Assignment 3 – Port uCOS to the STM32L475 SoC

EMBSYS 320 Winter 2021

Due: in one week

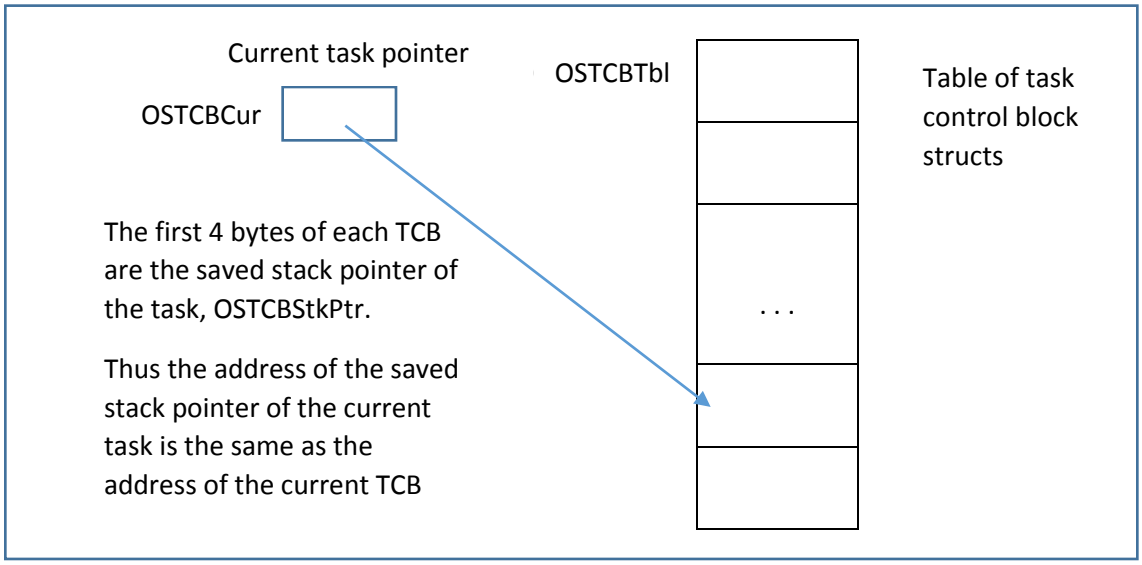
The goal of this assignment is to complete the port of uCOS to our board. Much of the port is already complete (the easy part!). Your job is to implement the assembly language code for critical section entry and exit, and context switch.

1. Download and unzip the uCOSPortHW project contained in the zip file: uCOSPortHW.zip
2. Open the uCOSPortHW.eww workspace in the EWARM IDE.
3. Make sure the uCOSPortHW project builds.
4. Launch TeraTerm (Baud rate: 38400)
5. Build and upload the project and start it running.
6. Notice that it prints some initial messages on the UART but doesn't get very far before hanging.
7. Break into the program.
8. Notice it is stuck in a tight loop at the label OSStartHang in the file os_cpu_a.asm.
9. Your job is to read the explanatory comments and add missing code to file os_cpu_a.asm to achieve the following:
    a. **Implement OS_CPU_SR_Save()**. This function gets called throughout uCOS when entering a critical section. It should capture the current interrupt enable/disable status given by register PRIMASK, then disable interrupts, then return the captured status as the function's return value.
    b. **Implement OS_CPU_SR_Restore().** This function gets called throughout uCOS when exiting a critical section. It takes an argument consisting of the interrupt enable/disable status and copies it to the PRIMASK register, then returns.
    c. **Implement ContextSwitch().** This code handles the PendSV exception. PendSV is only triggered by software in uCOS and only executes when no other interrupt is active. UCOS triggers PendSV by calling either OSCtxSw() or OSIntCtxSw() if it determines that a context switch is necessary.

10. Clean your completed project by removing the Debug folder and zipping it into a file named uCOSPortHW_<yourName>.zip
    e.g. uCOSPortHW_janedoe.zip
11. Submit your zip file by the due date.

Hints

- See Labrosse chapter 13 for the steps of OSCtxSw(). Note that on Cortex-M4 OSCtxSw() just triggers PendSV and the actual work of OSCtxSw() is done by your ContextSwitch.
- OSTCBCur->OSTCBStkPtr
    - OSTBCur is a global pointer to the task block of the current task.
    - OSTCBStkPtr is the first field of the task control block struct



- uCOS priorities are 1 byte so use appropriate assembly language data movement instructions, i.e. LDRB/STRB.
- In uCOS, priorities are unique. No two tasks can have the same priority therefore the priority of a task also doubles as its task ID.
- This uCOS port uses only the Main stack pointer (MSP) throughout. However the Process stack pointer (PSP) is used as a flag to indicate whether the initial context switch has occurred: 4 means no, 0 means yes.
- Your uDebugger diagnostic Hard Fault handler may be useful for debugging; just remove the portion that hacks into the stack and replace it by a spin loop instead of returning from the handler.