]= 390.000000 c[1][8]= 390.000000 c[1][9]= 390.000000
c[2][0]= 390.000000 c[2][1]= 390.000000 c[2][2]= 390.000000 c[2][3]= 390.000000 c[2][4]= 390.000000 c[2][5]= 390.000000 c[2][6]= 390.000000 c[2][7]= 390.000000 c[2][8]= 390.000000 c[2][9]= 390.000000
c[3][0]= 390.000000 c[3][1]= 390.000000 c[3][2]= 390.000000 c[3][3]= 390.000000 c[3][4]= 390.000000 c[3][5]= 390.000000 c[3][6]= 390.000000 c[3][7]= 390.000000 c[3][8]= 390.000000 c[3][9]= 390.000000
```