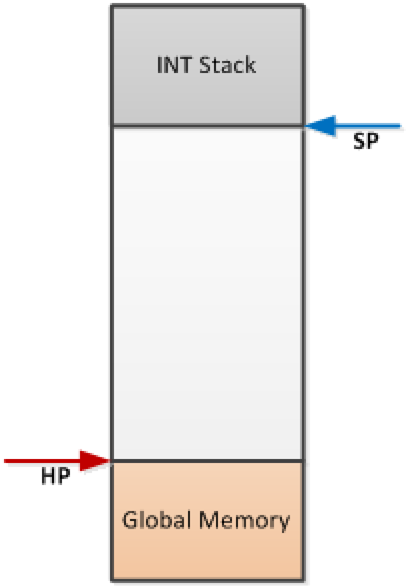
Stack & Heap Demonstration

Preet Kang

April 2012

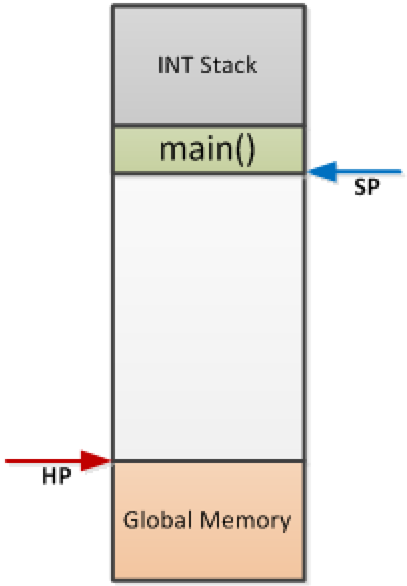
CPU Starts

- Startup File configures initial Heap & Stack Pointers



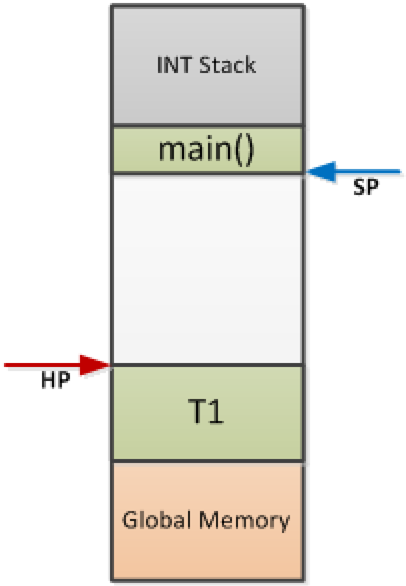
Main Allocates Memory

- Stack Pointer moves down based on variables declared in main()



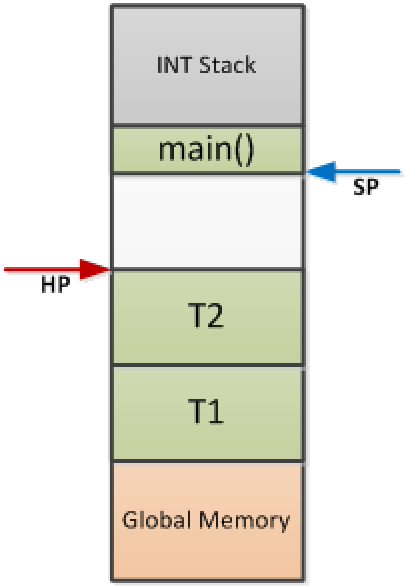
Task 1 Created

- Task 1 gets its memory from HEAP to be later used for its STACK



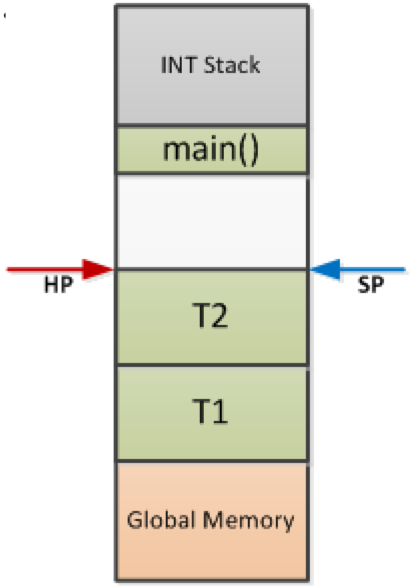
Task 2 Created

- Another Task gets memory from the HEAP for its STACK



FreeRTOS Starts & T1 Starts

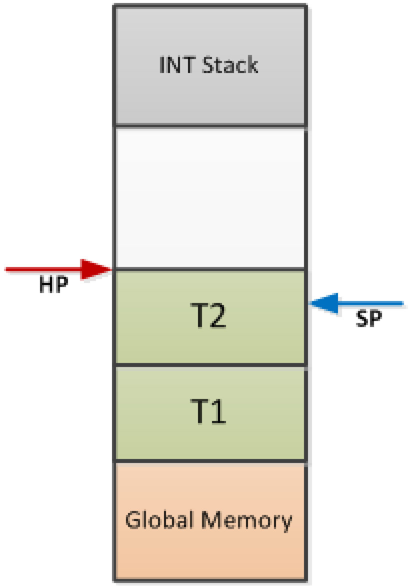
- main() now essentially gives up CPU and FreeRTOS will never enter it again.



- FreeRTOS now manipulates STACK pointer based on which task is currently in context.

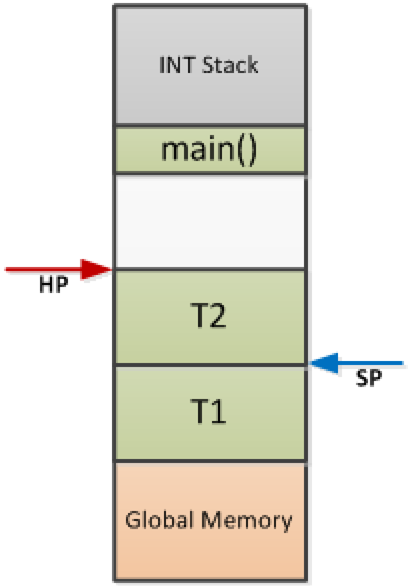
T1 Allocates Memory On Stack

- T1's STACK moves down to make room for local variables



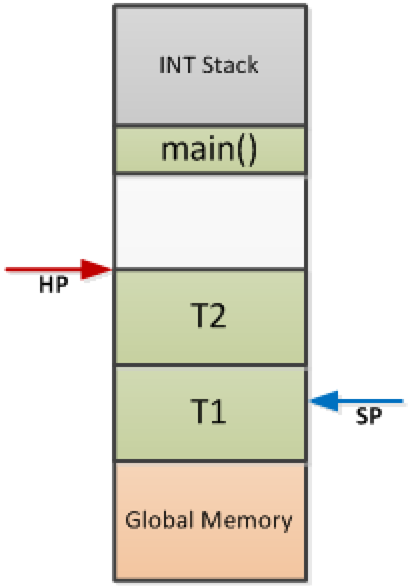
Context Switch to T2

- FreeRTOS manipulates STACK pointer to run T2 so it is looking at its vars.



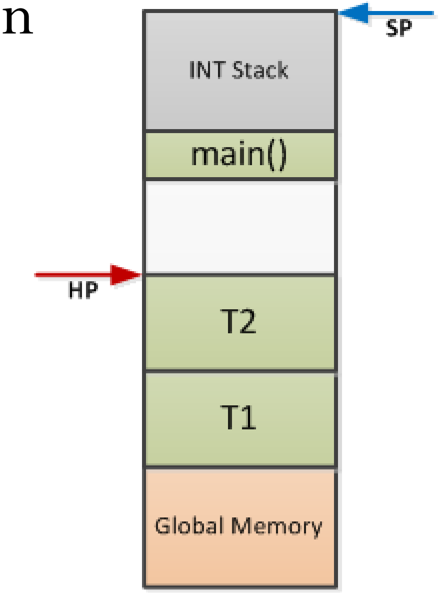
T2 Allocates Memory on Stack

- When T2 allocates memory, it comes from its STACK



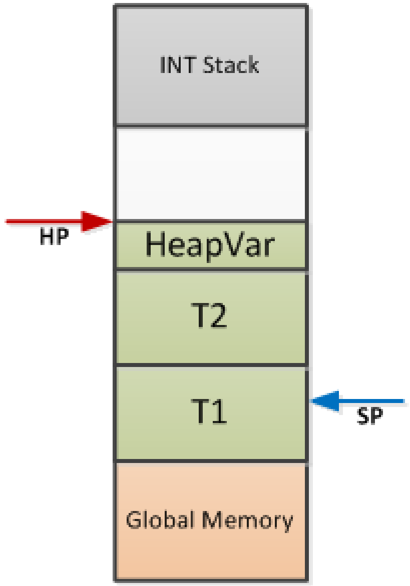
Interrupt

- When Interrupt occurs, the Hardware moves STACK Ptr to its dedicated region



Any Task Allocating Heap

- Any task allocating memory from HEAP comes from global Heap Memory


INT Stack
HP
HeapVar
T2
T1
SP
Global Memory