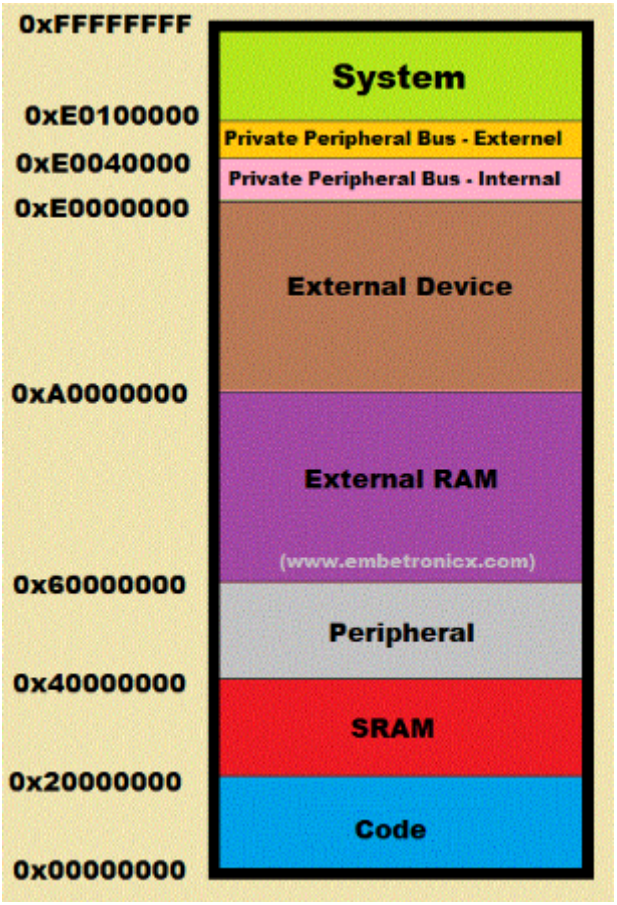
Booting Flow

INDEX

1.Microcontroller Memory Architecture

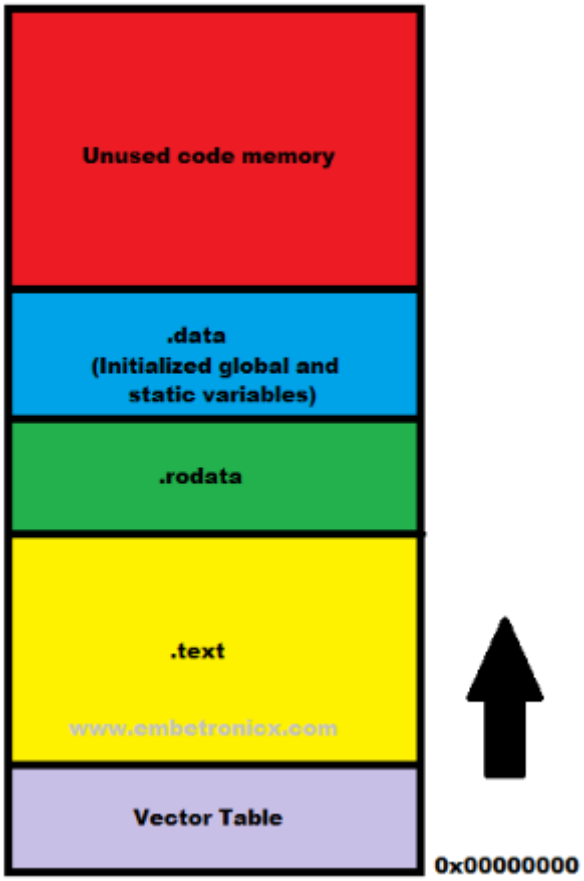
Before starting this we must know about the microcontroller memory architecture



- Here, we have a **code** region, where we are going to write our final binary. That memory is starting from **0x00000000** to **0xFFFFFFFF**. Now we will forget about other regions. We will take only the SRAM and Code region.
- We know that the code region has the final output of our program (.hex or .bin or etc). SRAM will be having stack, heap, global RW variables, and Static variables, etc.

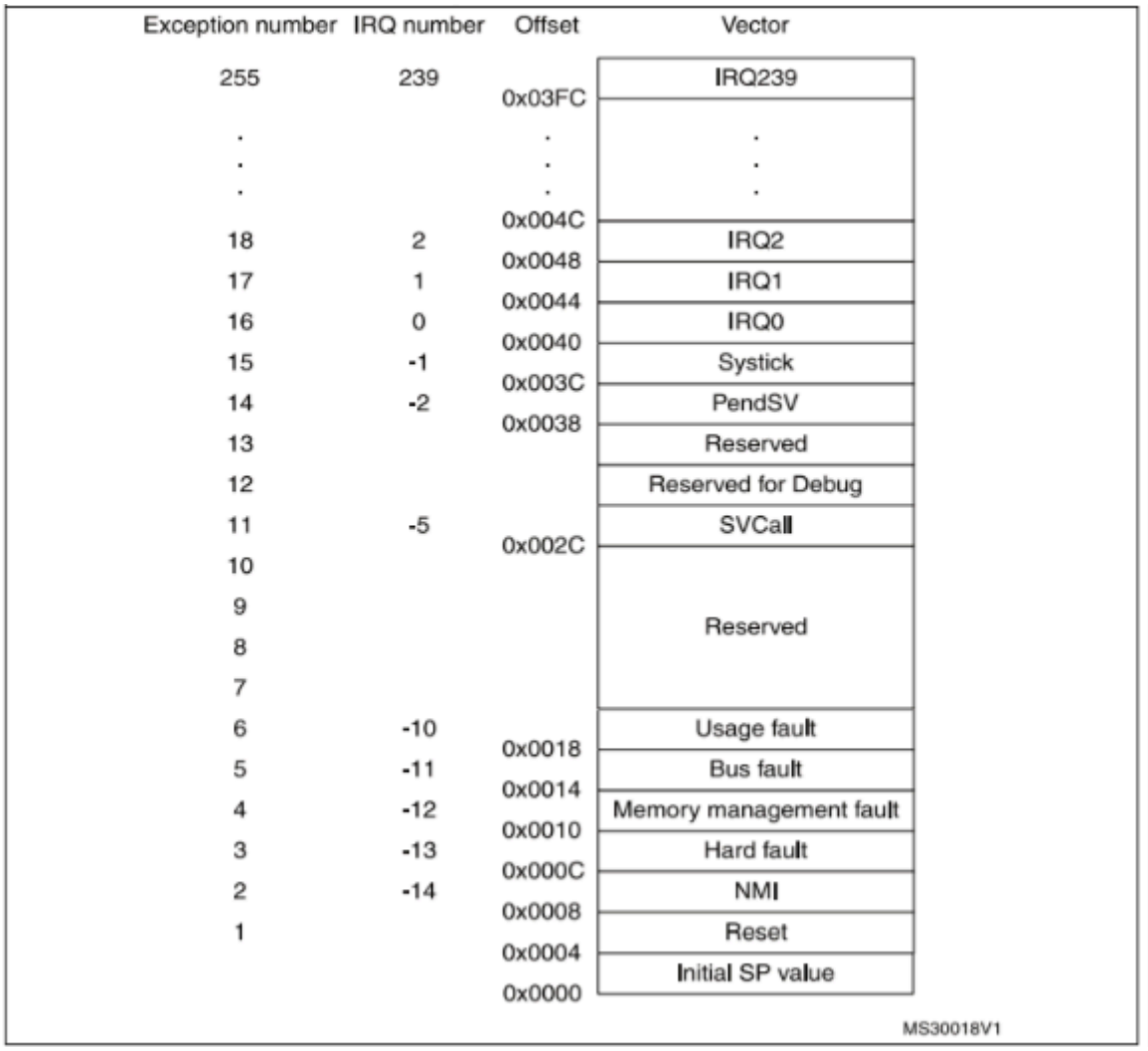
2. Memory layout of the program

- Now we have to know how the final output of our program is stored in the code region. The below image will explain how our program is stored in the code region.
- the final output of our program will be ordered in this way by using a linker. If you want to know the memory layout of the program, you can refer to this tutorial. So, the vector table is starting from 0x00000000.
- *Note: By default vector table will be starting from 0x00000000.*



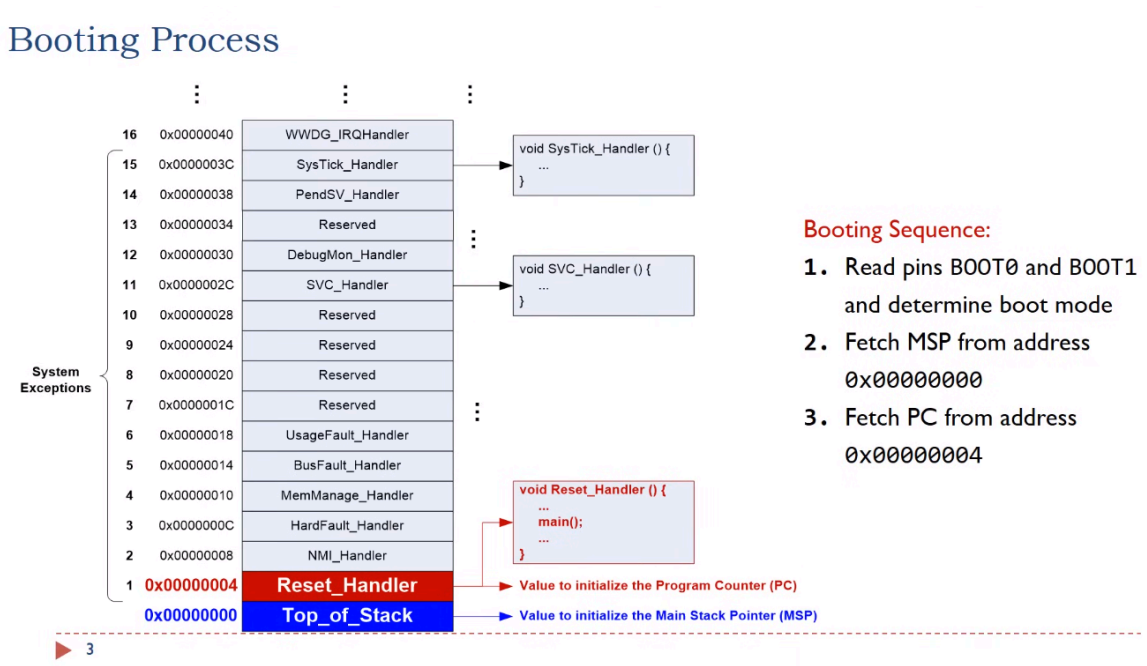
3.Vector table of ARM Cortex_M4

- That vector table will contain all the locations of the exception and interrupt ISR. As Cortex-M4 has below exceptions, interrupts and those things are ordered in the below image.
- So, the **0x00000000** address contains the initial **Stack Pointer** Value. Then 0x00000004 has the address of the **reset handler.**



4.What happens When you press the reset button in ARM Cortex-M4?

1. **PC** (Program Counter) will be loaded with 0x00000000. So will start from the address 0x00000000.
2. Since the address has an Initial Stack Pointer value, It will be fetched to the MSP (Main Stack Pointer). So, that value will be starting of the stack.
3. Then PC will be loaded with the Reset handler's address.



Note: These above 3 steps are done by the hardware (This is architecture-specific).

4. After that, the reset handler will perform the below operations.

1. Initialize the system.
2. Copy the Initialized global variable, static variable (.data) to SRAM.
3. Copy the Un-initialized data (.bss) to SRAM and initialize it to 0.
4. It Calls main().

This is how your **main()** is getting called while power-up or press the reset button. You may understand clearly if you see the execution below.

Summary of Booting

- When you press the reset button, it will copy the Stack pointer to **MSP** (Main Stack Pointer) from the location of 0x00000000.
- Then it moves to Reset_Handler.
- In the Reset_handler, it will initialize the hardware (system), then copy the initialized data (Initialized global variable and static variable) to SRAM.
- Then copy the uninitialized data to SRAM and initialize with 0.
- Finally, it calls our main function.

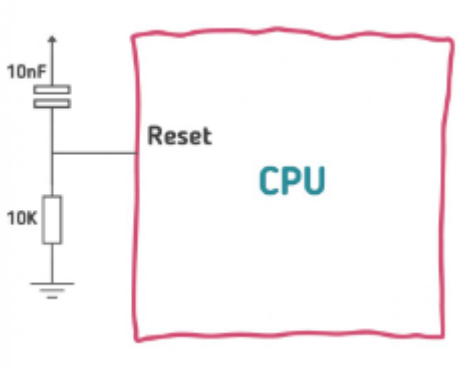
Note: This process is applicable when you don't have a bootloader. If you have a bootloader, then it will execute in a different way.

5. Main steps in booting process

The booting process in ARM-based systems is crucial for initializing the hardware and preparing the system to run an operating system or application software. Here is a detailed explanation of the booting process in ARM-based systems:

5.1 Power-On Reset (POR)

- The Boot Process begins with the Power On Reset (POR). When an MCU is first powered up, it has to run predefined checks in the hardware to bring the MCU up, only when all the voltage and clock conditions are proper for operation. Until then this hardware circuitry holds the processor in the reset. This process is called the Power On Reset (POR).



- POR typically is the first step in this initialization process. It ensures that stable and adequate supply voltages are available to the circuit.
- It is a circuit that watches voltage levels and holds circuit operation until voltage levels reach suitable levels. The PoR circuit can reset the system if voltage levels fall outside required levels or if other signals, such as a manual reset button or watchdog timer inputs in some PoR chips, trigger a reset state. PoR is generally called a *hardware reset*
- The most basic POR system includes a resistor and capacitor connected together with values set so that the capacitor takes a predictable and constant time to charge up when power is first applied.

- Once the power-on reset circuit recognizes suitable and stable voltage levels to the system, it delivers a digital signal that enables the rest of the system initialization process to continue

5.2 Memory Aliasing (Remapping) and Architecture

- After the POR Stage MCU comes out of the reset state. The processor starts pointing from the **0x0000_0000** address. This zero address can also be remapped to any other address in the address space with the help of **Remapper hardware**
- **Remapper Hardware** : It is also known as a memory remapping unit, is a specialized hardware component that can change the address mapping of memory regions. This allows the system to present a different memory layout to the CPU and other peripherals without physically moving the data in memory.
- With the remapping hardware, The processor still looks for **0x0000_0000** but now the Remapper hardware window changes the address to a new remap address. This results in booting/execution from different address spaces or different applications. This is also called **Memory Aliasing**.
- This boot setting for **Remapper hardware** is typically configured from the **Boot Mode** selection pin of the MCU
- Processor internal circuitry reads this boot mode pin configuration to set the Remapper address as per the selected configuration. Only after this stage processor starts fetching instructions and data from the correct targeted memory address.

5.3 Firmware Booting

- After the **Memory Alias** stage, where the processor takes the boot mode selection pin configuration and is ready to fetch the instruction from the target address, firmware booting starts.
- Firmware boot starts with the fetch of a WORD (32 Bit Memory Address) to the **Program counter (PC)**. The program counter (PC) is loaded with the address 0x0000_0000 value. As PC points to the next address for the execution, the **0x0000_0000** address is loaded to the **Main Stack Pointer (MSP)**. This is done in the hardware circuitry. PC after **0x0000_0000** points to the next instruction which is **0x0000_0004**. The memory map of the ARM based MCU has **0x0000_0000** as the address to the top of the stack and **0x0000_0004** as the Reset Handler. This initial memory map, is part of the Memory segment that is called **Vector Table**.
  Also in the ARM based MCU, the Initial top of the stack is loaded before the first function

(Reset Handler) of the application, allowing the **Reset Handler** to be written completely in C. As C code needs stack to run.

5.4 Reset Handler

Reset Handler in ARM based MCU is considered as an entry point of the firmware. This value of the reset handler in the vector table is basically a pointer to the actual function.

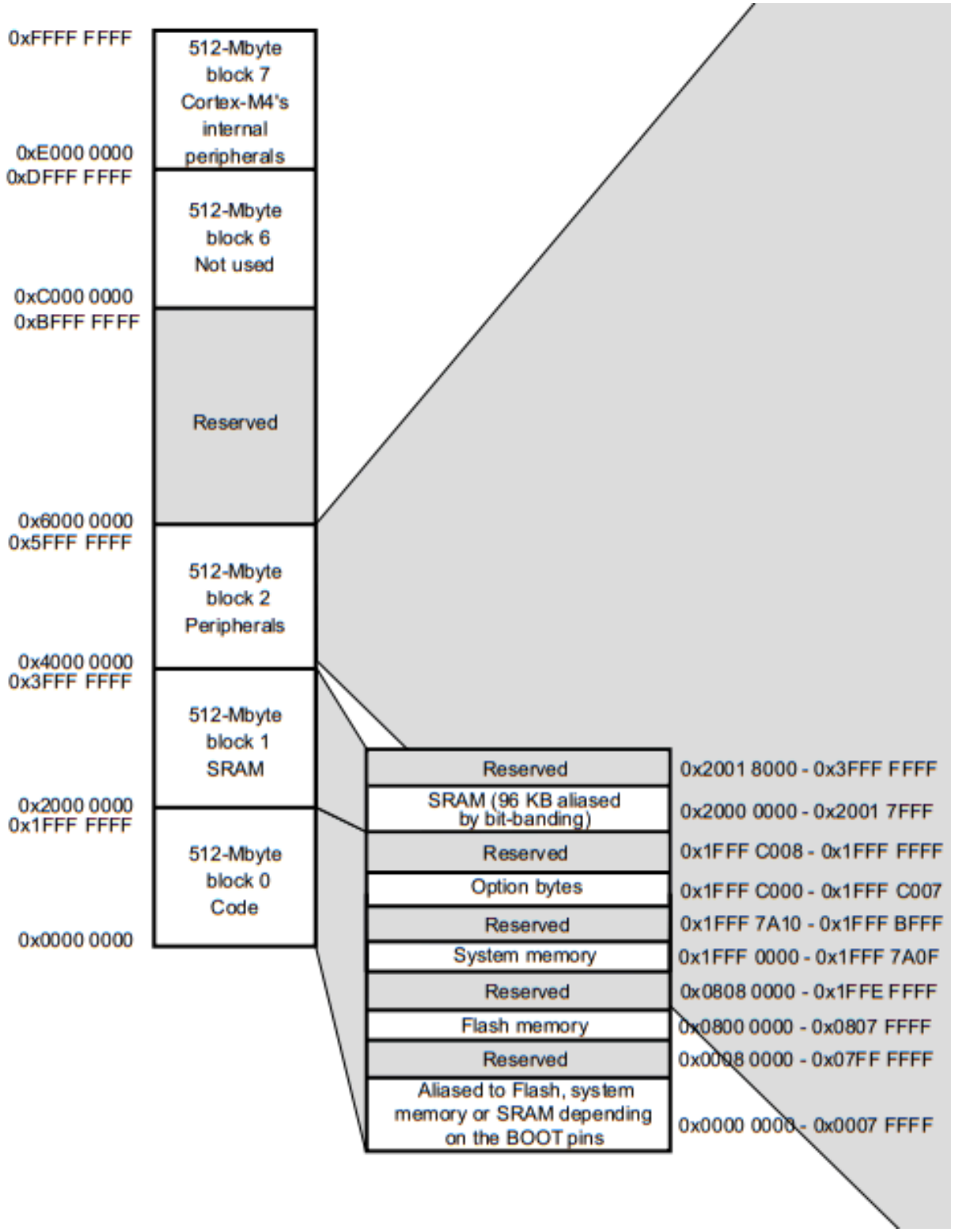
- **Reset Handler Does Below Things :**
    - Initializes the stacks and **CPU** Registers
    - Copies the Memory Data segments from the FLASH to the RAM and fills the **BSS** Segment variables with 0's
    - Initializes the Peripherals with the default states
    - Initializes the **MMU** (If Available)
    - Jump to the main function

6. Boot configuration

- Due to its fixed memory map, the code area starts from address 0x0000 0000 while the data area (SRAM) starts from address 0x2000 0000
- The Cortex®-M4 with FPU CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). STM32F4xx microcontrollers implement a special mechanism to be able to boot from other memories (like the internal SRAM).
- **Icode Bus :** The ICode interface is a 32-bit AHB-Lite bus interface. Instruction fetches and vector fetches from Code memory space (0x00000000 - 0x1FFFFFFF) are performed over this bus
- **Reser Vector :** The *reset vector* of a processor is the default location where, upon a reset, the processor will go to find the first instruction to execute. In other words, the reset vector is a

pointer or address where the processor should always begin its execution. This first instruction typically branches to the system initialization code.

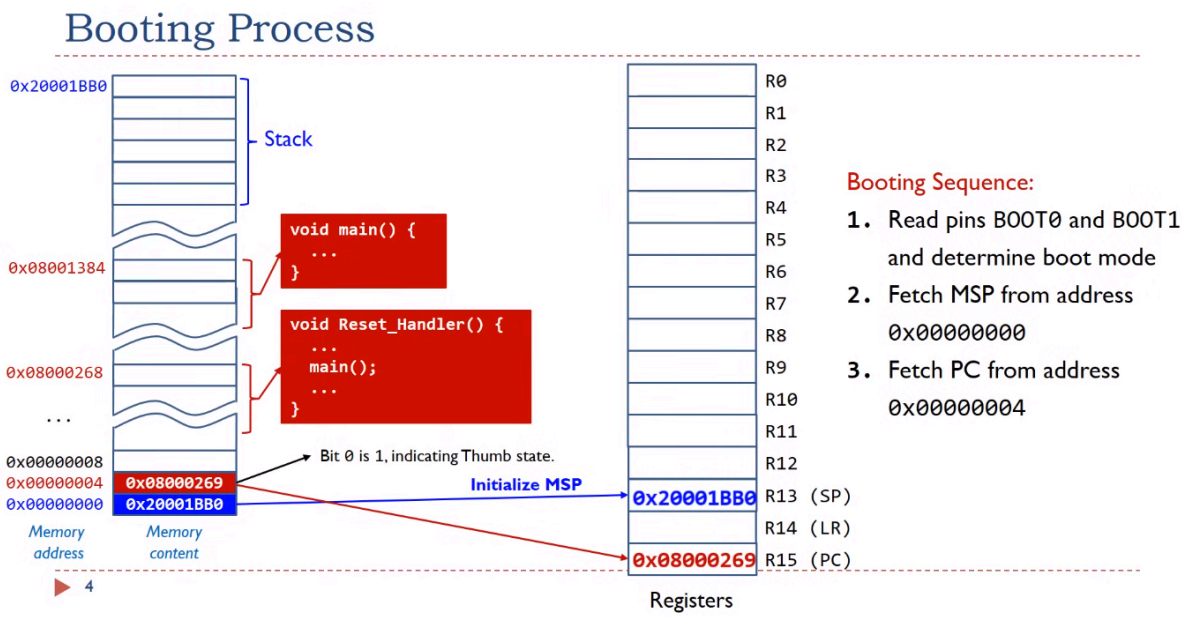
- ARM family processors have the address 0x00000000 reserved for the vector table (including reset vector, undefined instruction vector, software interrupt vector, interrupt request vector, fast interrupt vector), and the reset vector is at 0x00000000.



7.Booting sequence

Now we understand the basic concepts, lets see how they work in booting time.

1. Power On Reset
2. Execute boot loader in boot ROM
3. Processor read BOOT pins from SYSCFG memory remap register (SYSCFG_MEMRMP) to determine the booting mode. For example if BOOT0 is 0, Main Flash memory is mapped into Alias space from address 0x00000000
4. Next processor fetch the first word stored at memory address 0x00000000 from memory space. As the beginning of the memory space contains the vector table, and the first word in the vector table is the initial value for the Main Stack Pointer (MSP). Processor set up the MSP after that.
5. Processor continues read the next word stored at memory address 0x00000004 in vector table, now is reset vector, which contains the Reset_Handler() function address.
6. Reset_Handler() function address is moved into Program Counter (PC) and processor starts running Reset_Handler().
   Program counter : it always holds the memory address of next instruction to be executed by processor
7. Inside Reset_Handler(), System_Init( ) function will be run first, follow is our application (main())



- **Reset_handler()** : it performs some hardware initialization first like initializing of data segment and BSS (block start by symbol : the memory space for uninitialized variables of your code.)
- In the above picture we see that the reset handler address is different where LSB bit of the address indicate whether the processor rum in thumb mode / ARM mode

Differences between ARM and THUMB

THUMB	ARM
• Thumb mode is status that the T bit of CPSR(Current Program Status Register) is 1	• ARM mode is status that the T bit of CPSR(Current Program Status Register) is 0
• Thumb mode provides 16 bit instruction set	• ARM mode provides 32 bit instruction set
• Thumb mode fetch's instructions by 2 bytes	• ARM mode fetch's instructions by 4 bytes
• Thumb encodes a subset of the 32-bit ARM instructions into a 16-bit instruction set space	• More regular binary encoding
• Since Thumb has higher performance than ARM on a processor with a 16-bit data bus, but lower performance than ARM on a 32-bit data bus, use Thumb for memory-constrained systems	• ARM provides good performance with low power requirements, at the cost of code size
• Only the branch relative instruction can be conditionally executed	• All instructions are conditional
• The limited space available in 16 bits causes the barrel shift operations ASR, LSL, LSR, and ROR to be separate instructions in the Thumb ISA	• Pre-processing or shift occurs within the cycle time of the instruction. (MUL, CLZ and QADD are not having preprocessing)
• r0-r7 fully accessible, r8-r12 only accessible by MOV, ADD, and CMP, r13(sp), r14(lr), r15(pc) are limited access, cpsr only direct access, spsr no access.	• All registers are fully accessible

- ARM Cortex-M processors only support thumb mode, LSB of Program counter should always 1

- After the processor determine boot mode by reading BOOT0,BOOT1 processor copies the first word to stack pointer register (SP) and second word to program counter (PC)
- Here comes the boot mode that from where we need to boot the processor

8.Boot modes

- In the STM32F4xx, three different boot modes can be selected through the BOOT[1:0] pins

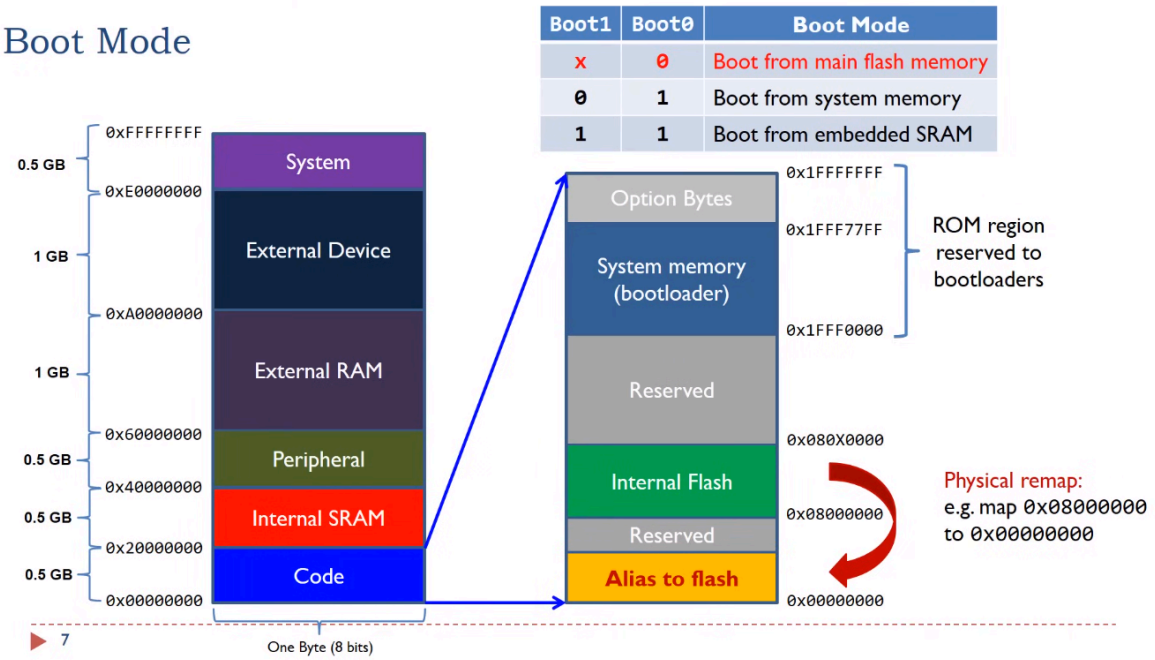
Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
X	0	User Flash memory	User Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

- The values on the BOOT pins are latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set the BOOT1 and BOOT0 pins after reset to select the required boot mode.
- BOOT0 is a dedicated pin while BOOT1 is shared with a GPIO pin. Once BOOT1 has been sampled, the corresponding GPIO pin is free and can be used for other purposes.
- The BOOT pins are also resampled when the device exits the Standby mode. Consequently, they must be kept in the required Boot mode configuration when the device is in the Standby mode
- After this startup delay is over, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory starting from 0x0000 0004
- Even when aliased in the boot memory space, the related memory is still accessible at its original memory space

Addresses	Boot/Remap in main Flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory
0x2000 0000 - 0x2001 7FFF	SRAM1 (96 KB)	SRAM1 (96KB)	SRAM1 (96KB)
0x1FFF 0000 - 0x1FFF 77FF	System memory	System memory	System memory
0x0804 0000 - 0x1FFE FFFF	Reserved	Reserved	Reserved
0x0800 0000 - 0x0807 FFFF	Flash memory	Flash memory	Flash memory
0x0400 000 - 0x07FF FFFF	Reserved	Reserved	Reserved
0x0000 0000 - 0x0007 FFFF(1)	Flash (512 KB) Aliased	SRAM1 (96 KB) Aliased	System memory (30 KB) Aliased

8.1 Boot from Main Flash Memory

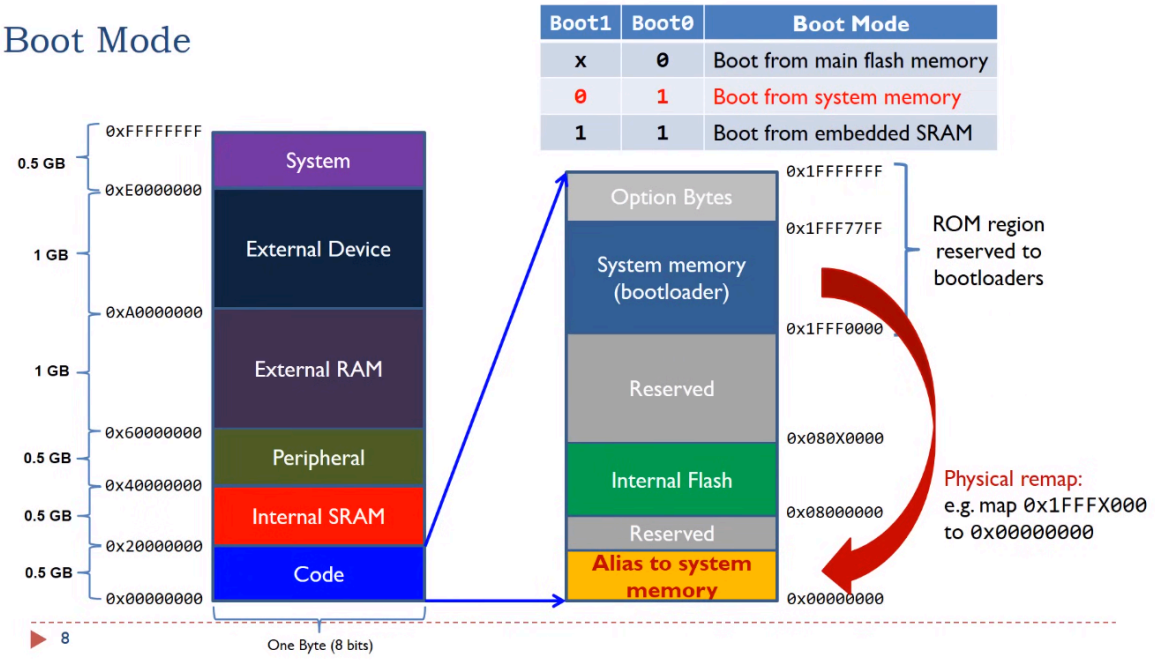
- **Description:** The system boots from the internal Flash memory, starting execution from the reset vector located in the Flash.
- If pin BOOT0 is grounded , the processor will physically map the internal flash memory to bottom area (alias to flash)
- Memory address 0x80000000 mapped to 0x00000000
- So flash memory contents can be accessed starting from address 0x00000000 to 0x80000000
- When the processor boots it always fetches the value of stack pointer and program counter from address 0x00000000 and 0x00000004 , because internal flash memory is remapped with address 0x0000000,flash memory is infact the boot memory
- **Typical Usage:** This is the standard boot mode for normal operation where the application code resides in Flash memory.
- **BOOT Pin Configuration:** BOOT0 = 0 (BOOT1 can be either 0 or 1).



8.2 Boot from System Memory (ROM)

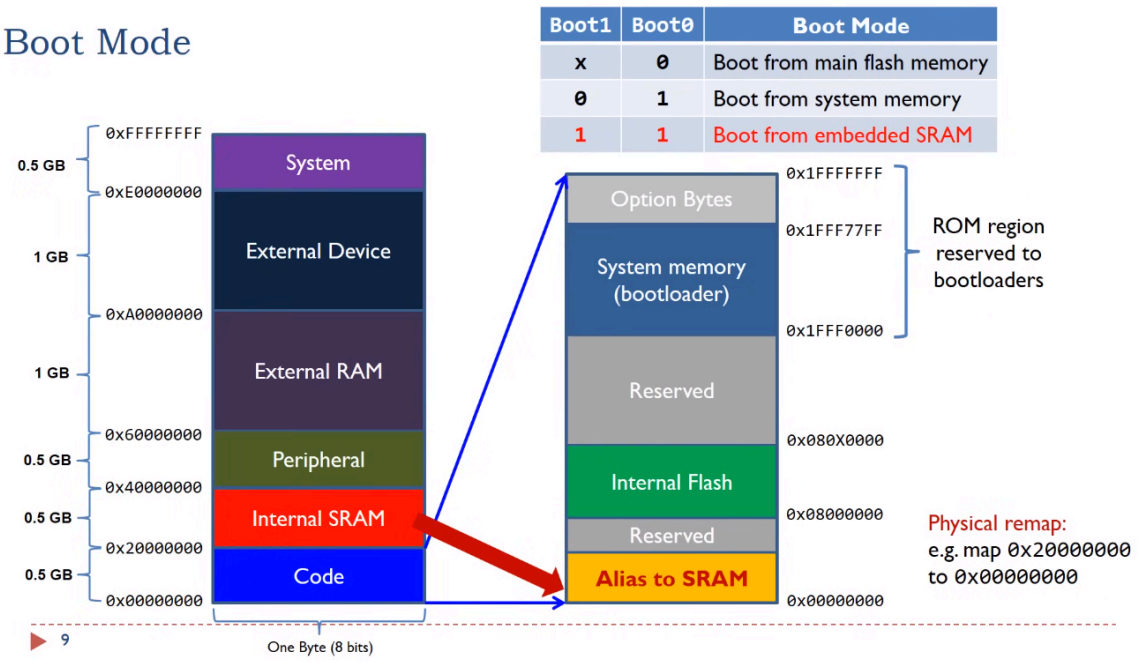
- **Description:** The system boots from a ROM area that contains the built-in bootloader. This bootloader can be used to update the firmware inside the internal flash memory through various communication interfaces (e.g., UART, USB, CAN, I2C, SPI).

- When BOOT1 is low system memory is mapped to bottom area and becomes its alias
- When processor fetch the value of program counter from memory address, processor infact fetch the value from system memory (it is chosen as boot memory)
- **Typical Usage:** This mode is used for reprogramming or updating the firmware inside the internal flash memory of the device, often via a serial interface.
- **BOOT Pin Configuration:** BOOT0 = 1, BOOT1 = 0.



8.3 Boot from Embedded SRAM

- **Description:** The system boots from the internal SRAM. This mode is primarily used for debugging or executing code directly from SRAM.
- When both pins are high the internal SRAM is mapped to bottom area 0x20000000 is physically mapped to 0x00000000
- Processor boots from SRAM
- **Typical Usage:** Useful for development and debugging purposes where the code is loaded into SRAM and executed directly from there.
- **BOOT Pin Configuration:** BOOT0 = 1, BOOT1 = 1.



9.SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap:

• Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT pins.

• After reset these bits take the value selected by the BOOT pins. When booting from main Flash memory with BOOT0 pin set to 0 this register takes the value 0x00.

In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.

Address offset: 0x00 Reset value: 0x0000 000X (X is the memory mode selected by the BOOT pins)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														MEM_MODE	
														rw	rw

- Bits 31:2 Reserved, must be kept at reset value.
- Bits 1:0 MEM_MODE: Memory mapping selection

- Set and cleared by software. This bit controls the memory internal mapping at address 0x0000 0000. After reset these bits take the value selected by the Boot pins .
    - **00:** Main Flash memory mapped at 0x0000 0000
    - **01:** System Flash memory mapped at 0x0000 0000
    - **11:** Embedded SRAM mapped at 0x0000 0000