# ARM CORTEXM ARCHITECTURE

By: Muhammad ElZeiny

# CORTEXM4 PROGRAMING MODEL

# CORTEXM4 PRIVILEGE LEVELS

## USER (UNPRIVILEGED)

*In this mode, software has the following restrictions:*

- Limited access to the MSR and MRS instructions and no use of the CPS instruction
- No access to the system timer, NVIC, or system control block
- Possibly restricted access to memory or peripherals

## PRIVILEGED

In this mode, software can use all the instructions and has access to all resources.

# CORTEXM4 MODES OF OPERATION

## THREAD MODE

Used **to execute application software**.
The processor enters Thread mode when it comes out of reset.

***Privilege \ Unprivileged***

## HANDLER MODE

Used **to handle exceptions.**
When the processor has finished exception processing,
it returns to Thread mode.

***Always Privilege.***

# CORTEXM4 CORE REGISTER

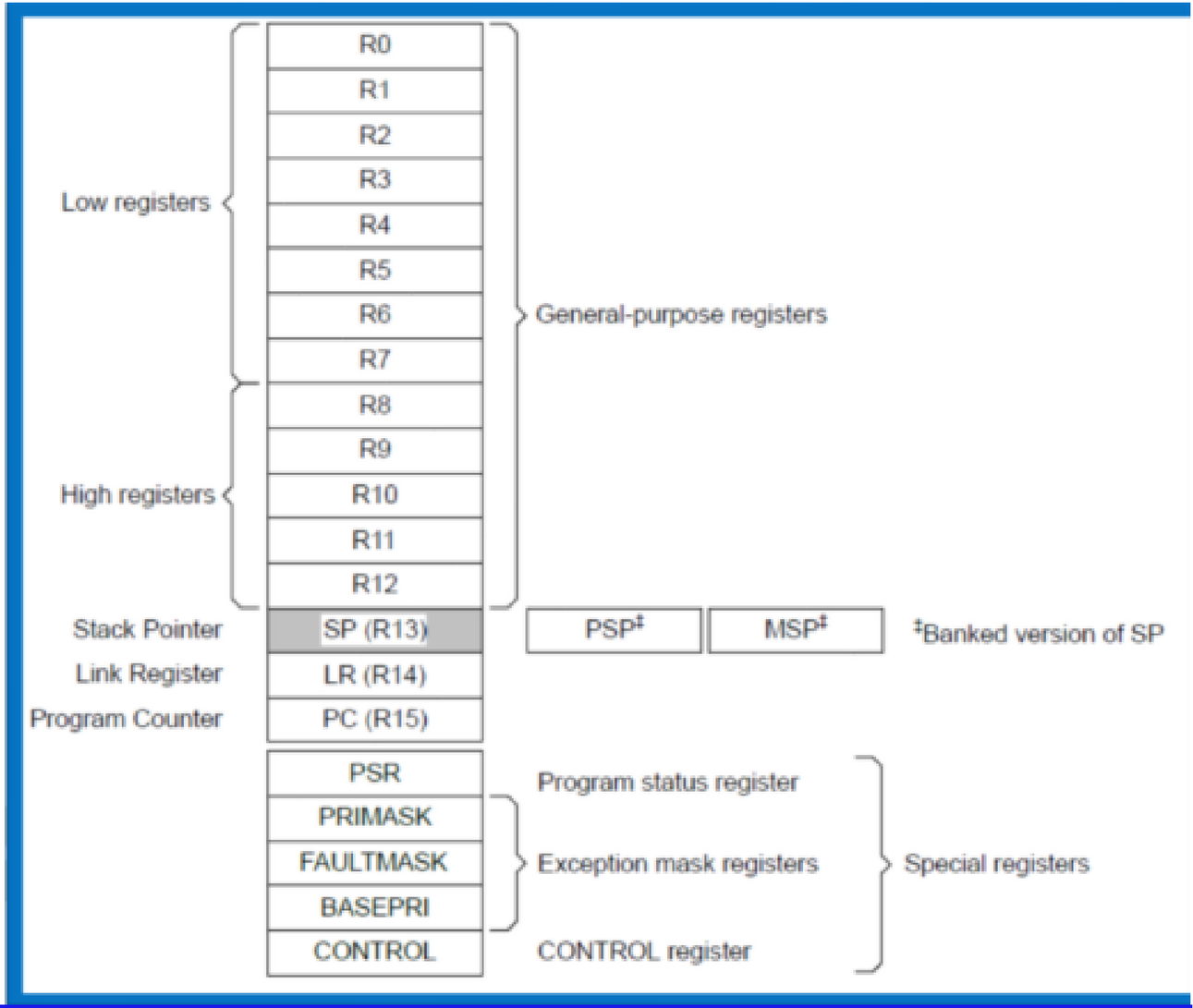
# CORTEXM4 CORE REGISTER

- **Stack pointer**

  • Register r13 is used as the Stack Pointer (SP).
  • The processor implements two stacks with a pointer for each held in independent register “Main SP” and “Process SP”.

- **Link register**

  • Register r14 is the subroutine Link Register (LR). The LR receives the return address from PC when a Branch and Link.
  • The LR is also used for exception return.
  • At all other times, you can treat r14 as a general-purpose register.



# CORTEXM4 CORE REGISTER

- **PRIMASK**

  • prevents activation of all exceptions with programmable priority.

- **FAULTMASK**

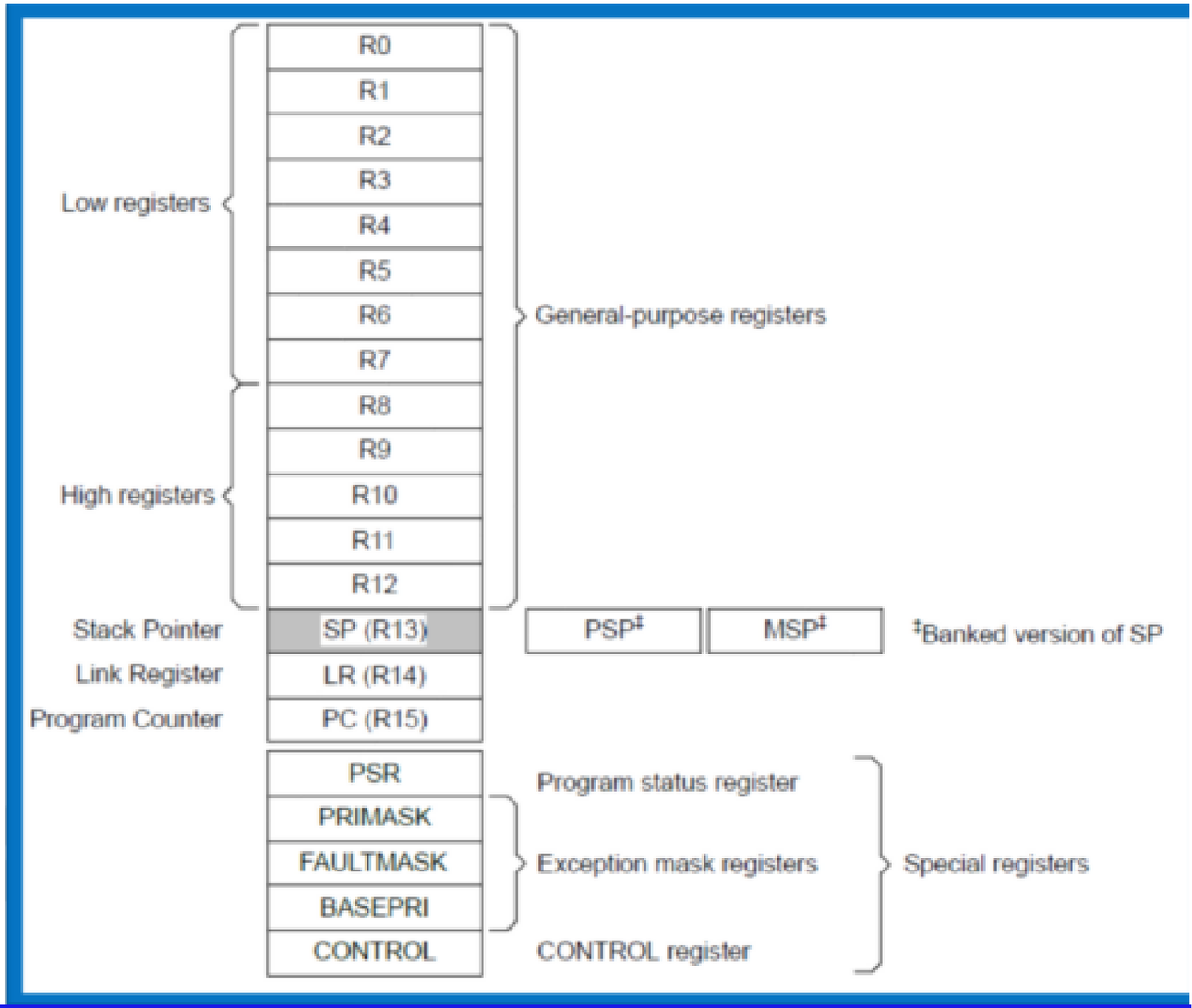
  • register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI).

- **Both PRIMASK and FAULTMASK**

  • This register is only accessible in privileged mode.

  • The MSR and MRS instructions are used to access the **PRIMASK** register, and

  • the CPS instruction may be used to change the value of the **PRIMASK** register.



# CORTEXM4 CORE REGISTER

- **PRIMASK**

  • prevents activation of all exceptions with programmable priority.

- **FAULTMASK**

  • register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI).

- **Both PRIMASK and FAULTMASK**

  • This register is only accessible in privileged mode.

  • The MSR and MRS instructions are used to access the **PRIMASK** register, and

  • the CPS instruction may be used to change the value of the **PRIMASK** register.

**CPS**

Change Processor State.

CPSIE i
f

**Syntax**

*CPSeffect iflags*

where:

*effect* Is one of:

| | |
|---|---|
| IE | Clears the special purpose register. |
| ID | Sets the special purpose register. |

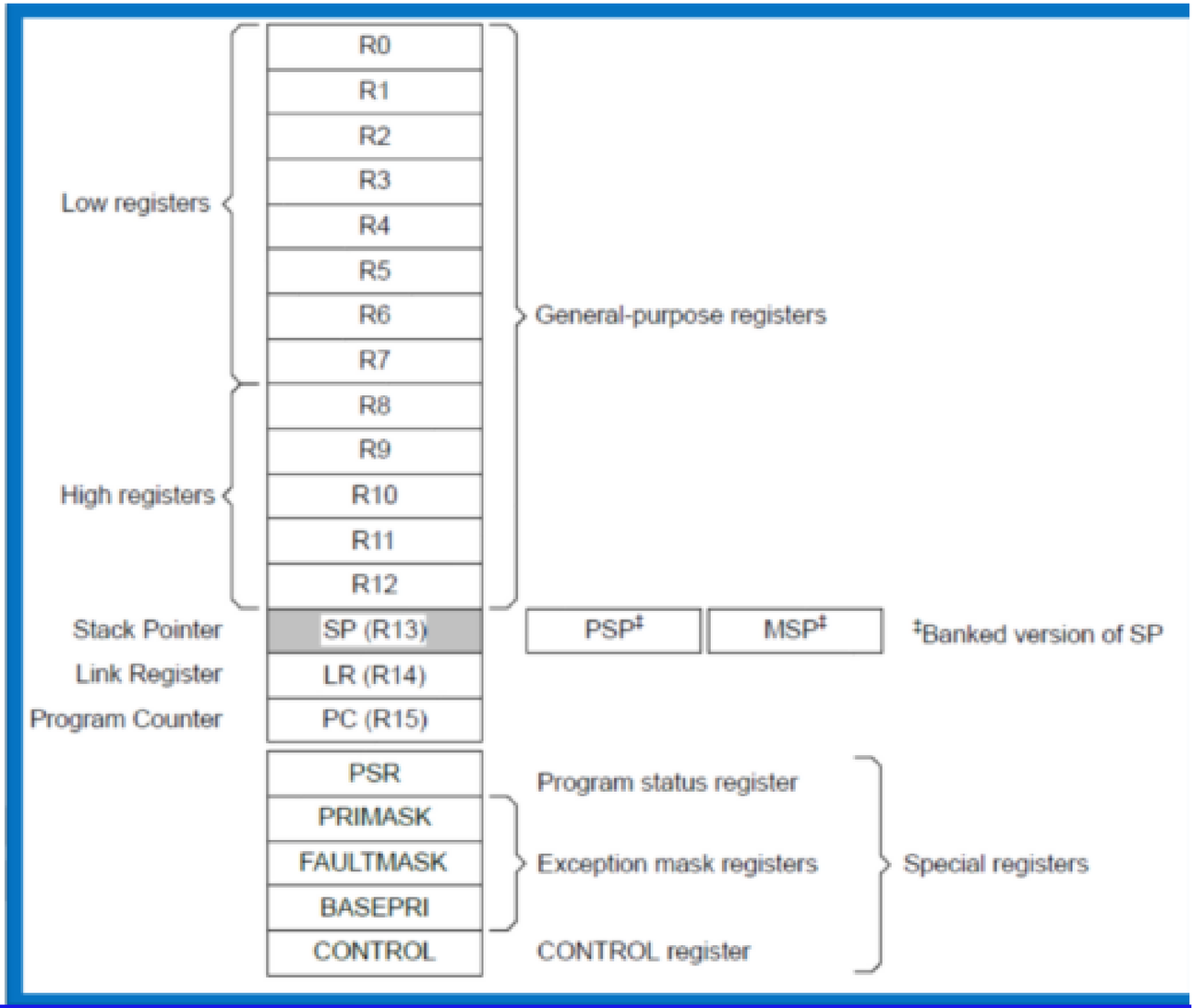
*iflags* Is a sequence of one or more flags:

| | |
|---|---|
| i | Set or clear PRIMASK. |
| f | Set or clear FAULTMASK. |

# CORTEXM4 CORE REGISTER

- **The CONTROL register controls**
  - the stack used
  - the privilege level for software execution when the processor is in Thread mode
  - indicates whether the FPU state is active.

- This register is only accessible in privileged mode.



# HOW TO SWITCH TO\FROM PRIVILEGED MODE

- In Thread mode, the **CONTROL** register controls whether software execution is privileged or unprivileged.

- In Handler mode, software execution is always privileged.

- Only privileged software can write to the **CONTROL** register !!

# CORTEX-M MEMORY MODEL

# CORTEX-M MEMORY MODEL

# CORTEX-M BUS INTERFACES

## ICODE MEMORY INTERFACE

Instruction fetches from Code memory space [0x00000000 to 0x1FFFFFFF] are performed over this bus

## DCODE MEMORY INTERFACE

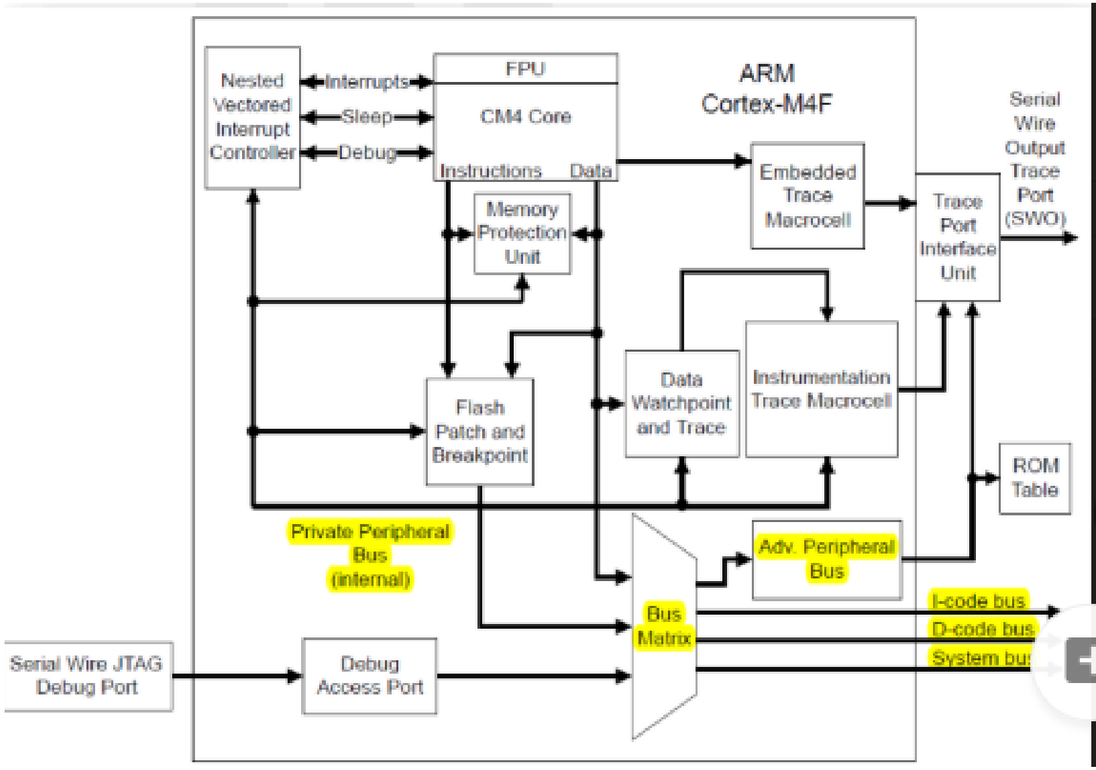
Data and debug accesses to Code memory space [0x00000000 to 0x1FFFFFFF] are performed over this bus

## SYSTEM INTERFACE

Instruction fetches, and data and debug accesses, to address ranges [0x20000000 to 0xDFFFFFFF] and [0xE0100000 to 0xFFFFFFFF] are performed over this bus.

## PRIVATE PERIPHERAL BUS (PPB)

Data and debug accesses to external PPB space, 0xE0040000 to 0xE00FFFFF, are performed over this bus.



# CORTEX-M BUS INTERFACES

# CORTEX-M BUS INTERFACES

## ICODE MEMORY INTERFACE

Instruction fetches from Code memory space [0x00000000 to 0x1FFFFFFF] are performed over this bus

## DCODE MEMORY INTERFACE

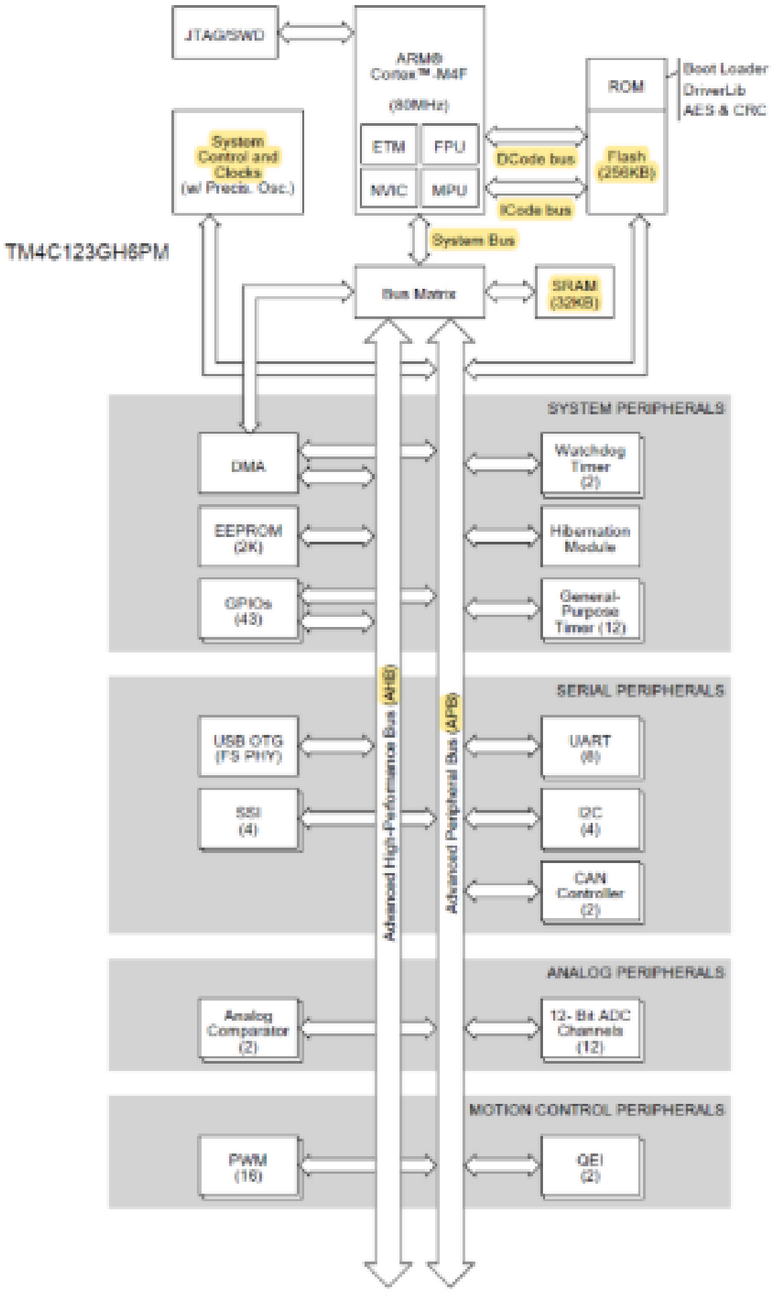
Data and debug accesses to Code memory space [0x00000000 to 0x1FFFFFFF] are performed over this bus

## SYSTEM INTERFACE

Instruction fetches, and data and debug accesses, to address ranges [0x20000000 to 0xDFFFFFFF] and [0xE0100000 to 0xFFFFFFFF] are performed over this bus.

## PRIVATE PERIPHERAL BUS (PPB)

Data and debug accesses to external PPB space, 0xE0040000 to 0xE00FFFFF, are performed over this bus.



# Memory Map

# BIT BANDING

# MEMORY REGIONS, TYPES AND ATTRIBUTES

**Normal:** The processor can re-order transactions for efficiency and perform speculative reads.

**Device:** The processor preserves transaction order relative to other transactions to Device or

**Strongly Ordered memory:** The processor preserves transaction order relative to all other transactions.

# BEHAVIOR OF MEMORY ACCESSES

| Address Range | Memory Region | Memory Type | Execute Never (XN) | Description |
|---|---|---|---|---|
| 0x0000.0000 - 0x1FFF.FFFF | Code | Normal | - | This executable region is for program code. Data can also be stored here. |
| 0x2000.0000 - 0x3FFF.FFFF | SRAM | Normal | - | This executable region is for data. Code can also be stored here. This region includes bit band and bit band alias areas (see Table 2-6 on page 97). |
| 0x4000.0000 - 0x5FFF.FFFF | Peripheral | Device | XN | This region includes bit band and bit band alias areas (see Table 2-7 on page 98). |
| 0x6000.0000 - 0x9FFF.FFFF | External RAM | Normal | - | This executable region is for data. |
| 0xA000.0000 - 0xDFFF.FFFF | External device | Device | XN | This region is for external device memory. |
| 0xE000.0000- 0xE00F.FFFF | Private peripheral bus | Strongly Ordered | XN | This region includes the NVIC, system timer, and system control block. |
| 0xE010.0000- 0xFFFF.FFFF | Reserved | - | - | - |

# REASONS OF MEMORY SYSTEM ORDERING

- The processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- The processor has multiple bus interfaces.
- Memory or devices in the memory map have different wait states.
- Some memory accesses are buffered or speculative.

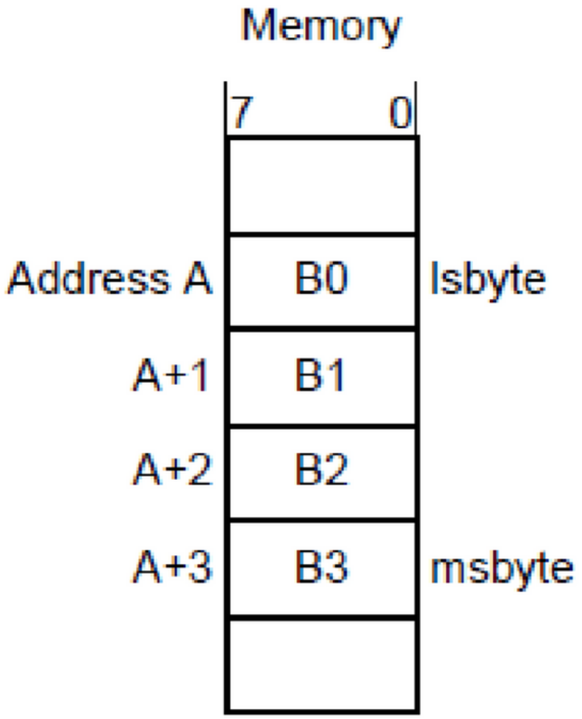
# SOFTWARE ORDERING OF MEMORY ACCESSES

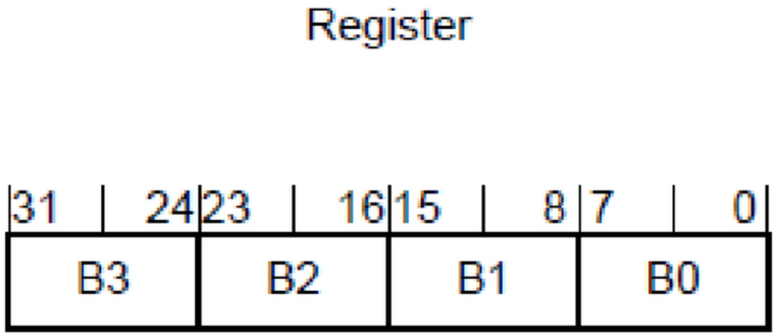
- if the order of memory accesses is critical, software must include memory barrier instructionsto force that ordering.
    - The Data Memory Barrier (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions.
    - The Data Synchronization Barrier (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute.
    - The Instruction Synchronization Barrier (ISB) instruction ensures that the effect of all completed memory transactions is recognizable by subsequent instructions.

# DATA STORAGE

- Data is stored in little-endian format.

# Memory System Access Ordering




# Data Storage

Data is stored in little-endian format.

# Synchronization Primitives

- The Cortex-M4F instruction set includes pairs of synchronization primitives which provide a non blocking mechanism that a thread or process can use to obtain exclusive access to a memory location.

  1. **Load-Exclusive instruction**, which is used to read the value of a memory location and requests exclusive access to that location.

  2. **Store-Exclusive instruction**, which is used to attempt to write to the same memory location and returns a status bit to a register.

# CORTEX-M4

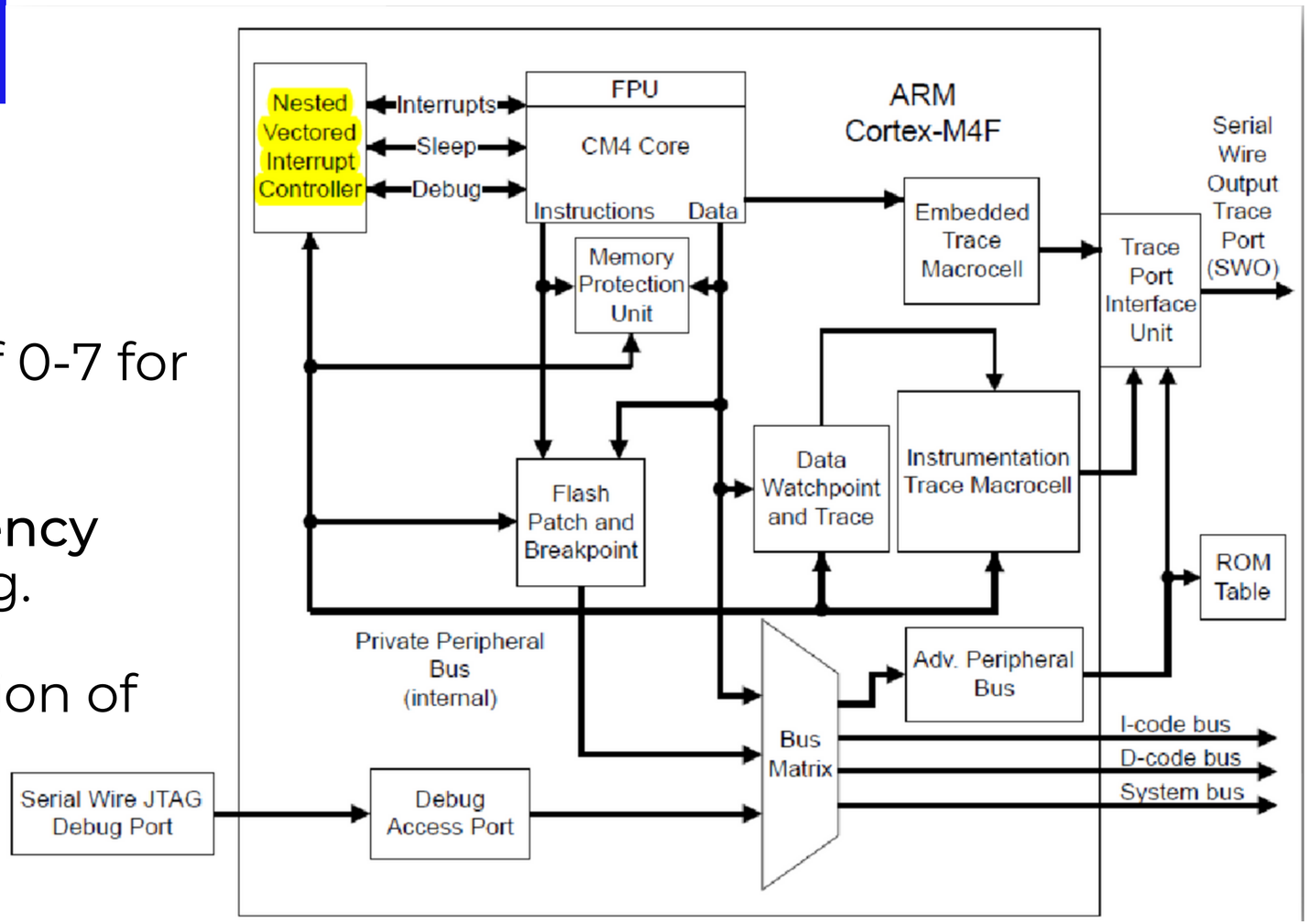
PERIPHERALS OVERVIEW

# SYSTEM TIMER

- Provides a simple, **24-bit** clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.
- The SysTick counter **runs on** either the system clock or the precision internal oscillator (PIOSC) divided by 4.
- When the processor is halted for debugging, the counter **does not** decrement.

# NESTED VECTOR INTERRUPT CONTROLLER (NVIC)

- Supports up to **78** interrupts.
- A **programmable priority** level of 0-7 for each interrupt.
- Provide mechanism for **Low-latency** exception and interrupt handling.
- Supports Level and pulse detection of interrupt signals.



# SYSTEM CONTROL BLOCK (SCB)

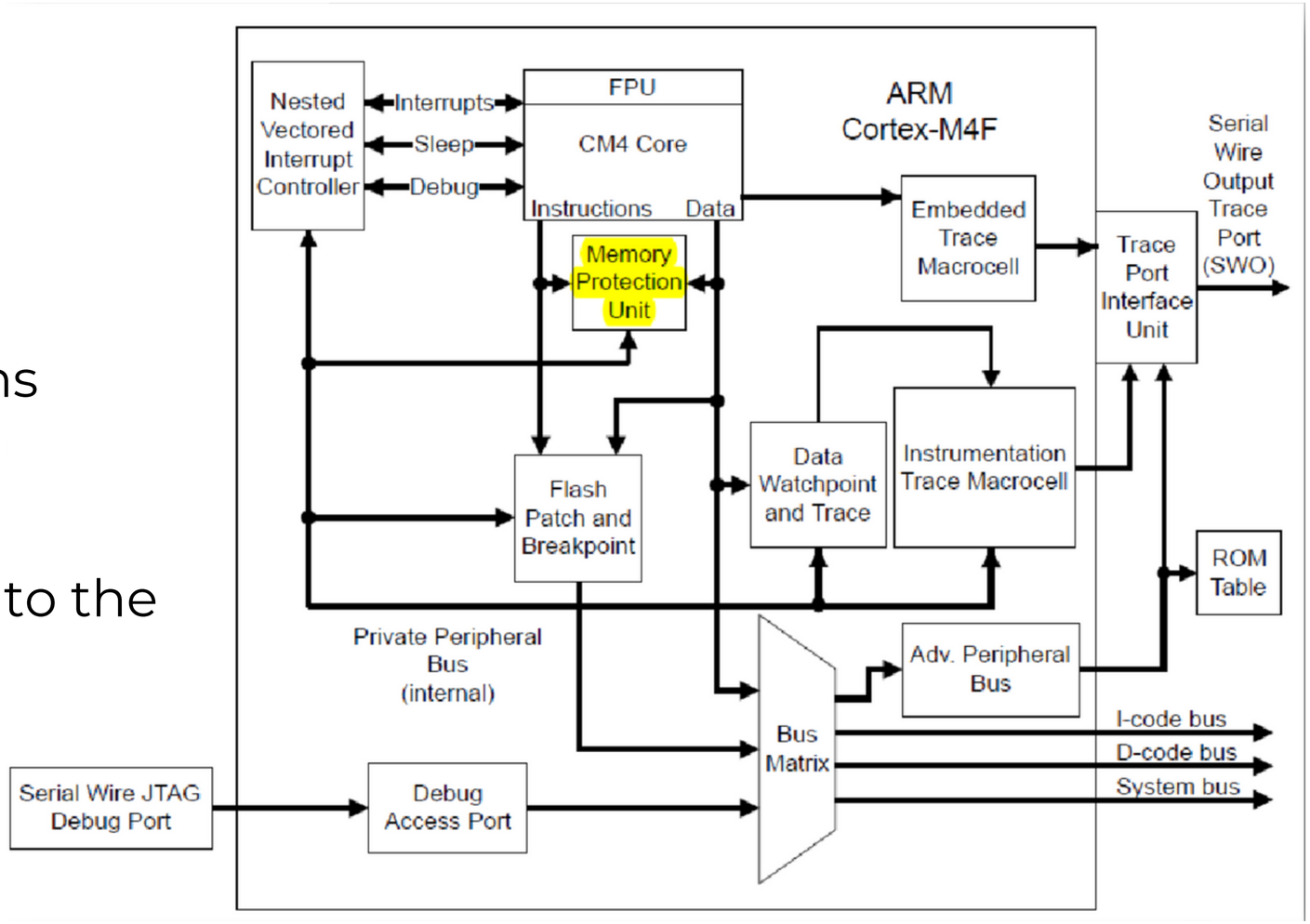
*Provides the followings*

- System implementation information
- System control and configuration
- Reporting of the system exceptions.

# MEMORY PROTECTION UNIT (MPU)

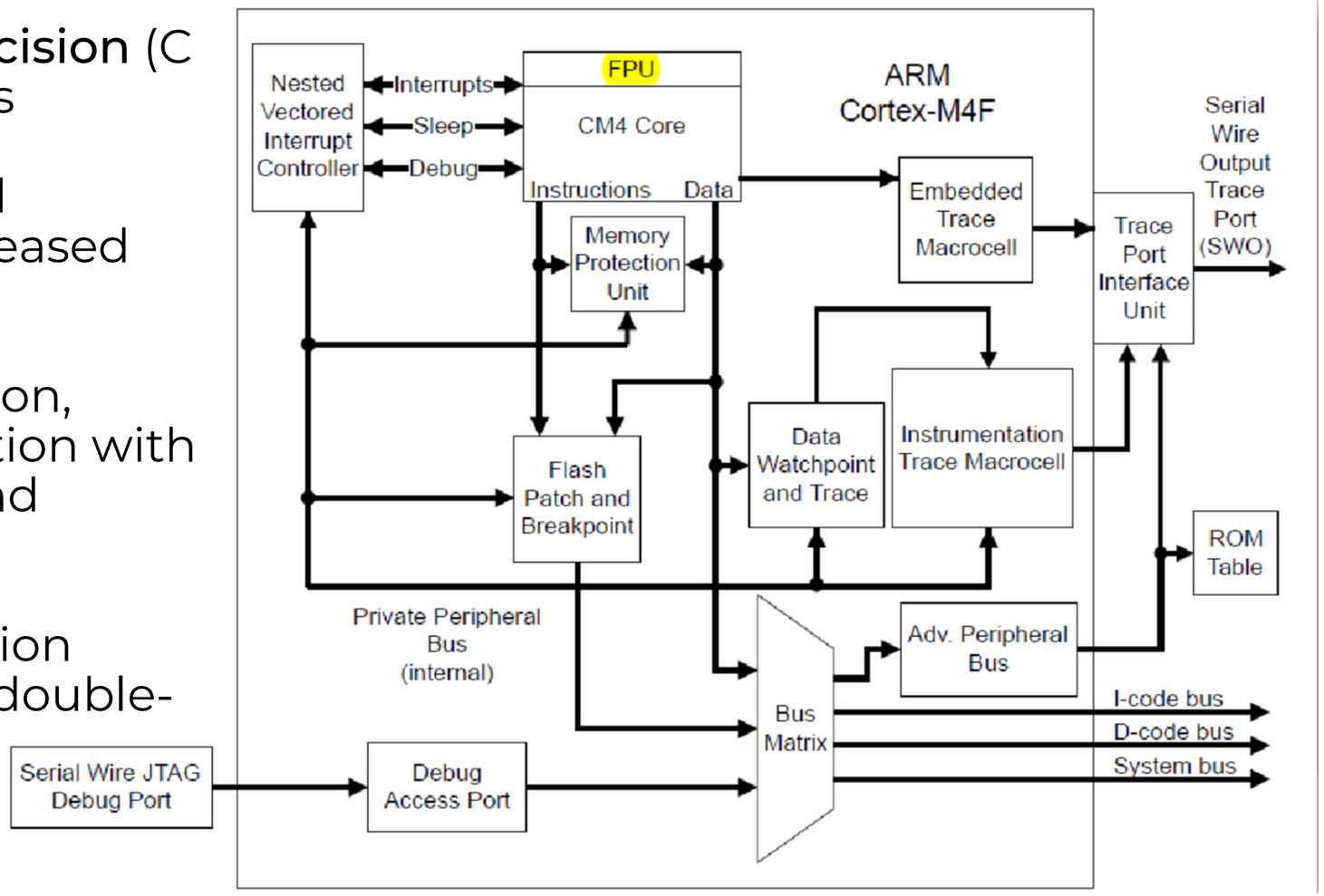
*Provides full support for*

- protection regions
- overlapping protection regions
- access permissions
- exporting memory attributes to the system



# FLOATING-POINT UNIT (FPU)

- 32-bit instructions for **single-precision** (C float) data-processing operations
- Support **Combined multiply and accumulate** instructions for increased precision
- A **Hardware** support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root
- 32 dedicated 32-bit single-precision **registers**, also addressable as 16 double-word registers
- Three stage **pipeline**



# Q&A

# QUIZ

• Link Register is only used to store return address.

❑True
❑False

• Special register can be written by software by accessing an specific address in memory map.

❑True
❑False

• The Cpu in Handler mode is always privileged.
❑True
❑False

• Process Stack pointer is not necessary for system without OS

❑True
❑False

# TASK

- Platform_Types, STD_Types and Bit_Math.h
- Write Cpu_Driver with the following APIs
    - Void CpuDriver_EnableGlobalInterrupt(void)
    - Void CpuDriver_DisableGlobalInterrupt(void)
    - Void CpuDriver_ StartCriticalSection(void)
    - Void CpuDriver_ StopCriticalSection(void)