Lab RTOS

Write up:

1. MRS RO PSP PendSV\_Handler runs during a context switch meaning that all the registers are going to get stored on the stack. R0 holds current stack pointer. Now this code starts swapping out the current process. In line 63, we subtract 0x20 from the PSP so that we can store all the values from R4-R11 on the stack. The lower addresses advance to unused space on the stack. STM will take the register at R0, now 0x20 lower than the PSP and start storing registers R4-R11 using that address. Registers 4 to 11 encompass 8 registers. LP1768 are 32 bit processors so each register holds 4 bytes. These registers are stored starting at the top of the stack going down (4 to 11). STM with no options stores first then increments, going downward on the stack. This is 0x20 bytes of data. Those registers then get stored on the stack above the current top of the stack. R0 still has the location on top of the stack that is free or where the current stack pointer should be. The idea then is first to set the stack pointer to the top where R4 would be stored. After each register (4 bytes is stored). Because no ! is used in the STM instruction R0 does not change.
2. nrk\_high\_ready holds the address of the current Task Control block. That address nrk\_high\_ready is a register that holds the address of the current TCB. Line 65 loads the data in that nrk\_high\_ready register into R1. The TCB is a struct stored in nrk\_task.h and it starts beginning at the address stored in R1. Line 66, LDR R1 [R1], loads the data in the first register of that structure into R1. The first register of os\_tcb is the address of the Task stack pointer. Now R1 holds the address of the Stack pointer of the current Task Control block, that is where the stack pointer of the current task should be stored. The last line 67 STR R0 [R1] stores what is in R0 into the register whose address is R1. That register holds the address where the current TCB’s stack pointer is. This last line then stores R0, the new top of the stack into the address for the currest TCB. It stores the current stack pointer into the TCB’s stack Pointer field. This task will be swapped out so this completes the process by moving the stack pointer into the TCB’s stack pointer register. The current stack pointer now points to where R4 for that register was.
3. In line 80, LDR R1, =nrk\_high\_ready\_TCB, takes value in register nrk\_high\_ready\_TCB, and loads it into R1. R1 now holds the address of the current TCB, the TCB that will be switched in. Line 81, LDR R2, [R1], loads into register R2 the value in R1. R2 now has the address for the TCB where the stack pointer should be. The last line, LDR R0, [R2], loads into R0, the value in register R2. The value in that register is the TCB’s stack pointer.