Fully Preemptive nxtOSEK Kernel with Preemption Threshold Scheduling

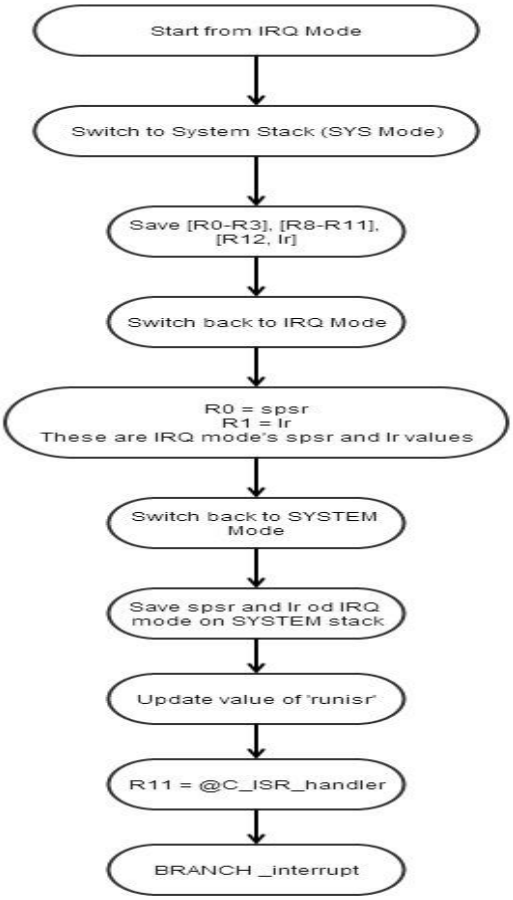
Jimit Doshi (jdoshi@ncsu.edu) and Saransh Gupta (sgupta20@ncsu.edu)

We present below the tasks done so far for this project. Essentially, we have mapped out the entire flow chart for normal scheduler operation as documented below:

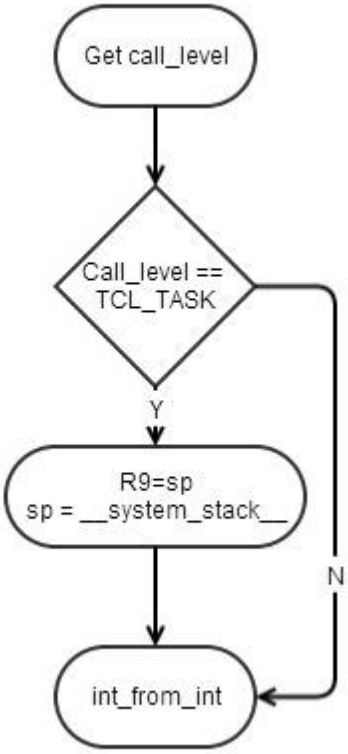
Normal Scheduler Operation

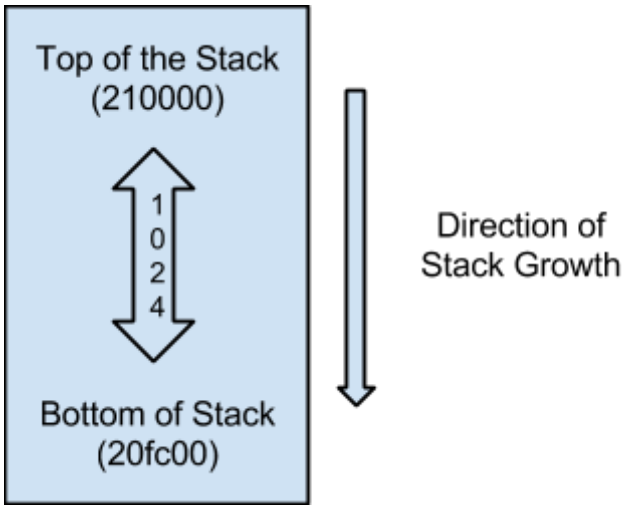
We begin with the timer ISR handler in the irq.s file from where the execution begins in case of a systick interrupt.

- File: irq.s
  Routine : (macro) irq_wrapper_type2, C_function, isr



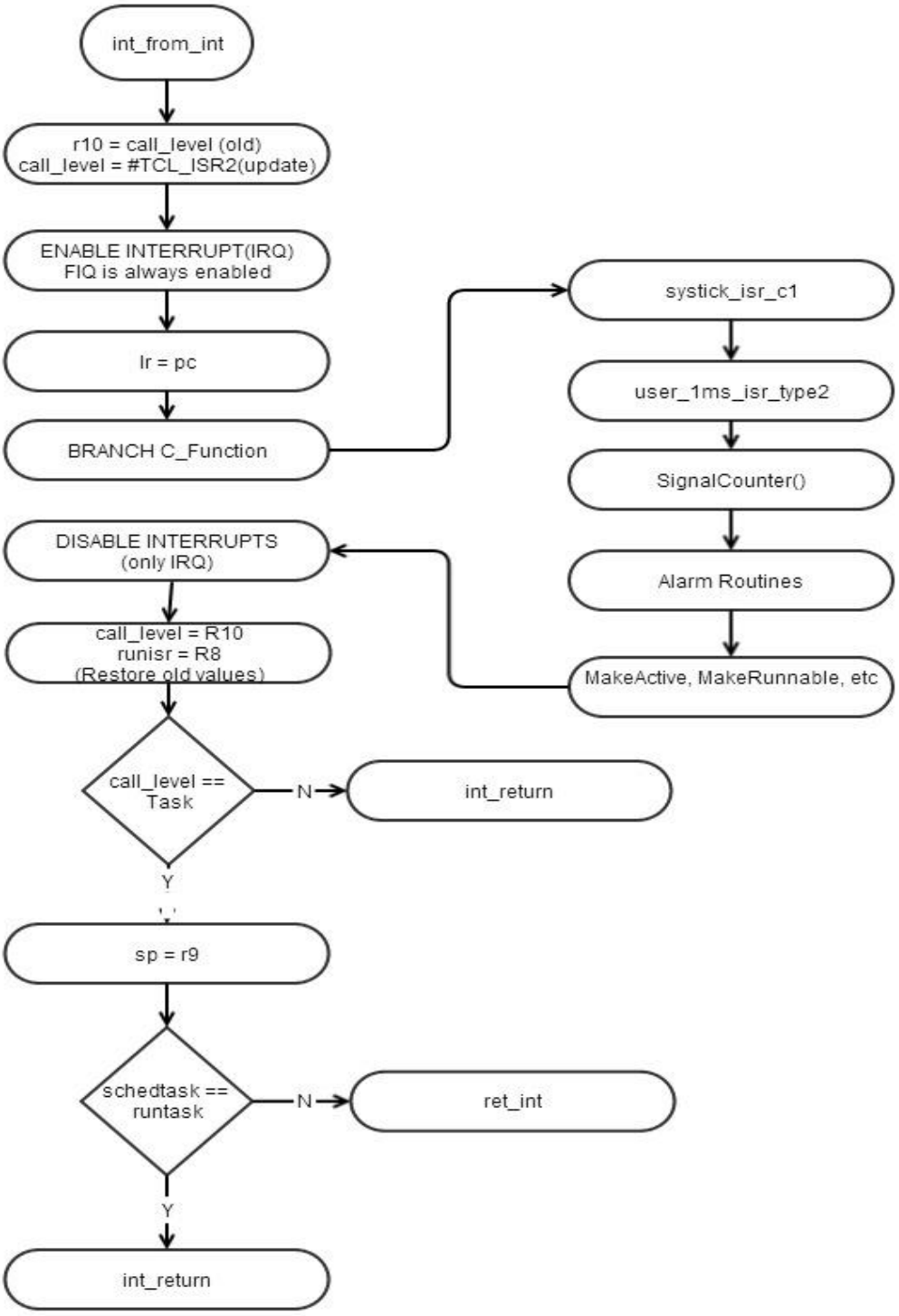
- File : cpu_support.s
  Routine : _interrupt



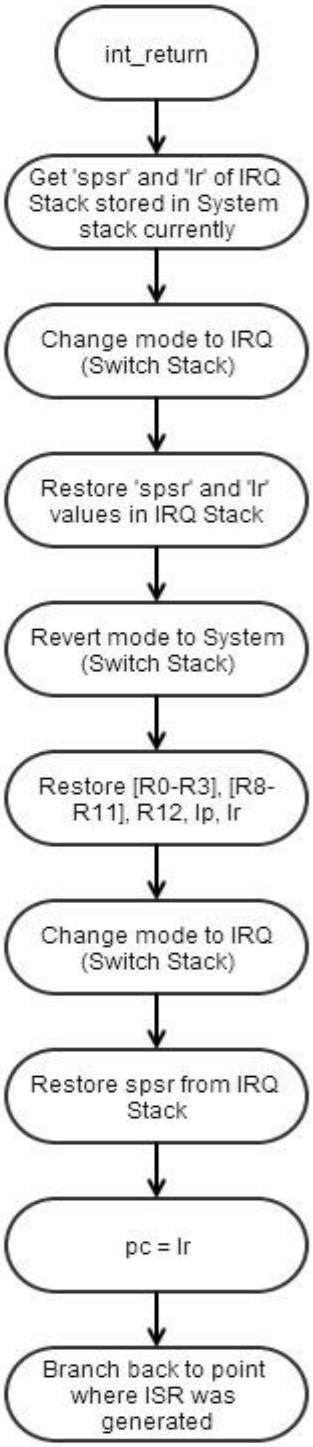


Only for call_level == TCL_TASK does the sp gets pointed to the top of the stack (_system_stack_). This is because otherwise it's an irs_to_isr call in which case the data from the previous isr is still saved on the system stack, and so sp should not be pointed to the top of the stack.

- File : cpu_support.s
  Routine : int_from_int



- File : cpu_support.s
  Routine : int_return



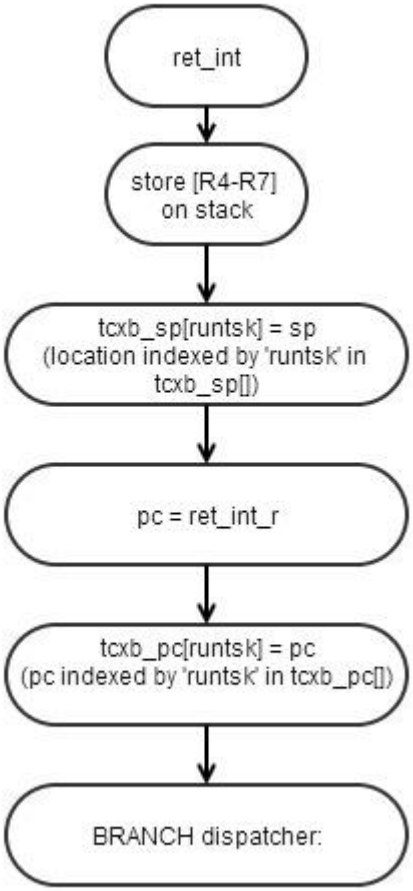
Whenever interrupt occurs the core does the below:

1. CSPR is stored in the SPSR of the new mode (IRQ Mode in our case).
2. lr_irq = pc

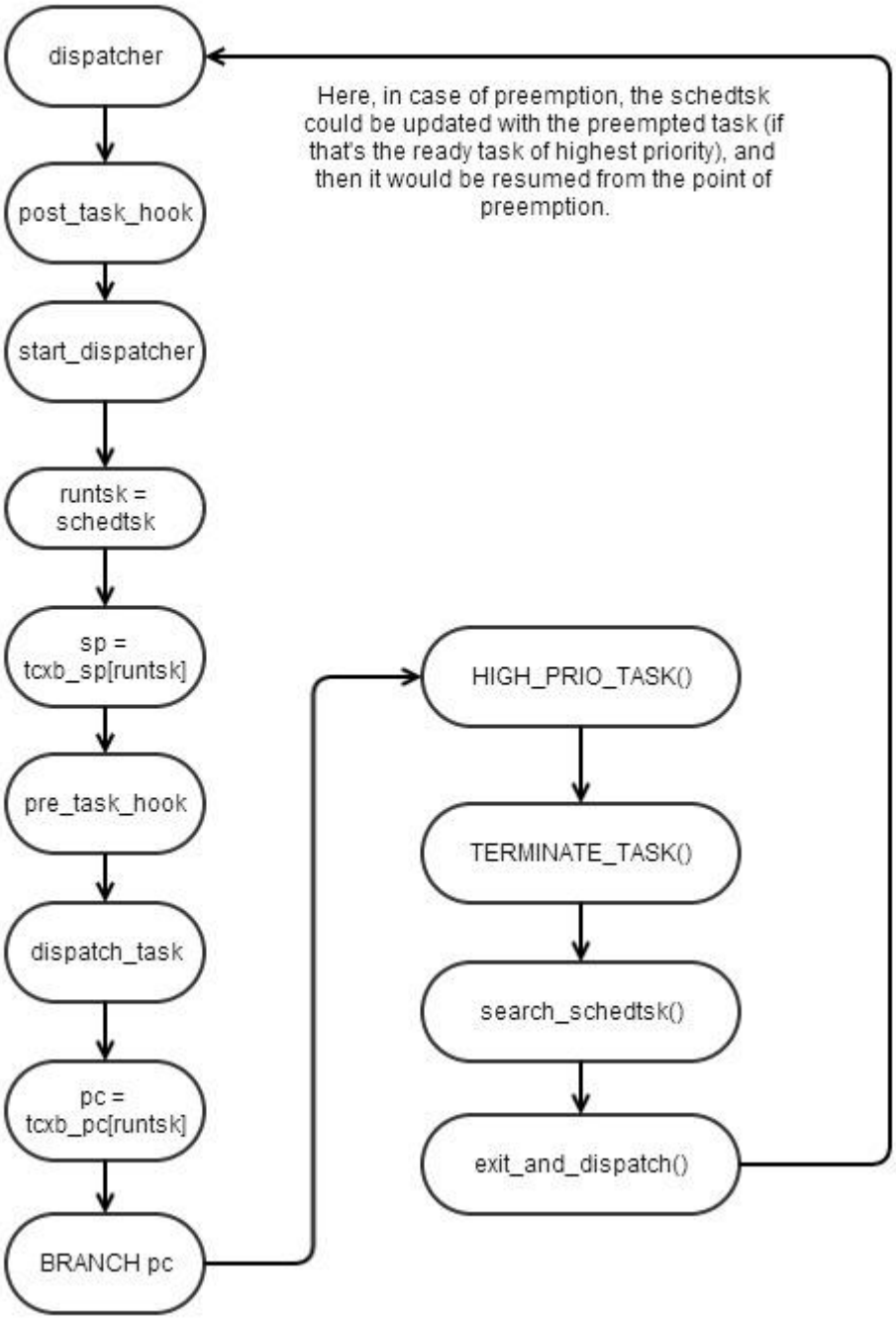
Hence if nested interrupts are allowed without any precautions, lr would get corrupted (previous isr's lr would be overwritten) . Therefore to support nested ISR's move processing to System stack after saving lr_irq in System stack (and this can be done recursively)

Therefore, towards the end of the interrupt, the lr_irq value needs to be popped from the System stack (as it was stored there at the beginning of the ISR).

- File : cpu_support.s
  Routine : ret_int



- File : cpu_support.s
  Routine : dispatcher



Open Points / Key Observations:

1. Entire ISR is executed from the System stack and this architecture is followed to support nested interrupts, so that a given isr's lr doesn't get corrupted when another isr occurs within it.
2. While the IRQ interrupts are disabled explicitly in int_from_int, we couldn't figure out yet where/when are they enabled again.
3. Analysis of the int_from_int code shows that in case of preemption, before the preempting task's execution is begun sp is pointed back to the preempted task's stack (and not the system stack). This implies execution of the preempting task occurs in the preempted task's stack. As such, this should support nested preemptions. We need to figure out what is preventing this (nested preemptions) from happening. Of course, the fact that interrupts are disabled (and hence the scheduler/kernel is suppressed) is the root cause, however, if we resolve that, then do we have any other blocking point to prevent nested preemptions? We don't think so, based on our current understanding of the code.
4. We have studied the theoretical aspects of PTS algorithm but haven't started with the implementation yet, as it would be premature to do so without having a kernel that supports nested preemption. We plan to take it up once we are done with the implementation and testing to support nested preemptions.

Tasks Accomplished (with reference to the Project proposal)**:**

Task	Team Member
Understand the complete assembly code used for 'dispatch' and ISR routines	Jimit
Analyze the impact of making changes in the kernel scheduler on any other part of nxtOSEK	Saransh
Documentation of understanding, preparation of interim report	Jimit & Saransh

Tasks in progress / planned:

Task	Team Member	Date
Coordinate with nxtOSEK kernel designers/contributors and also Dr.Mueller for their support in understanding the issue with preemption or the rationale behind present implementation	Saransh & Jimit	4/4/2014
Design algorithm and implementation to resolve the preemption issue	Jimit	4/15/2014

Design test cases and perform validation	Saransh	4/20/2014
PTS algorithm design and implementation	Saransh	4/22/2014
PTS test and validation	Jimit	4/24/2014
Final report preparation	Saransh & Jimit	4/25/2014
Presentation	Saransh & Jimit	4/27/2014

PTS implementation is subject to timely and successful completion of nested preemption implementation.