Memory management

- No! (DO-178B)

- Standard C functions: malloc()/free()
  - » Typically maintain a linked list of blocks (allocated and free)
  - » (De)allocation involves complex search operations on a list -> *hard to predict run-time performance*
- RTOS Technique: Pools
  - » Pool: Memory area containing a fixed number of blocks of fixed size

Pools

- Trick: algorithms to manage memory are much simpler/predictable
- API: (ID: unique identifier for a pool)

  Initialization a pool:

  » int init_pool(uint ID, void* mem, uint bufSize, uint bufCount);

  Find and get a block (wait max. for timeOut ticks)

  » void* getbuf(uint ID, uint timeOut);

  Get a block immediately

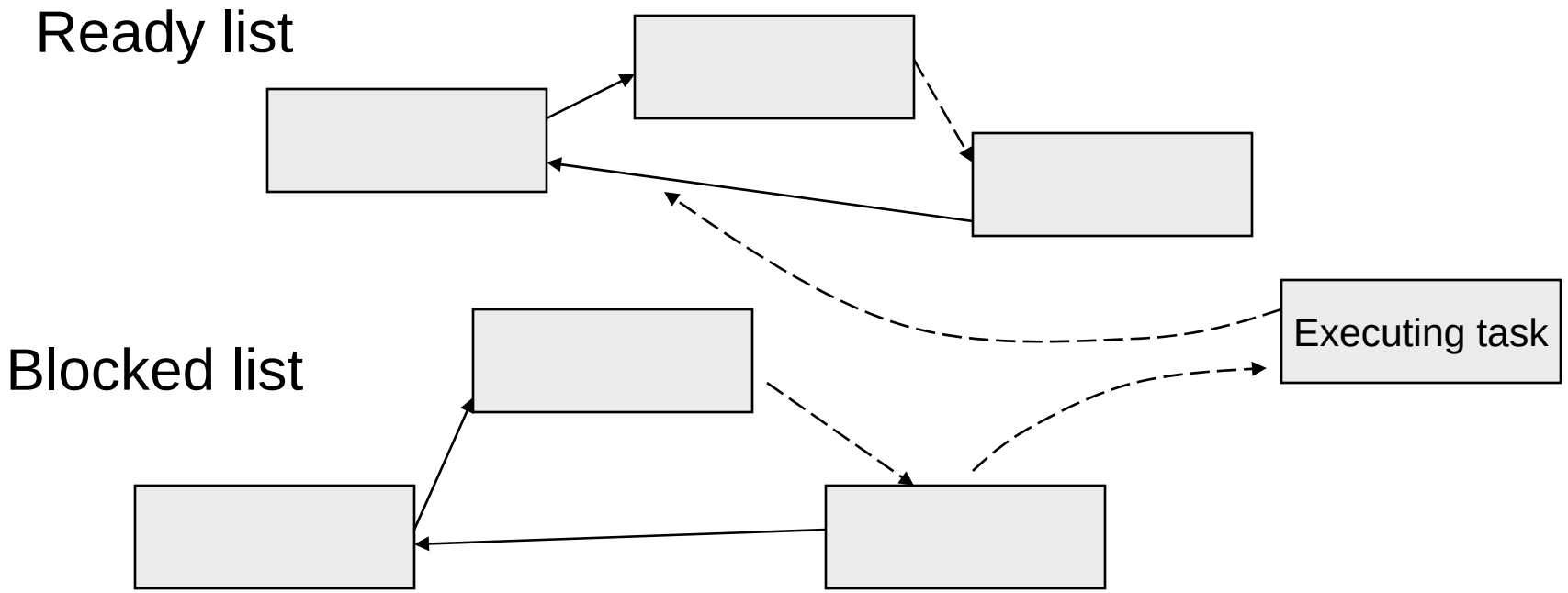
  » Void* reqbuf(uint ID);

  Return a block to the pool

  » void relbuf(uint ID, void* buffer);

Other memory management issues

- Good practice: Each task has its own stack
  - » TCB has a pointer to the task's stack
- Task control block management: 2 lists



Other memory management issues

- TCB management
  - » Exec. Task releases resource high-priority (blocked) task is waiting for
  - » Executing task placed on ready list
  - » High-priority task is removed from blocked list and starts execution
- Garbage collection:
  - » Automatic management of unused memory
  - » Mostly non-deterministic (although a few “real-time” approaches are known – with bounded response times)

ISRs in RTOS

Rule 1: ISR must not call an RTOS function that could block the caller.

- Cannot call semaphore P()
  - » RTOS would lock if the semaphore is already taken
- Cannot read from empty queues/mailboxes
- Cannot wait for events

Scenario:

Task runs -> ISR is called -> ISR locks up, thus Task locks -> potential deadlock.

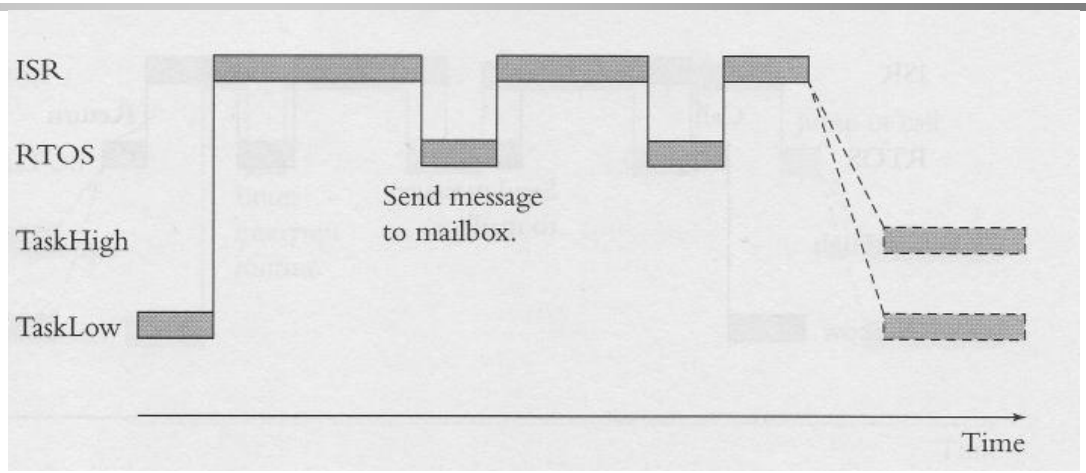
ISRs in RTOS

Rule 2: ISR must not call any RTOS function that could lead to task switch, unless the RTOS is told that an ISR, not a task, is executing.

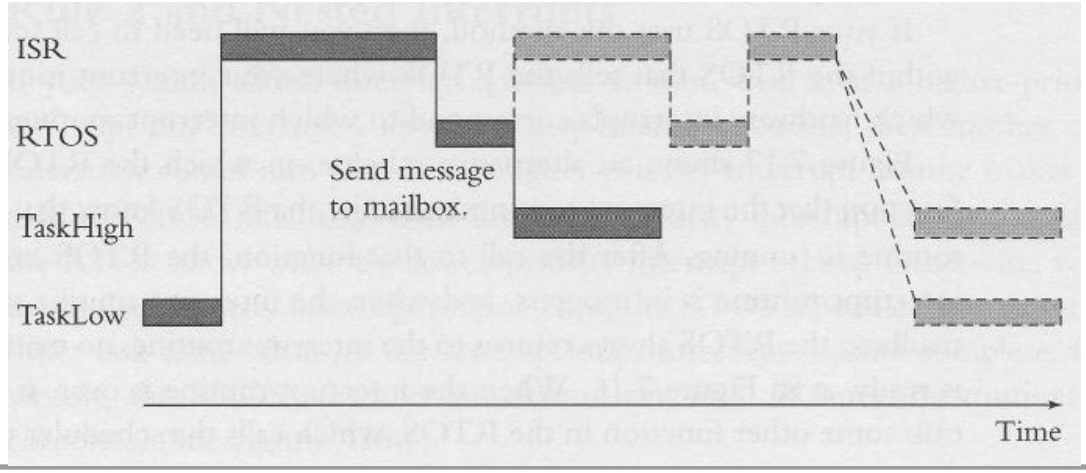
- If the RTOS is called to enqueue data into a message queue, RTOS might schedule a higher priority task.
- Solution: RTOS must be informed that an ISR is running (and not a task), s.t. re-scheduling will not happen.

ISRs in RTOS

What you expect:



What really happens:



Simon '99