- thread priorities:

from <cmsis_os.h>

```
typedef enum osPriority {
        osPriorityIdle = -3,
        osPriorityLow = -2,
        osPriorityBelowNormal = -1,
        osPriorityNormal = 0,
        osPriorityAboveNormal = +1,
        osPriorityHigh = +2,
        osPriorityRealtime = +3,
        ...
} osPriority;
```

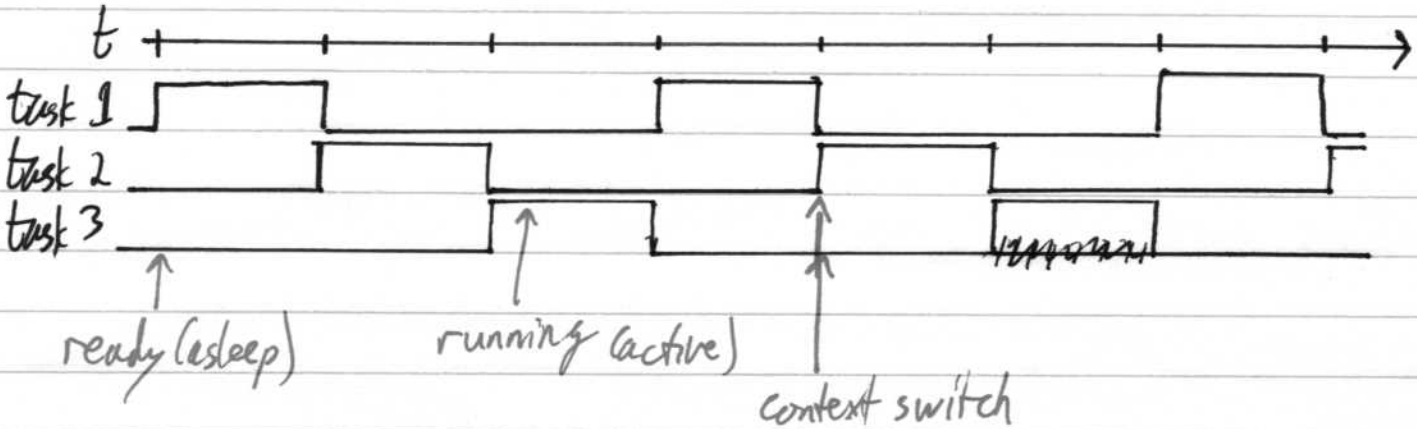
- the scheduler runs the threads that are ready and have highest priority
- when neither t1, t2, nor t3 are ready, the scheduler runs the os_idle_demon(): (=osPriorityIdle)

```
for(;;);   ← os_idle_demon() body
```

3) Concurrent Execution

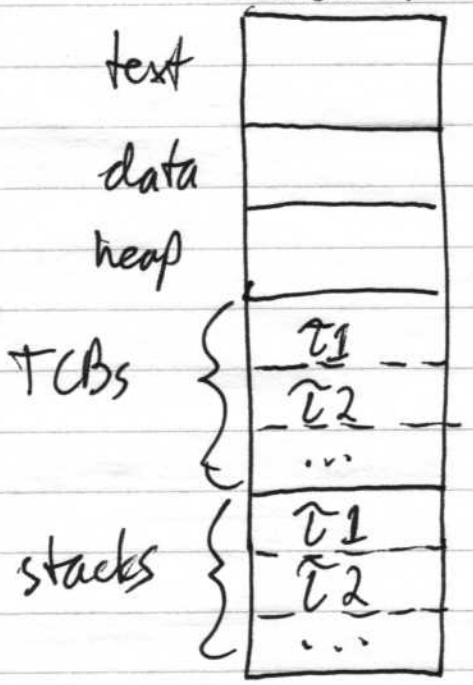
- task executions are interleaved on one (or more) processors with indeterminate ordering

  e.g. 1 processor, 3 tasks of equal priority, round-robin scheduling



- a real-time application consists of a set of concurrent tasks (e.g. "threads in CMSIS-RTOS)
  - all tasks share the text and data sections and heap
  - each task has its own stack and Task Control Block (TCB)

RTOS Memory Map

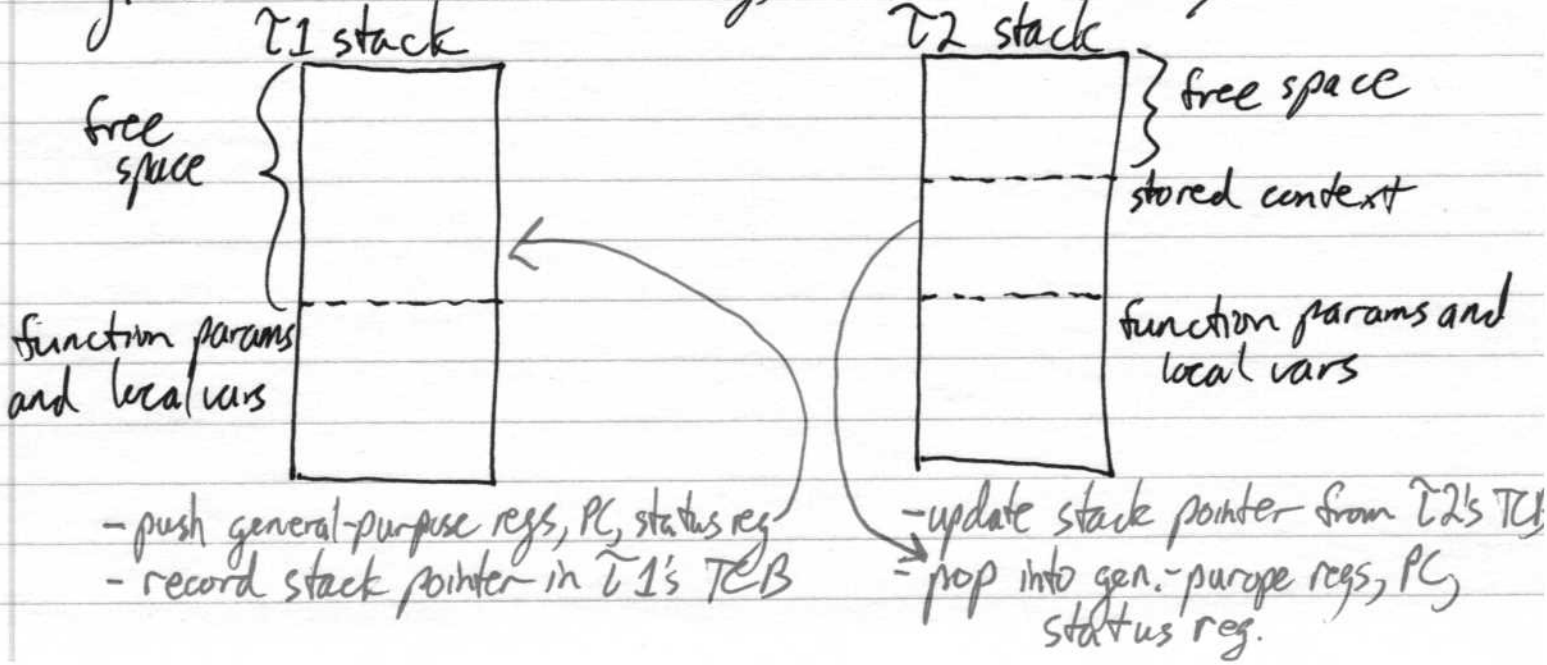


- the OS uses the TCB to manage a task
- TCB contents:
  - task ID
  - stack pointer
  - state = {Ready, Running, Waiting, Inactive}
  - priority
  - pointers to wait lists (linked lists)
  - delay time
  - event flags
  - ...

3.1) Context Switch

- to switch between running tasks the OS must:
  1. store the execution context of the running task
  2. restore the execution context of the ready task

e.g. switch from $\tau 1$ (running) to $\tau 2$ (ready)



4) Scheduling

[7.3]

4.1) Task States