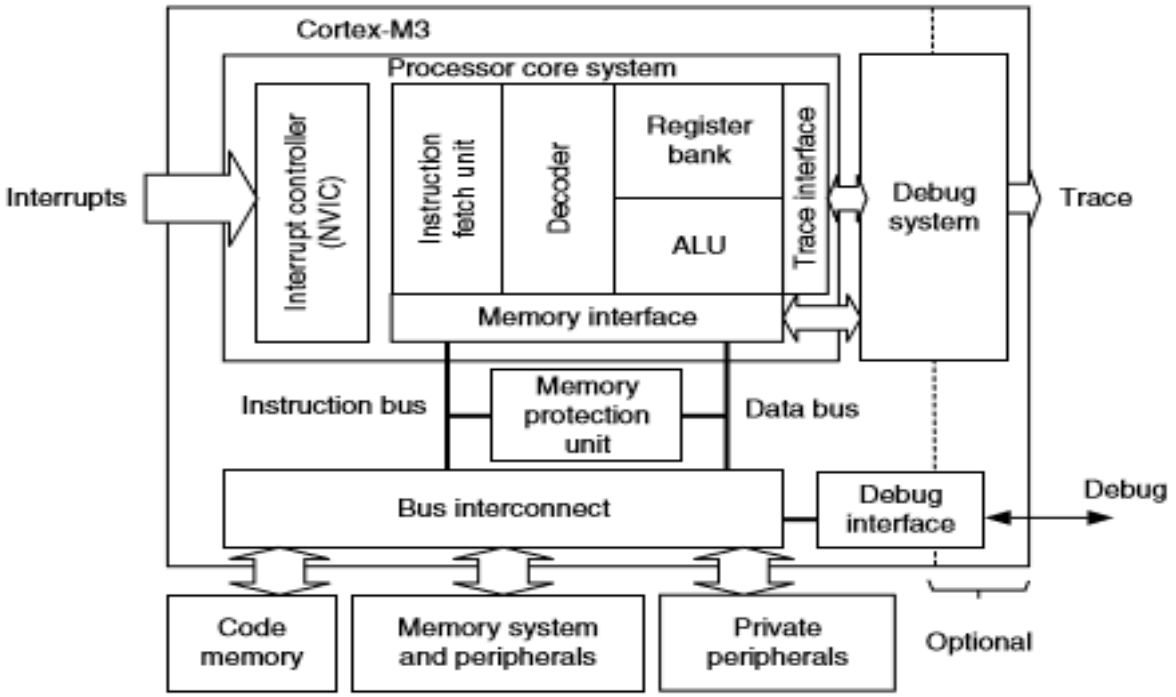
1. **Describe with neat diagram, the architecture of Cortex M3 Processor.**



The Cortex-M3 is a 32-bit microprocessor. It has a 32-bit data path, a 32-bit register bank, and 32-bit memory interfaces.The processor has a Harvard architecture.For complex applications that require more memory system features, the Cortex-M3 processor has an optional Memory Protection Unit (MPU). The Cortex-M3 processor includes a number of fixed internal debugging components. These components provide debugging operation supports and features, such as breakpoints and watchpoints. The Cortex-M3 processor has registers R0 through R15. R0–R12 are 32-bit general-purpose registers for data operations. The Cortex-M3 contains two stack pointers (R13).

- Main Stack Pointer (MSP): The default stack pointer, used by the operating system (OS) kernel and exception handlers.
- Process Stack Pointer (PSP): Used by user application code.

R14: The Link Register, R15: The Program Counter

The Cortex-M3 processor also has a number of special registers.

- Program Status registers (PSRs)
- Interrupt Mask registers (PRIMASK, FAULTMASK, and BASEPRI)
- Control register (CONTROL)

The Cortex-M3 processor has two modes and two privilege levels. The operation modes (thread mode and handler mode) determine whether the processor is running a normal program or running an exception handler like an interrupt handler or system exception handler. The privilege levels (privileged level and user level) provide a

mechanism for safeguarding memory accesses to critical regions as well as providing a basic security model.

The Cortex-M3 processor includes an interrupt controller called the Nested Vectored Interrupt Controller (NVIC). It is closely coupled to the processor core and provides a number of features such as:

- Nested interrupt support
- Vectored interrupt support
- Dynamic priority changes support
- Reduction of interrupt latency
- Interrupt masking

The Cortex-M3 has a predefined memory map. This allows the built-in peripherals, such as the interrupt controller and the debug components, to be accessed by simple memory access instructions. Thus, most system features are accessible in C program code. The predefined memory map also allows the Cortex-M3 processor to be highly optimized for speed and ease of integration in system-on-a-chip (SoC) designs.

There are several bus interfaces on the Cortex-M3 processor. They allow the Cortex-M3 to carry instruction fetches and data accesses at the same time. The main bus interfaces are as follows:

- Code memory buses
- System bus
- Private peripheral bus

The Cortex-M3 supports the Thumb-2 instruction set. This is one of the most important features of the Cortex-M3 processor because it allows 32-bit instructions and 16-bit instructions to be used together for high code density and high efficiency. It is flexible and powerful yet easy to use.

2. Write short notes on Exceptions, Interrupts and Vector Table.

The Cortex-M3 supports a number of exceptions, including a fixed number of system exceptions and a number of interrupts, commonly called IRQ. The number of interrupt inputs on a Cortex-M3 microcontroller depends on the individual design. Interrupts generated by peripherals, except System Tick Timer, are also connected to the interrupt input signals. The typical number of interrupt inputs is 16 or 32. Besides the interrupt inputs, there is also a nonmaskable interrupt (NMI) input signal. The actual use of NMI depends on the design of the microcontroller or system-on-chip (SoC) product you use. In most cases, the NMI could be connected to a watchdog timer or a voltage-monitoring block that warns the processor when the voltage drops below a certain level. The NMI exception can be activated any time, even right after the core exits reset.

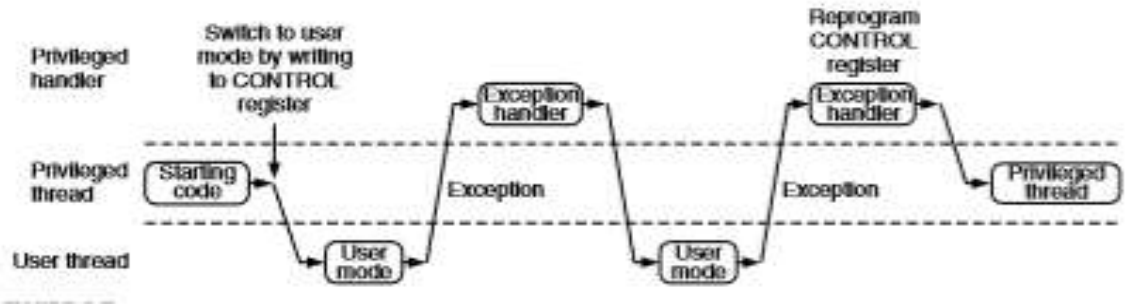
Vector Table: When an exception event takes place on the Cortex-M3 and is accepted by the processor core, the corresponding exception handler is executed. To determine the starting address of the exception handler, a vector table mechanism is used. The vector table is an array of word data inside the system memory, each representing the starting address of one exception type. The vector table is relocatable, and the relocation is controlled by a relocation register in the NVIC.

After reset, this relocation control register is reset to 0; therefore, the vector table is located in address 0x0 after reset. The LSB of each exception vector indicates whether the exception is to be executed in the Thumbstate. Because the Cortex-M3 can support only Thumb instructions, the LSB of all the exception vectors should be set to 1.

Table 3.4 Exception Types in Cortex-M3

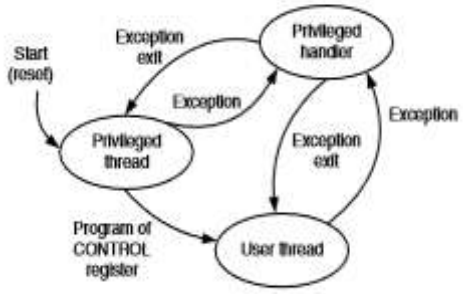
Exception Number	Exception Type	Priority	Function
1	Reset	–3 (Highest)	Reset
2	NMI	–2	Nonmaskable interrupt
3	Hard fault	–1	All classes of fault, when the corresponding fault handler cannot be activated because it is currently disabled or masked by exception masking
4	MemManage	Settable	Memory management fault; caused by MPU violation or invalid accesses (such as an instruction fetch from a nonexecutable region)
5	Bus fault	Settable	Error response received from the bus system; caused by an instruction prefetch abort or data access error
6	Usage fault	Settable	Usage fault; typical causes are invalid instructions or invalid state transition attempts (such as trying to switch to ARM state in the Cortex-M3)
7–10	—	—	Reserved
11	SVC	Settable	Supervisor call via SVC instruction
12	Debug monitor	Settable	Debug monitor
13	—	—	Reserved
14	PendSV	Settable	Pendable request for system service
15	SYSTICK	Settable	System tick timer
16–255	IRQ	Settable	IRQ input #0–239

3a. Explain the operating modes of Cortex M3 with the help of neat diagrams indicating the switching of states on the occurrence of exceptions.



The Cortex-M3 processor supports two modes and two privilege levels: When the processor is running in thread mode, it can be in either the privileged or user level, but handlers can only be in the privileged level. When the processor exits reset, it is in thread mode, with privileged access rights. In the user access level (thread mode), access to the system control space (SCS)—a part of the memory region for configuration registers and debugging components—is blocked. Furthermore, instructions that access special registers (such as MSR, except when accessing APSR) cannot be

used. If a program running at the user access level tries to access SCS or special registers, a fault exception will occur. Software in a privileged access level can switch the program into the user access level using the control register. When an exception takes place, the processor will always switch to a privileged state and return to the previous state when exiting the exception handler. A user program cannot change back to the privileged state directly by writing to the control register. It has to go through an exception handler that programs the control register to switch the processor back into privileged access level when returning to thread mode.



3b. List the applications of Cortex M3

With its high performance and high code density and small silicon footprint, the Cortex-M3 processor is ideal for a wide variety of applications:

- **Low-cost microcontrollers:** The Cortex-M3 processor is ideally suited for low-cost microcontrollers, which are commonly used in consumer products, from toys to electrical appliances. It is a highly competitive market due to the many well-known 8-bit and 16-bit microcontroller products on the market. Its lower power, high performance, and ease-of-use advantages enable embedded developers to migrate to 32-bit systems and develop products with the ARM architecture.
- **Automotive:** Another ideal application for the Cortex-M3 processor is in the automotive industry. The Cortex-M3 processor has very high-performance efficiency and low interrupt latency, allowing it to be used in real-time systems. The Cortex-M3 processor supports up to 240 external vectored interrupts, with a built-in interrupt controller with nested interrupt supports and an optional MPU, making it ideal for highly integrated and cost-sensitive automotive applications.
- **Data communications**: The processor's low power and high efficiency, coupled with instructions in Thumb-2 for bit-field manipulation, make the Cortex-M3 ideal for many communications applications, such as Bluetooth and ZigBee.
- **Industrial control:** In industrial control applications, simplicity, fast response, and reliability are key factors. Again, the Cortex-M3 processor's interrupt feature, low interrupt latency, and enhanced fault-handling features make it a strong candidate in this area.
- **Consumerproducts:** In many consumer products, a high-performance microprocessor (or several of them) is used. The Cortex-M3 processor, being a small processor, is highly efficient and low in power and supports an MPU enabling complex software to execute while providing robust memory protection.

4a. Explain the Program Status registers with bit pattern.

The PSRs are subdivided into three status registers:

- Application Program Status register (APSR)
- Interrupt Program Status register (IPSR)
- Execution Program Status register (EPSR)

The three PSRs can be accessed together or separately using the special register access instructions MSR and MRS. When they are accessed as a collective item, the name xPSR is used. You can read the PSRs using the MRS instruction. You can also change the APSR using the MSR instruction, but EPSR and IPSR are read-only.

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
xPSR	N	Z	C	V	Q	ICI/IT	T			ICI/IT		Exception number				

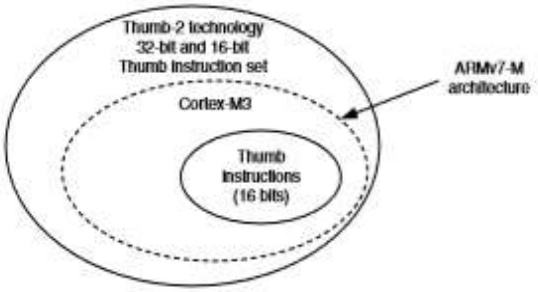
Bit Description

- N - Negative
- Z - Zero
- C - Carry/borrow
- V - Overflow
- Q - Sticky saturation flag
- ICI/IT - Interrupt-Continuable Instruction (ICI) bits, IF-THEN instruction status bit.
- T - Thumb state, always 1; trying to clear this bit will cause a fault exception Exception number Indicates which exception the processor is handling.

4b. Explain Thumb2 technology.

The Thumb-2 technology extended the Thumb Instruction Set Architecture (ISA) into a highly efficient and powerful instruction set that delivers significant benefits in terms of ease of use, code size, and performance. The extended instruction set in Thumb-2 is a superset of the previous 16-bit Thumb instruction set, with additional 16-bit instructions alongside 32-bit instructions.It allows more complex operations to be carried out in the Thumb state, thus allowing higher efficiency by reducing the number of states switching between ARM state and Thumb state.Focused on small memory system devices such as microcontrollers and reducing the size of the processor, the Cortex-M3 supports only the Thumb-2 (and traditional Thumb) instruction set.Instead of using ARM instructions for some operations, as in traditional ARM processors, it uses the Thumb-2 instruction set for all operations. As a result, the Cortex-M3 processor is not backward compatible with traditional ARM processors. With support for both 16-bit and 32-bit instructions in the Thumb-2 instruction set, there is no need to switch the processor between Thumb state (16-bit instructions) and ARM state (32-bit instructions). For example, in ARM7 or ARM9 family processors, you might need to switch to ARM state if you want to carry out complex calculations or a large number of conditional operations and good performance is needed, whereas in the Cortex-M3 processor, you can mix 32-bit instructions with 16-

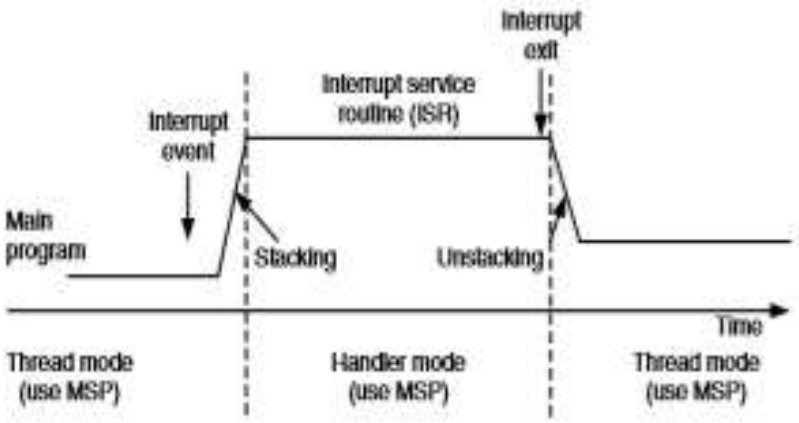
bit instructions without switching state, getting high code density and high performance with no extra complexity.The Thumb-2 instruction set is a very important feature of the ARMv7 architecture.



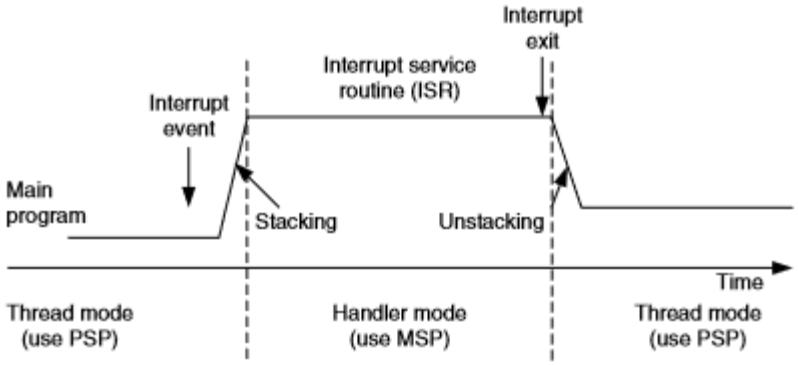
5a. Explain the 2-level Stack model in Cortex M3.

The Cortex-M3 has two SPs: the MSPS and the PSP. The SP register to be used is controlled by the control register bit 1 (CONTROL[1].When CONTROL[1] is 0, the MSP is used for both thread mode and handler mode. In this arrangement, the main program and the exception handlers share the same stack memory region. This is the default setting after power-up.

When the CONTROL[1] is 1, the PSP is used in thread mode. In this arrangement, the main program and the exception handler can have separate stack memory regions. This can prevent a stack error in a user application from damaging the stack used by the OS. Note that in this situation, the automatic stacking and unstacking mechanism will use PSP, whereas stack operations inside the handler will use MSP. It is possible to perform read/write operations directly to the MSP and PSP, without any confusion of which R13 you are referring to.

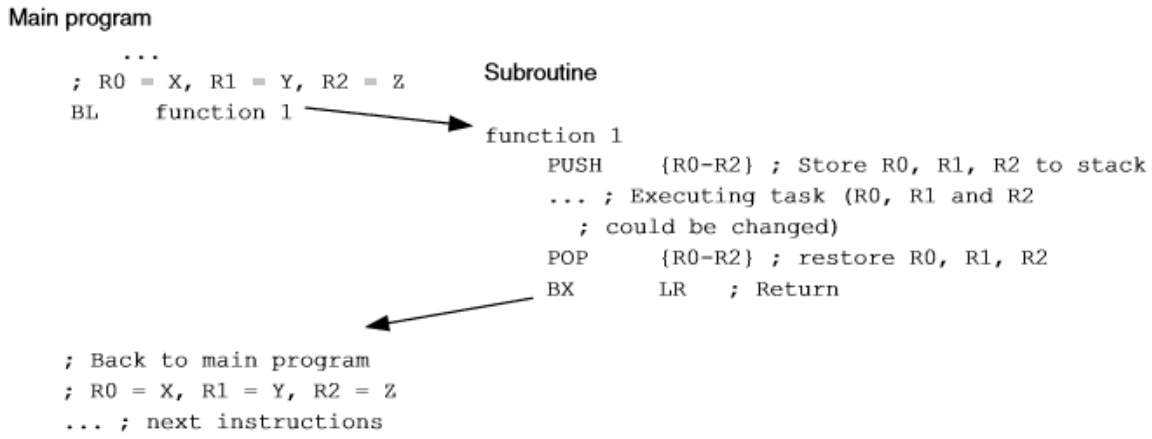


CONTROL[1]=0: Both Thread Level and Handler Use Main Stack.



CONTROL[1]=1: Thread Level Uses Process Stack and Handler Uses Main Stack.

5b What is Stack? Explain the multiple register Stack operation.

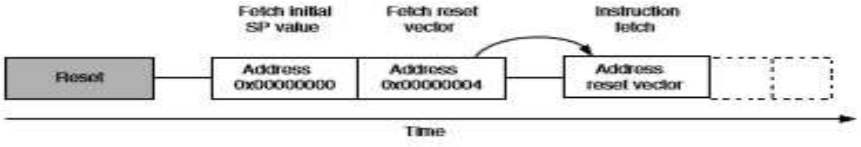


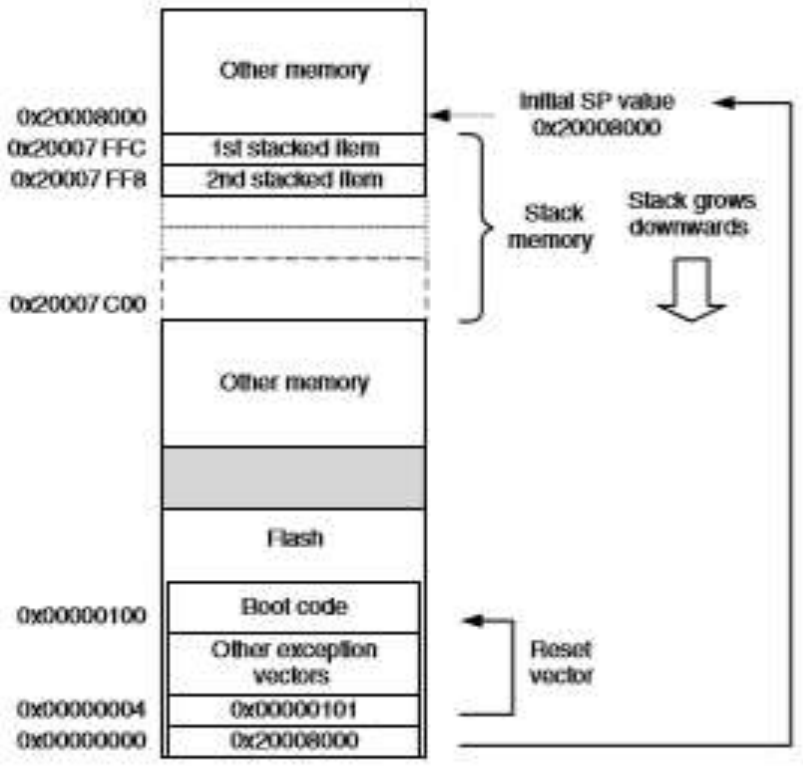
6a Explain the reset sequence with the help of memory map.

After the processor exits reset, it will read two words from memory:

- Address 0x00000000: Starting value of R13 (the SP).
- Address 0x00000004: Reset vector (the starting address of program execution; LSB should be set to 1 to indicate Thumb state).

This differs from traditional ARM processor behavior. Previous ARM processors executed program code starting from address 0x0. Furthermore, the vector table in previous ARM devices was instructions (you have to put a branch instruction there so that your exception handler can be put in another location). In the Cortex-M3, the initial value for the MSP is put at the beginning of the memory map, followed by the vector table, which contains vector address values.(The vector table can be relocated to another location later, during program execution.) In addition, the contents of the vector table are address valuesnot branch instructions. The first vector in the vector table (exception type 1) is the reset vector, which is the second piece of data fetched by the processor after reset.





6b. Explain the Control register .

The control register is used to define the privilege level and the SP selection. This register has 2 bits.

CONTROL[1]:In the Cortex-M3, the CONTROL[1] bit is always 0 in handler mode. However, in the thread or base level, it can be either 0 or 1.This bit is writable only when the core is in thread mode and privileged. In the user state or handler mode, writing to this bit is not allowed. Aside from writing to this register, another way to change this bit is to change bit 2 of the LR when in exception return.

CONTROL[0]: The CONTROL[0] bit is writable only in a privileged state. Once it enters the user state, the only way to switch back to privileged is to trigger an interrupt and change this in the exception handler.

To access the control register in assembly, the MRS and MSR instructions are used:

MRS r0, CONTROL ; Read CONTROL register into R0

MSR CONTROL, r0 ; Write R0 into CONTROL register

To access the control register in C, the following CMSIS functions are available in CMSIS compliant device driver libraries:

x = __get_CONTROL(); // Read the current value of CONTROL

__set_CONTROL(x); // Set the CONTROL value to x