

# Lecture5

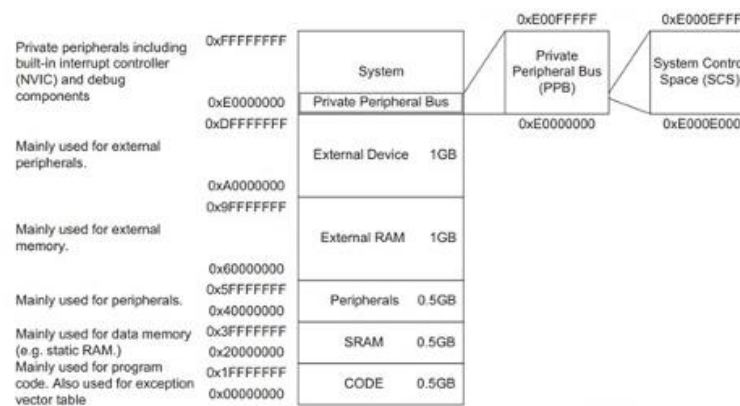
---

## STACK MEMORY

By: Abdelrahman matarawy

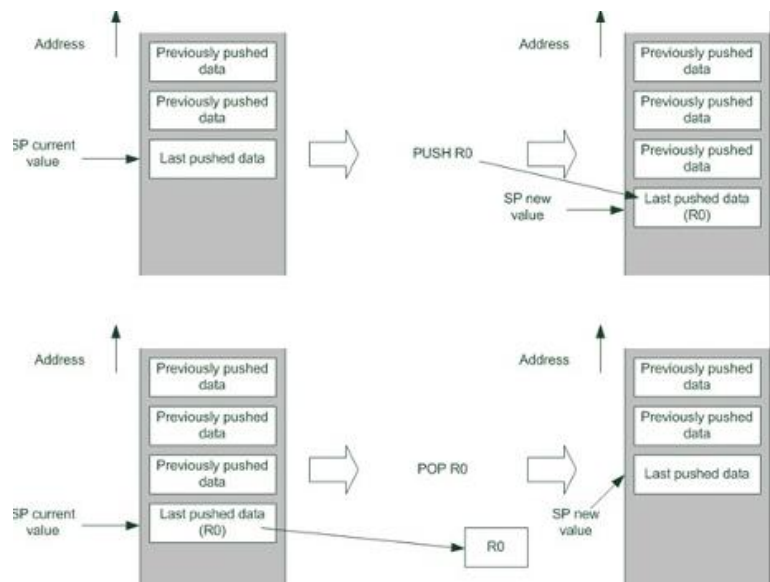
## Memory Map:

- Cortex M processor contain 4Gbytes of address space is portioned into 4 groups:
  - Program code (Code Region)
  - Data access (SRAM Region)
  - Peripherals (Peripheral region)
  - Processor internal control and debug components

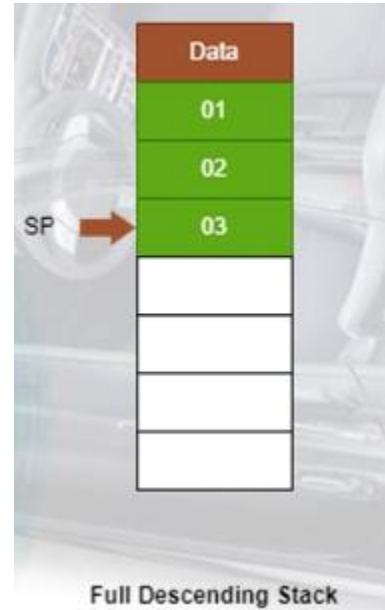


## Stack Memory:

- Stack is kind of Memory use mechanism LIFO data storage buffer.
- ARM processors use main system memory for stack memory operations and push instructions to store data in stack and pop instruction to retrieve data from stack.

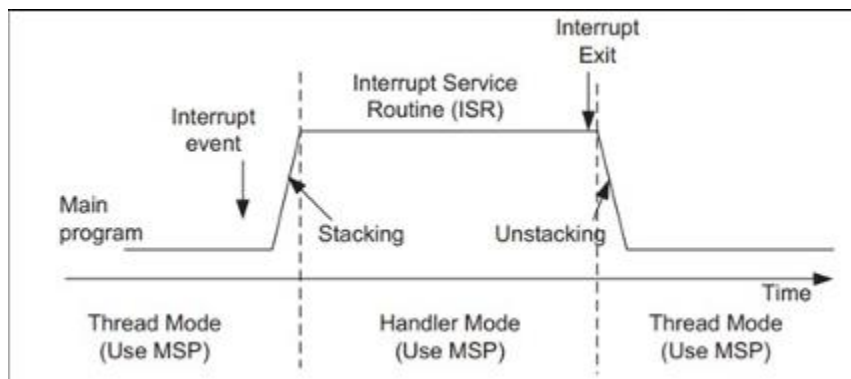


- Cortex M Processor use stack memory model called “Full Descending Stack”
- For each Push:
  1. Processor decrement SP.
  2. Store Value in Memory.
- For each POP:
  1. Reading Value from memory location as SP point.
  2. Then increment value of SP.



## ✓ Physically There are two stack pointers in Cortex M Processor

- Main Stack Pointer:
  - This is the default stack pointer used after reset/power ON and used in Exceptional Handlers.
- Process Stack Pointer:
  - This is alternative stack pointer used only on Thread mode.
  - It's usually used in Application Tasks in embedded system running on Embedded OS.



## Lab:

- We Will use estack as stack top and start of main stack pointer.

```

STM32F103C6TX_FLASH.ld
16 * <h2><center>&copy; Copyright (c) 2020 STMicroel
17 * All rights reserved.</center></h2>
18 *
19 * This software component is licensed by ST under
20 * the "License"; You may not use this file except
21 * License. You may obtain a copy of the License a
22 *      opensource.org/licenses/
23 *
24 *****
25 */
26
27 /* Entry Point */
28 ENTRY(Reset_Handler)
29
30 /* Highest address of the user mode stack */
31 estack = ORIGIN(RAM) + LENGTH(RAM); /* end of
32
33 _Min_Heap_Size = 0x200; /* required amount of heap
34 _Min_Stack_Size = 0x400; /* required amount of
35
36 /* Memories definition */
37 MEMORY
38 {
39   RAM   (xrw)  : ORIGIN = 0x20000000, LENGTH
40   ROM   (rx)   : ORIGIN = 0x80000000, LENGTH =

startup_stm32f103c6tx.s
126 .section .isr_vector,"a",%progbits
127 .type g_pfnVectors, %object
128 .size g_pfnVectors, .-g_pfnVectors
129
130 g_pfnVectors:
131 .word estack
132 .word Reset_Handler
133 .word NMI_Handler
134 .word HardFault_Handler
135 .word MemManage_Handler
136 .word BusFault_Handler
137 .word UsageFault_Handler
138 .word 0
139 .word 0
140 .word 0
141 .word 0
142 .word SVC_Handler
143 .word DebugMon_Handler
144 .word 0
145 .word PendSV_Handler
146 .word SysTick_Handler
147 .word WWDG_IRQHandler /* Win
148 .word PVD_IRQHandler /* PVD
149 .word TAMPER_IRQHandler /* Tam
150 .word RTC_IRQHandler /* RTC
151 .word FLASH_IRQHandler /* Fla
152 .word BRR_IRQHandler /* BRR

```

## ➤ Notices

- that when we start the Firmware SP stopped at address 0x20002800
- We work for STM32F103C6 SRAM length is 10K bytes (10 \* 1024 = 0x2800)
- Start of SRAM(0x20000000) the result is 0x20002800
- Start of MSP set on same address.

```

162 int main(void)
163 {
164   clk_init();
165
166   /* Set Configuration of EXTI */
167   EXTI_Config_t EXTI_Conf; // Make Variable of EXT
168   EXTI_Conf.EXTI_PIN = EXTI_LINE9_PB9; // Set line
169   EXTI_Conf.Trigger_Case = EXTI_Trigger_RASING; //
170   EXTI_Conf.IRQ_Case = EXTI_IRQ_Enable; // Enable In
171   EXTI_Conf.P_IRQ_Callback = EXTI_Callback; // Make
172

```

Registers:

R11	0x00000000
R12	0x00000000
R13 (SP)	0x20002800
R14 (LR)	0x08000863
R15 (PC)	0x08000ADC
xPSR	0x61000000

Watch 1:

Name	Value	Type
val3	<cannot evaluate>	uchar
_S_MSP	0x20002800	uint
<Enter expression>		

## ➤ After first trigger:

- PSP carry address of Start\_TaskA.
- SP switch from MSP to PSP.
- switch from privileged to unprivileged.

The screenshot shows the Keil uVision IDE with the following details:

- Registers:** R13 (SP) is 0x200025F8, R15 (PC) is 0x0800A6E. The MSP register is also shown at 0x200025F8.
- Disassembly:** The assembly code shows the execution of the instruction `TaskA_Flag = TASKA(1, 2, 3);` at address 0x0800A6E.
- Watch1:** The watch window shows the values of `_S_MSP` (0x20002800), `_S_PSP_Task_B` (0x2000258C), and `_S_PSP_Task_A` (0x200025F8).
- GPIOB Configuration:** The window shows the configuration for the GPIOB pin, with the mode set to 'Input' and the configuration register (GPIOB\_CRH) set to 0x44444444.

## ➤ After second trigger

- PSP carry address of Start\_TaskA
- SP switch from MSP to PSP
- Switch from privileged to unprivileged.

The screenshot shows the Keil uVision IDE with the following details:

- Registers:** R13 (SP) is 0x2000258C, R15 (PC) is 0x0800ABC. The MSP register is also shown at 0x2000258C.
- Disassembly:** The assembly code shows the execution of the instruction `TaskB_Flag = TASKB(1, 2, 3, 4);` at address 0x0800ABC.
- Watch1:** The watch window shows the values of `_S_MSP` (0x20002800), `_S_PSP_Task_B` (0x2000258C), and `_S_PSP_Task_A` (0x200025F8).
- GPIOB Configuration:** The window shows the configuration for the GPIOB pin, with the mode set to 'Input' and the configuration register (GPIOB\_CRH) set to 0x44444444.