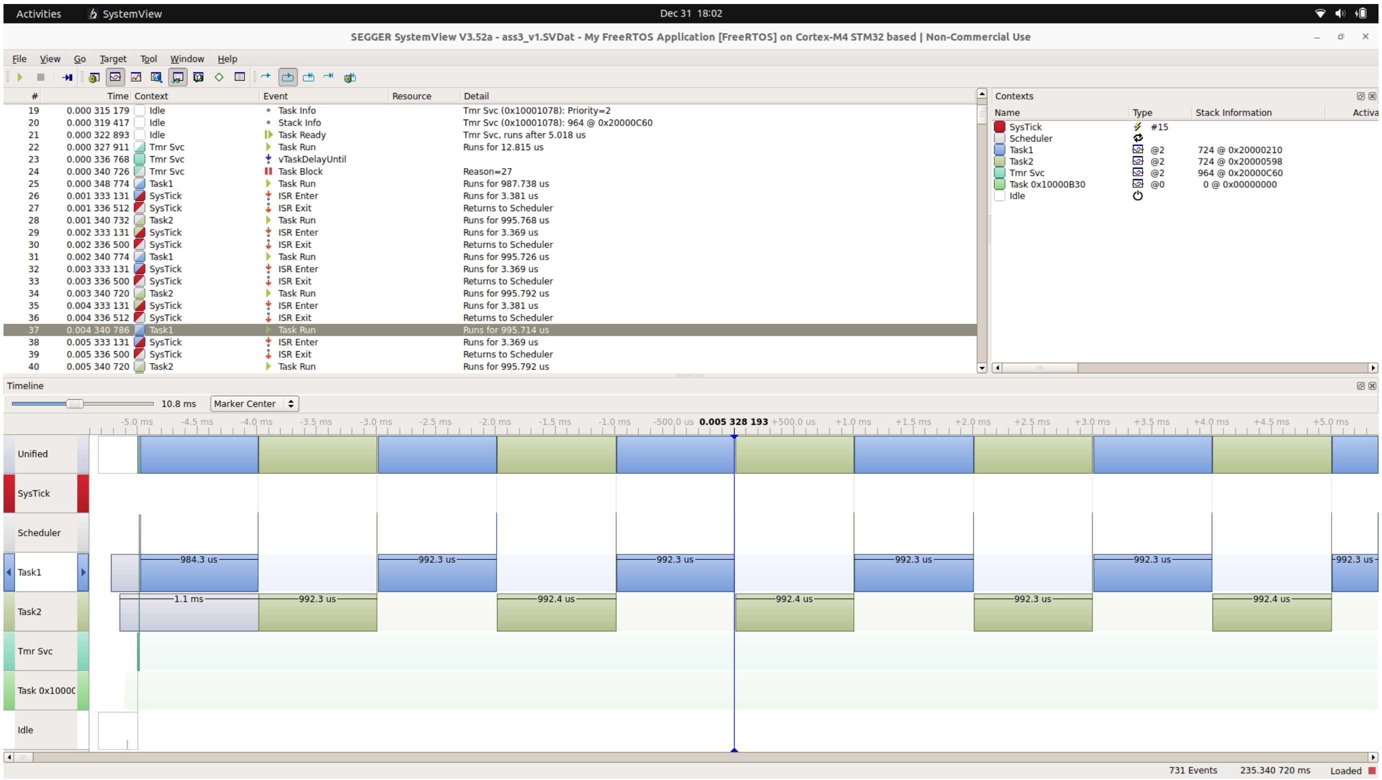
Assignment 3

Scenario 1



SEGGER SystemView V3.52a - ass3_v1.SVDat - My FreeRTOS Application [FreeRTOS] on Cortex-M4 STM32 based | Non-Commercial Use

#	Time	Context	Event	Resource	Detail
19	0.000 315 179	Idle	Task Info		Tmr Svc (0x10001078): Priority=2
20	0.000 319 417	Idle	Stack Info		Tmr Svc (0x10001078): 964 @ 0x20000C60
21	0.000 322 893	Idle	Task Ready		Tmr Svc, runs after 5.018 us
22	0.000 327 911	Tmr Svc	Task Run		Runs for 12.815 us
23	0.000 336 768	Tmr Svc	vTaskDelayUntil		
24	0.000 340 726	Tmr Svc	Task Block		Reason=27
25	0.000 348 774	Task1	Task Run		Runs for 987.738 us
26	0.001 333 131	SysTick	ISR Enter		Runs for 3.381 us
27	0.001 336 512	SysTick	ISR Exit		Returns to Scheduler
28	0.001 340 732	Task2	Task Run		Runs for 995.768 us
29	0.002 333 131	SysTick	ISR Enter		Runs for 3.369 us
30	0.002 336 500	SysTick	ISR Exit		Returns to Scheduler
31	0.002 340 774	Task1	Task Run		Runs for 995.726 us
32	0.003 333 131	SysTick	ISR Enter		Runs for 3.369 us
33	0.003 336 500	SysTick	ISR Exit		Returns to Scheduler
34	0.003 340 720	Task2	Task Run		Runs for 995.792 us
35	0.004 333 131	SysTick	ISR Enter		Runs for 3.381 us
36	0.004 336 512	SysTick	ISR Exit		Returns to Scheduler
37	0.004 340 786	Task1	Task Run		Runs for 995.714 us
38	0.005 333 131	SysTick	ISR Enter		Runs for 3.369 us
39	0.005 336 500	SysTick	ISR Exit		Returns to Scheduler
40	0.005 340 720	Task2	Task Run		Runs for 995.792 us

Contexts

Name	Type	Stack Information	Activa
SysTick	#15		
Scheduler			
Task1	@2	724 @ 0x20000210	
Task2	@2	724 @ 0x20000598	
Tmr Svc	@2	964 @ 0x20000C60	
Task 0x10000B30	@0	0 @ 0x00000000	
Idle			

Timeline

10.8 ms Marker Center

731 Events 235.340 720 ms Loaded

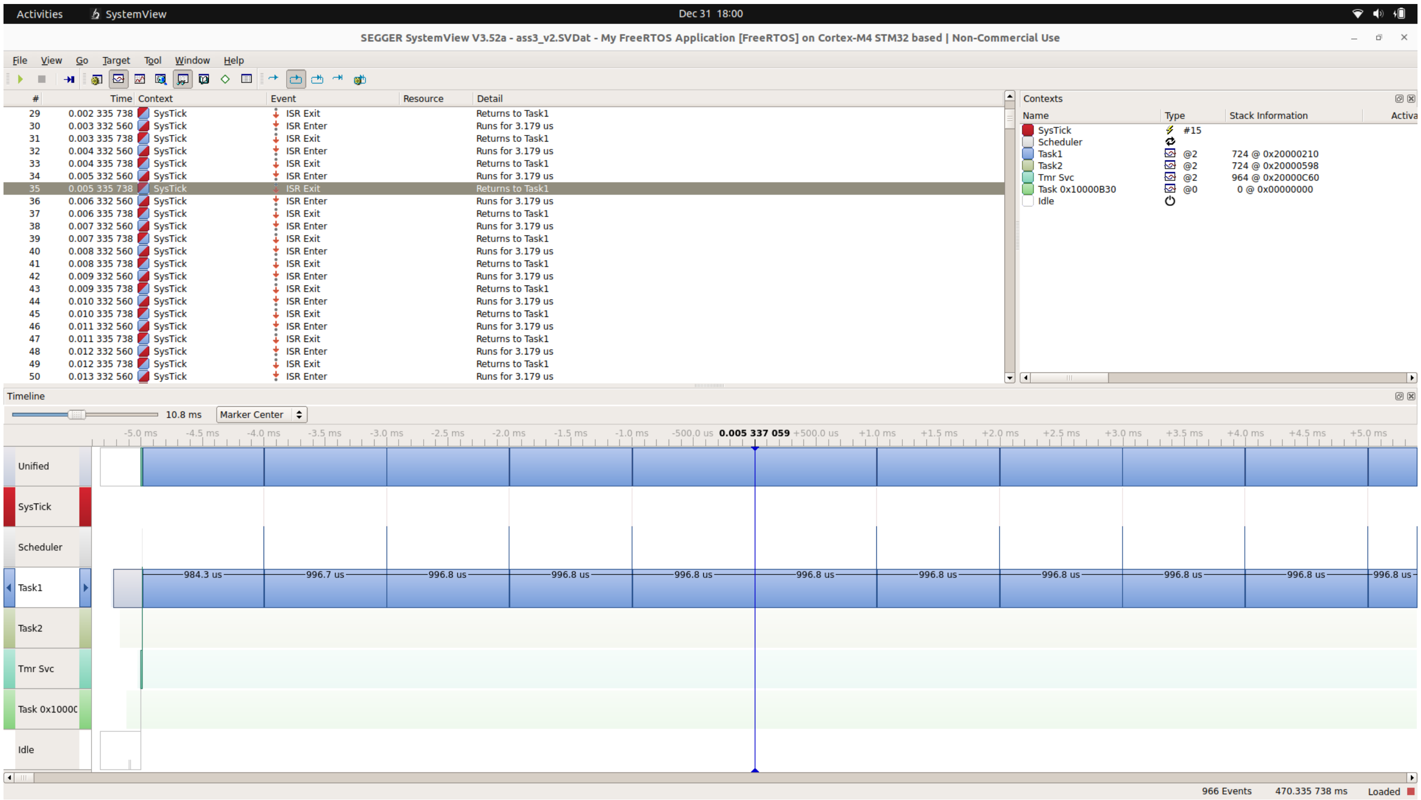
We know that, the both tasks have the same priorities. i.e. 2.

If we enable preemption from the FreeRTOSConfig.h file. We see the above result.

If we observe closely, task 1 and task 2 are in ready start, but due to priority being the same, the task which got created first, will execute first.
After execution of task 1 we can see that, the systick invokes the scheduler, and since the preemption is enabled we see that the task 1 is suspended and task 2 enters the ready state and thus starts executing.

This process continues infinitely, this type of scheduling is known as Pre-Emptive Scheduling.

Scenario 2



SEGGER SystemView V3.52a - ass3_v2.SVDat - My FreeRTOS Application [FreeRTOS] on Cortex-M4 STM32 based | Non-Commercial Use

#	Time	Context	Event	Resource	Detail
29	0.002 335 738	SysTick	ISR Exit		Returns to Task1
30	0.003 332 560	SysTick	ISR Enter		Runs for 3.179 us
31	0.003 335 738	SysTick	ISR Exit		Returns to Task1
32	0.004 332 560	SysTick	ISR Enter		Runs for 3.179 us
33	0.004 335 738	SysTick	ISR Exit		Returns to Task1
34	0.005 332 560	SysTick	ISR Enter		Runs for 3.179 us
35	0.005 335 738	SysTick	ISR Exit		Returns to Task1
36	0.006 332 560	SysTick	ISR Enter		Runs for 3.179 us
37	0.006 335 738	SysTick	ISR Exit		Returns to Task1
38	0.007 332 560	SysTick	ISR Enter		Runs for 3.179 us
39	0.007 335 738	SysTick	ISR Exit		Returns to Task1
40	0.008 332 560	SysTick	ISR Enter		Runs for 3.179 us
41	0.008 335 738	SysTick	ISR Exit		Returns to Task1
42	0.009 332 560	SysTick	ISR Enter		Runs for 3.179 us
43	0.009 335 738	SysTick	ISR Exit		Returns to Task1
44	0.010 332 560	SysTick	ISR Enter		Runs for 3.179 us
45	0.010 335 738	SysTick	ISR Exit		Returns to Task1
46	0.011 332 560	SysTick	ISR Enter		Runs for 3.179 us
47	0.011 335 738	SysTick	ISR Exit		Returns to Task1
48	0.012 332 560	SysTick	ISR Enter		Runs for 3.179 us
49	0.012 335 738	SysTick	ISR Exit		Returns to Task1
50	0.013 332 560	SysTick	ISR Enter		Runs for 3.179 us

Contexts

Name	Type	Stack Information	Activa
SysTick	#15		
Scheduler			
Task1	@2	724 @ 0x20000210	
Task2	@2	724 @ 0x20000598	
Tmr Svc	@2	964 @ 0x20000C60	
Task 0x10000B30	@0	0 @ 0x00000000	
Idle			

966 Events 470.335 738 ms Loaded

We know that, the both tasks have the same priorities. i.e. 2.

If we disable preemption from the FreeRTOSConfig.h file. We see the above result.

If we observe closely, task 1 and task 2 are in ready start, but due to priority being the same, the task which got created first, will execute first.
After execution of task 1 in first tick we can see that, the systick is invoked, but the scheduler is not invoked. This is due to the preemption not being enabled. Which causes the scheduling algorithm to go in cooperative scheduling.

The co-operative scheduling algorithm, never interrupts a running process to initiate a context switch from one task to another.