Lab Assignment - 4

Proving that µcos-II uses a preemptive kernel

Name : Dumvesh Naresh Patil
PRN : 2019BTEEN00035
Batch : EN-1

4.3.1 Preemptive & non-preemptive kernel.

1) Write what is a kernel & How it differs from os ?

⇒ Kernel is the central core part of the os. context switching is provided by the kernel. It is also a system program. It is part of os which convert user commands into machine language.

OS	Kernel
• It is a system software	• It is the part of the os
• Interface between user & hardware	• Interface betⁿ application & hardware

2) Write brief information of Preemptive & non-preemptive kernel.

⇒ Preemptive kernel : Preemptive kernel schedules the task as per the priority of the tasks. When higher priority task arrives it preempts the lower priority task & schedules the higher priority task.

Non-preemptive kernel : In preemptive kernel, it forcefully stop the lower priority task but in non-preemptive one it does not stop forcefully. First task will execute completely then it schedules the next task only.

3) Write advantage of Preemptive kernel over non-preemptive kernel.
⇒ (i) Pseudo parallel execution.
(ii) Effective CPU usage.
(iii) More responsive.

4.3.2 Logic of providing RTOS to be preemptive.

Write the logic to prove ucos preempts the low priority task if high priority task is ready.
⇒ When we run the program, CPU first schedule the higher priority ready task. As soon as the higher priority ready task than current running task arrives it preempts it & gives it CPU. We can observe the waveforms of Task0 & Task1. We can prove it using the myDelay() function.

4.3.3 The program

1) Give reasoning whether we can use the program in the previous experiments as it proves that ucos-II is preemptive.

Page No. Date: YOUVA

⇒ Yes, we can use the same program. But to prove that uC/OS-II is preemptive we need to ~~so~~ do some modifications.
We need to write user defined function so that ~~we need to write user~~ defined CPU will get engaged in executing that & observe whether lower priority task preempts or not.

2) Write steps in creating the modified program
⇒ 1) Declare & define the myDelay() function above main() function which is user defined function
2) Call the myDelay() function in the lower priority task
3) Observe the waveforms in the logic analyzer window.

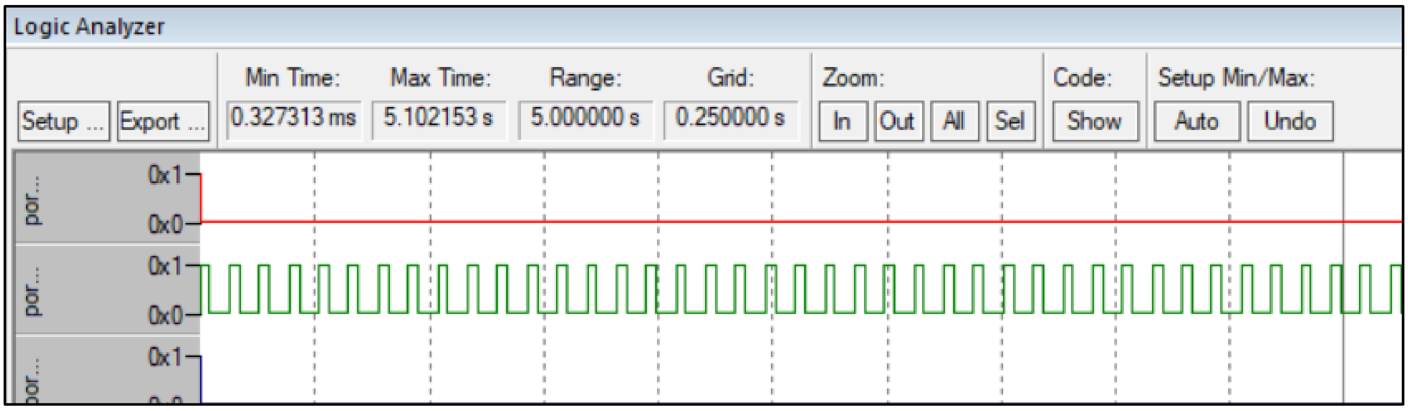
Q Write program with proper indenting, syntax & also taking care of ~~car~~ case sensitivity of C language

```
#include "config.h"
#include "stdlib.h"
#include <stdio.h>

#define       TaskStkLengh       64                              //Define the Task0 stack length

OS_STK        TaskStk0 [TaskStkLengh];                  //Define the Task stack
OS_STK        TaskStk1 [TaskStkLengh];                  //Define the Task stack

void    Task0(void *pdata);
void    Task1(void *pdata);


void myDelay()
{
        unsigned int i;
        for(i=0;i<65000;i++);
}

char buffer[25];

int main (void)
{
        LED_init();

        TargetInit();
        OSInit ();

        OSTaskCreate (Task0,(void *)0, &TaskStk0[TaskStkLengh - 1], 6);
        OSTaskCreate (Task1,(void *)0, &TaskStk1[TaskStkLengh - 1], 7);

        OSStart();
        return 0;

}
void Task0    (void *pdata)
{
        pdata = pdata;                                                  /* Dummy data */

        while(1)
        {
                LED_on(0);   // All LEDs on
                OSTimeDly(3);

                LED_off(0);  // All LEDs off
                OSTimeDly(3);


        }
}
```

```
void Task1    (void *pdata)
{

        pdata = pdata;                                                          /* Dummy data */

        while(1)
        {
                LED_on(1);   // All LEDs on

                myDelay();
                myDelay();


                //OSTimeDly(3);

                LED_off(1);   // All LEDs off

                myDelay();
                myDelay();

        }
}
```

4.3 Observations:

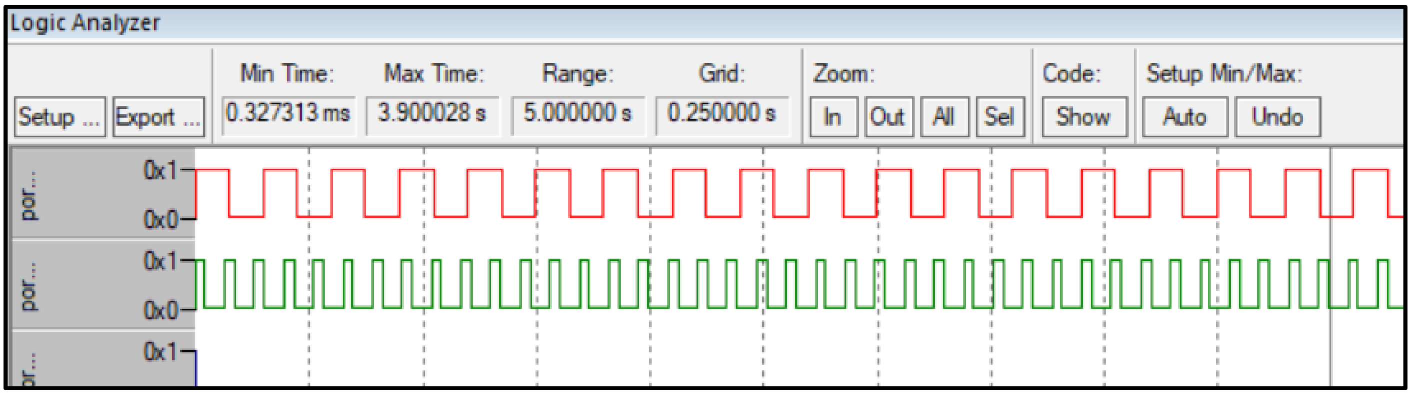
Take screenshots of the simulated output for myDelay function used in HP task



Min Time: 0.327313 ms
Max Time: 5.102153 s
Range: 5.000000 s
Grid: 0.250000 s

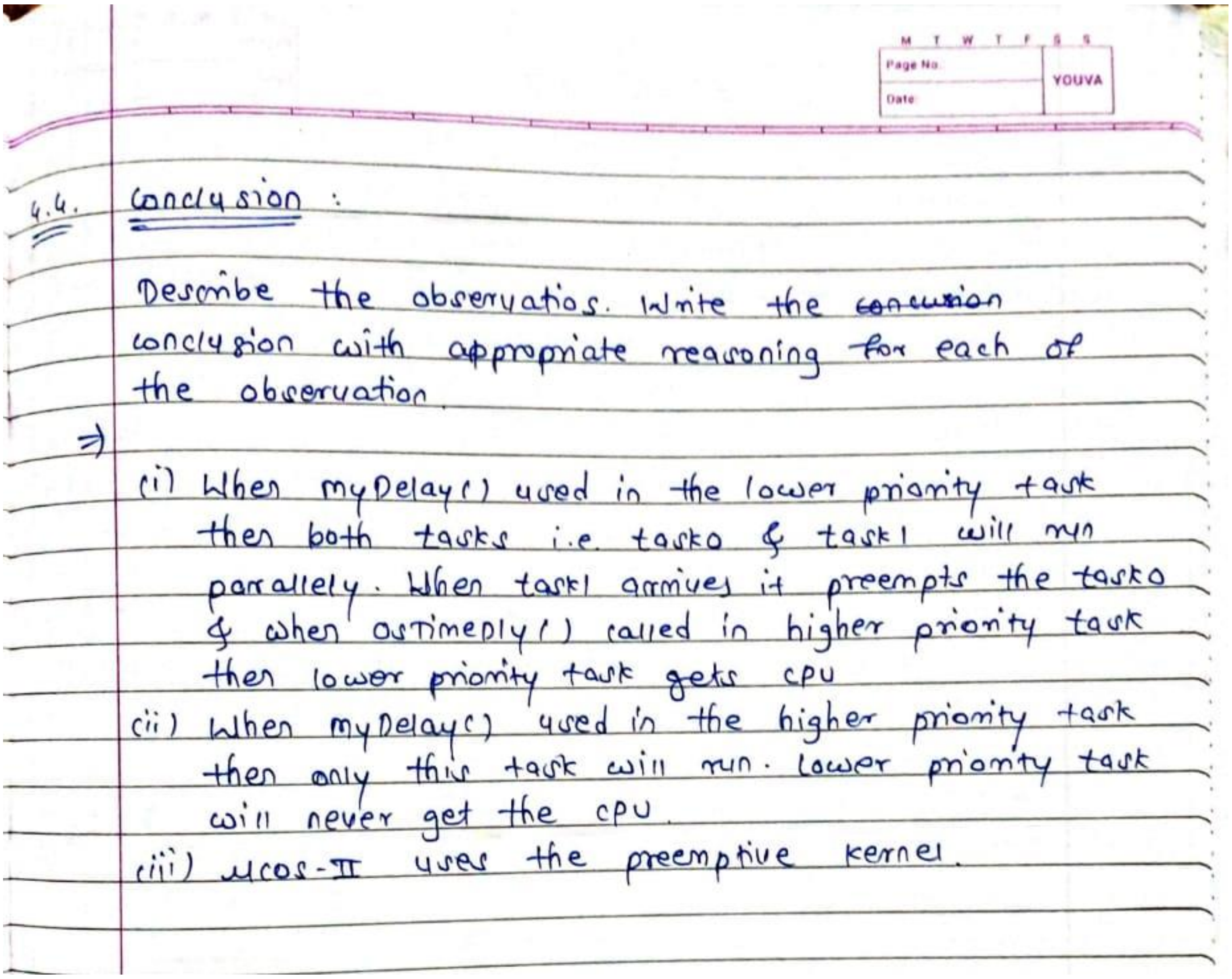
Only higher priority task is running. Lower priority task will never get the CPU.

Take screenshots of the simulated output for myDelay function used in LP task



Min Time: 0.327313 ms
Max Time: 3.900028 s
Range: 5.000000 s
Grid: 0.250000 s

Both high and low priority tasks are executing.



M T W T F S S
Page No.
Date
YOUVA

4.4. Conclusion :

Describe the observatios. Write the ~~concusion~~ conclusion with appropriate reasoning for each of the observation.

⇒

(i) When myDelay() used in the lower priority task then both tasks i.e. task0 & task1 will run parallely. When task1 arrives it preempts the task0 & when OSTimeDly() called in higher priority task then lower priority task gets cpu

(ii) When myDelay() used in the higher priority task then only this task will run. Lower priority task will never get the cpu.

(iii) μCOS-II uses the preemptive kernel.