ELC4438: Embedded System Design
ARM Cortex-M Architecture I

Liang Dong

Electrical and Computer Engineering
Baylor University

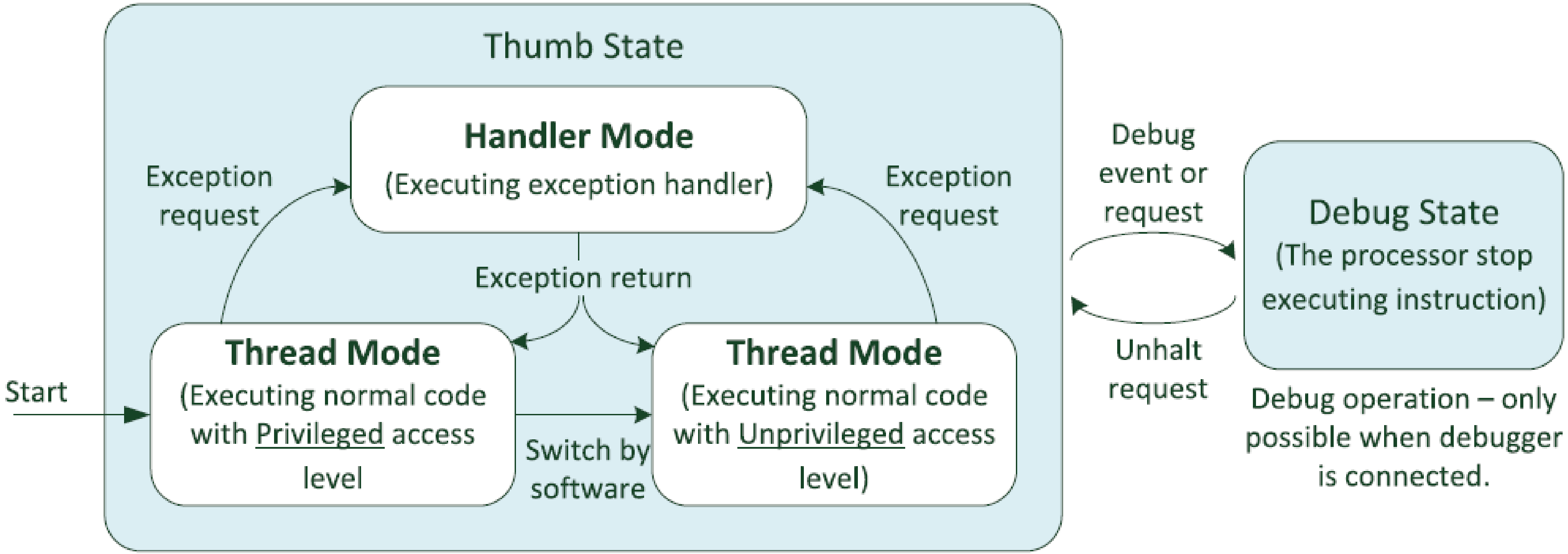
Introduction to the architecture

- ARMv7-M Architecture Reference Manual
  - processor's behavior
  - instruction set
  - memory system
  - debug support

- *The Manual* is for experts like processor designers, designers of C compilers, and development tools.

- We need it from a software developer's perspective.

Programmer’s model



Thumb State

Handler Mode
(Executing exception handler)

Exception request

Exception request

Exception return

Start

Thread Mode
(Executing normal code with Privileged access level

Switch by software

Thread Mode
(Executing normal code with Unprivileged access level)

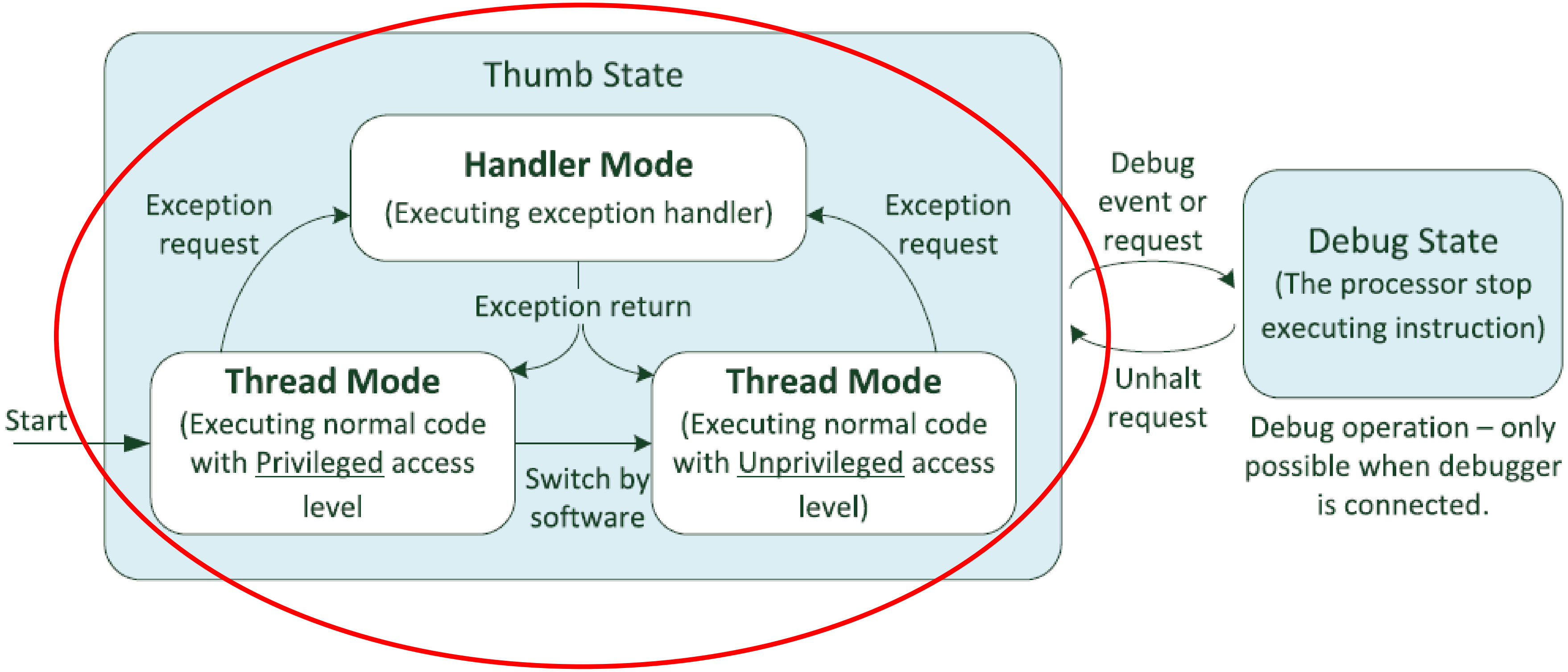
Debug event or request

Unhalt request

Debug State
(The processor stop executing instruction)

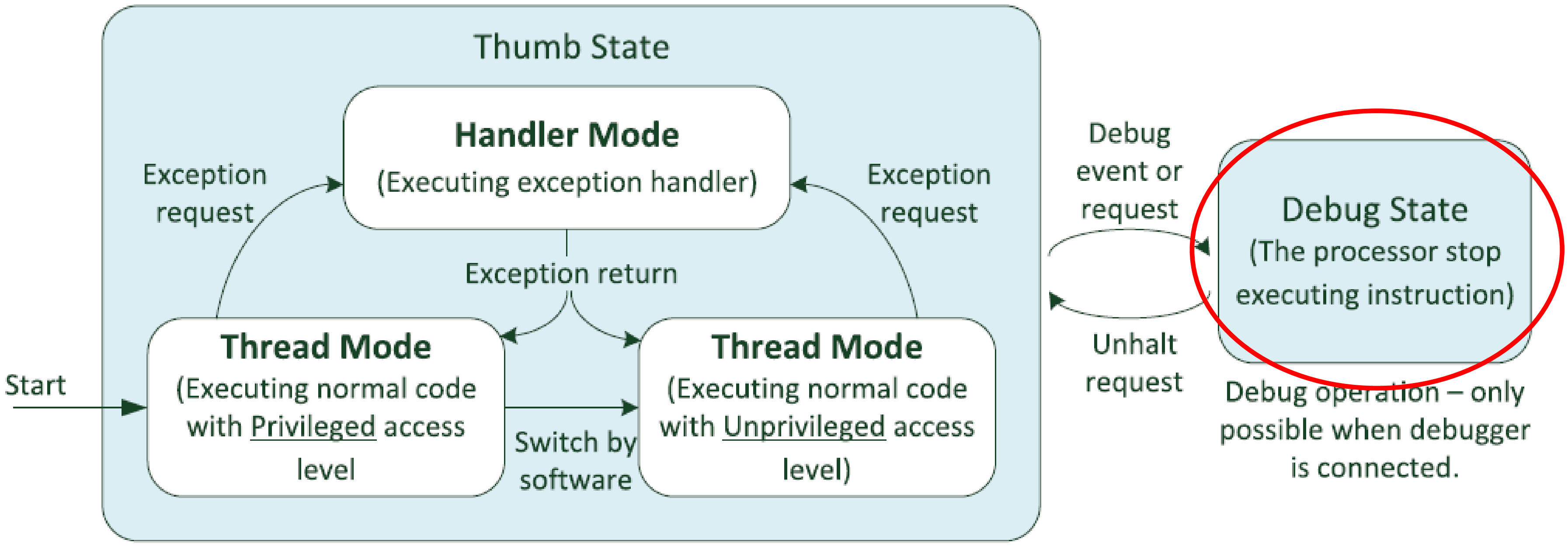
Debug operation – only possible when debugger is connected.

Programmer’s model


Thumb State
Handler Mode
(Executing exception handler)
Exception request
Exception request
Exception return
Thread Mode
(Executing normal code with Privileged access level
Thread Mode
(Executing normal code with Unprivileged access level)
Start
Switch by software
Debug event or request
Unhalt request
Debug State
(The processor stop executing instruction)
Debug operation – only possible when debugger is connected.

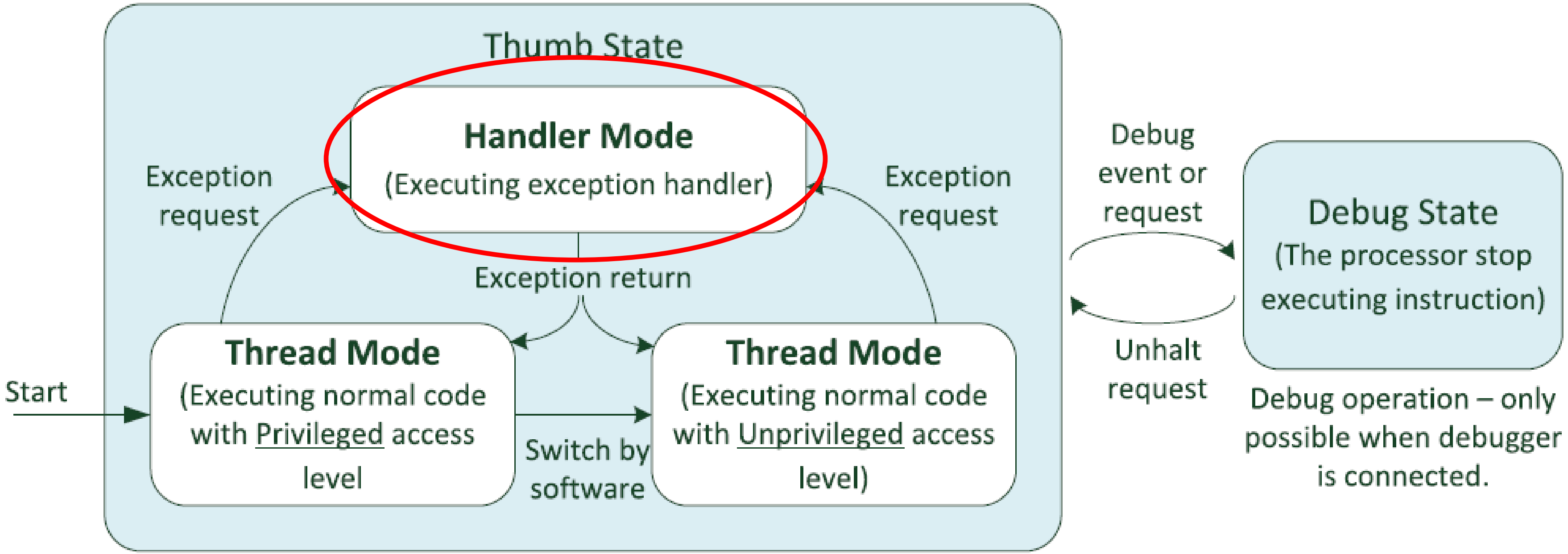
Thumb state: If the processor is running program code (Thumb instructions), it is in the Thumb state.

Programmer’s model


Thumb State
Handler Mode
(Executing exception handler)
Exception request
Exception request
Exception return
Thread Mode
(Executing normal code with Privileged access level
Thread Mode
(Executing normal code with Unprivileged access level)
Start
Switch by software
Debug event or request
Unhalt request
Debug State
(The processor stop executing instruction)
Debug operation – only possible when debugger is connected.

Debug state: When the processor is halted, it enters debug state and stops executing instructions.

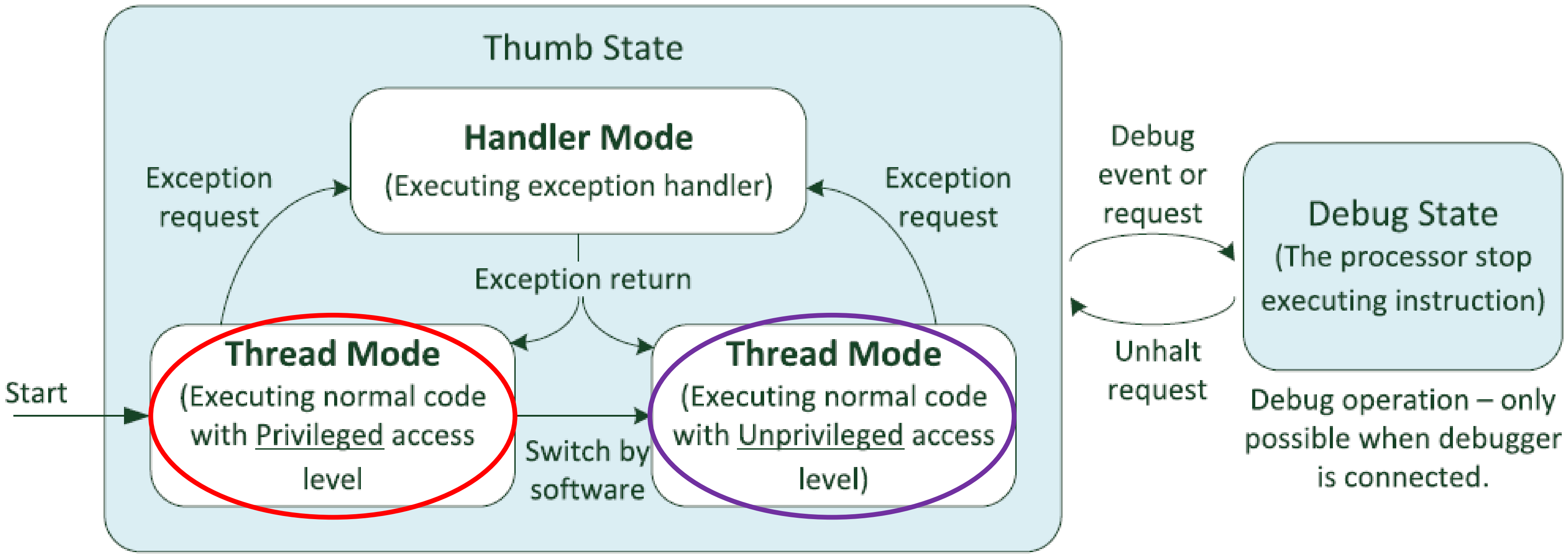
Programmer’s model



Thumb State
Handler Mode
(Executing exception handler)
Exception request
Exception request
Exception return
Start
Thread Mode
(Executing normal code with Privileged access level
Switch by software
Thread Mode
(Executing normal code with Unprivileged access level)
Debug event or request
Unhalt request
Debug State
(The processor stop executing instruction)
Debug operation – only possible when debugger is connected.

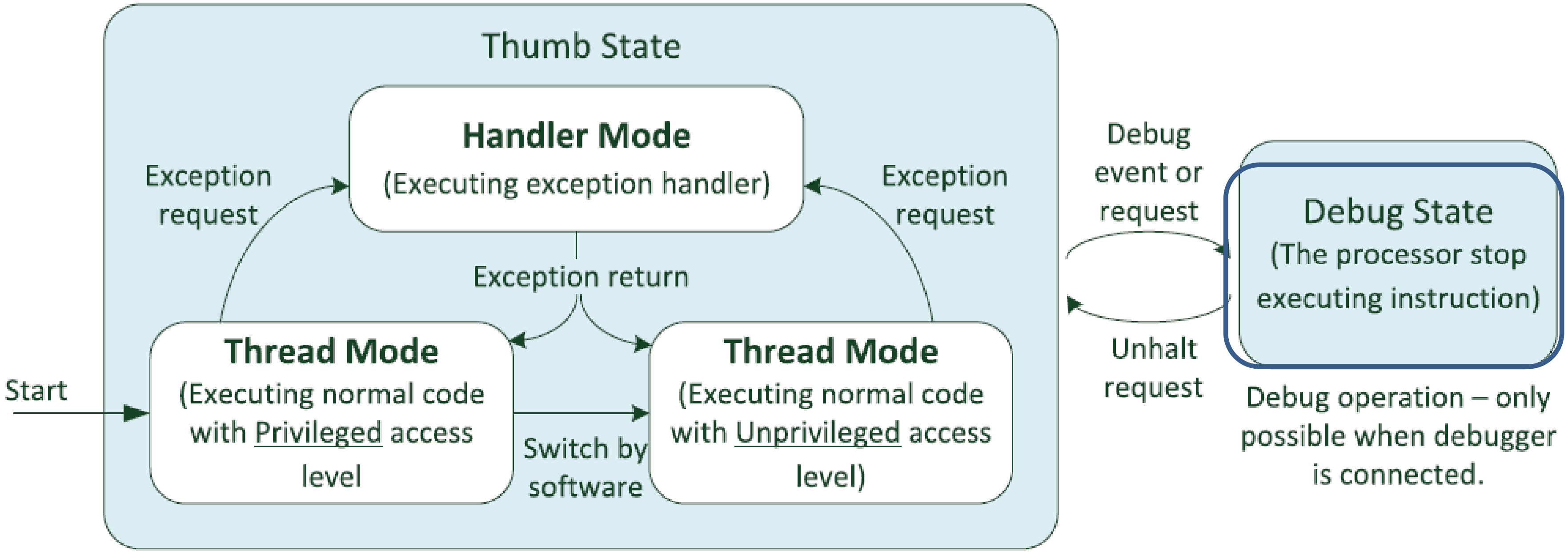
Handler mode: Executing an exception handler such as an Interrupt Service Routine (ISR). When in handler mode, the processor always has privileged access level.

Programmer's model


Thumb State
Handler Mode
(Executing exception handler)
Exception request
Exception request
Exception return
Start
Thread Mode
(Executing normal code with Privileged access level
Switch by software
Thread Mode
(Executing normal code with Unprivileged access level)
Debug event or request
Unhalt request
Debug State
(The processor stop executing instruction)
Debug operation – only possible when debugger is connected.

Thread mode: When executing normal application code, the processor can be either in privileged access level or unprivileged access level.

Programmer's model


Thumb State
Handler Mode
(Executing exception handler)
Exception request
Exception request
Exception return
Thread Mode
(Executing normal code with Privileged access level
Start
Switch by software
Thread Mode
(Executing normal code with Unprivileged access level)
Debug event or request
Unhalt request
Debug State
(The processor stop executing instruction)
Debug operation – only possible when debugger is connected.

Access to processor registers,
system memory, and
peripherals inside and outside the processor

Registers

- Most of the registers are grouped in a unit -register bank

- Each data processing instruction specifies the operation required, the source register(s), and the destination register(s) if applicable.

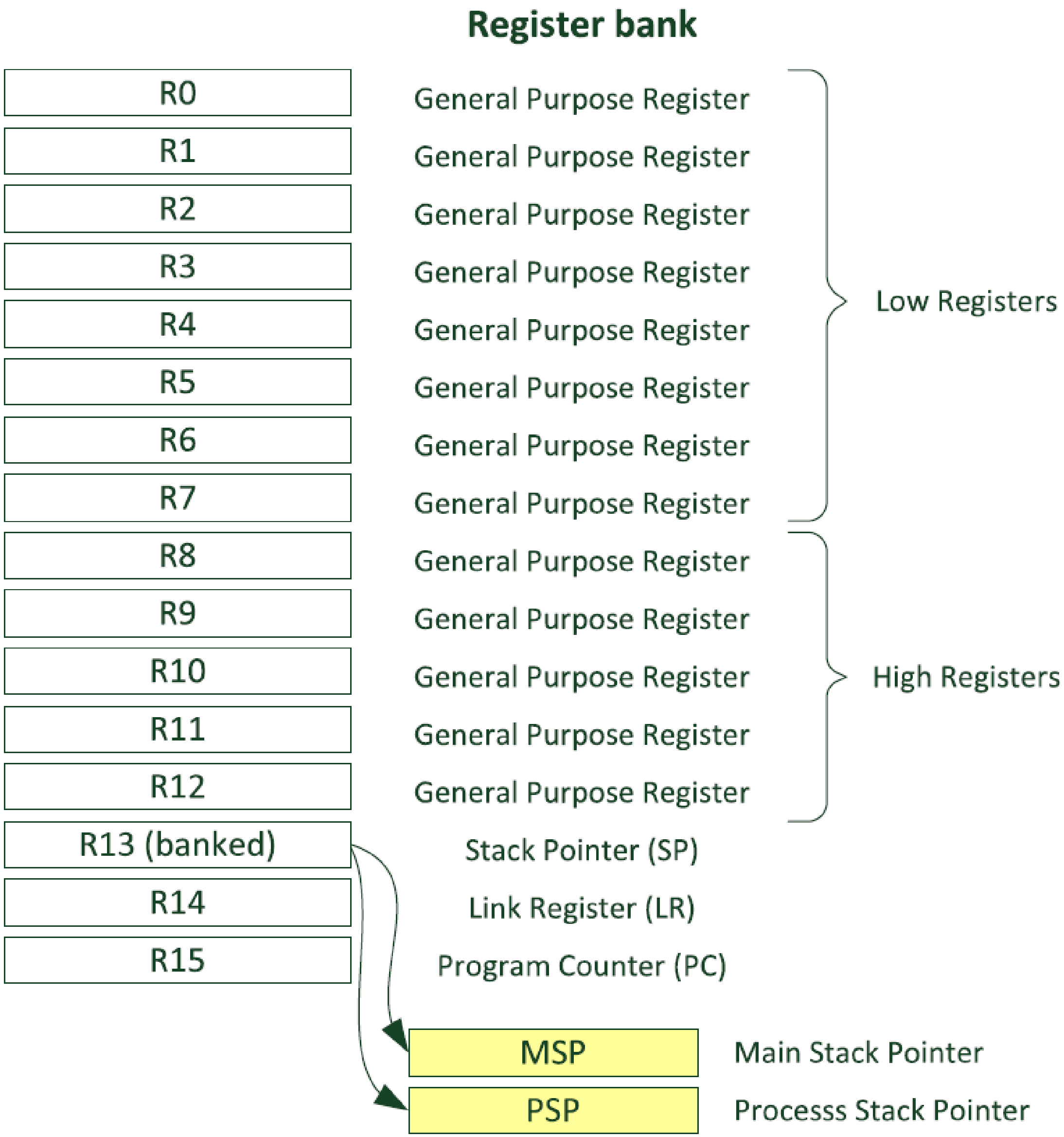
- Load-store architecture:
  - if data in memory is to be processed, it has to be loaded from the memory to registers in the register bank,

    processed inside the processor, and

    written back to the memory, if needed.

Register Bank

- The register bank in the Cortex-M3 and Cortex-M4 processors has 16 registers.



Register bank

R0	General Purpose Register	Low Registers
R1	General Purpose Register	
R2	General Purpose Register	
R3	General Purpose Register	
R4	General Purpose Register	
R5	General Purpose Register	
R6	General Purpose Register	
R7	General Purpose Register	
R8	General Purpose Register	High Registers
R9	General Purpose Register	
R10	General Purpose Register	
R11	General Purpose Register	
R12	General Purpose Register	
R13 (banked)	Stack Pointer (SP)	
R14	Link Register (LR)	
R15	Program Counter (PC)	

MSP	Main Stack Pointer
PSP	Processs Stack Pointer

Stack Pointer (SP) – R13

- Stack Pointer is used for accessing the stack memory via PUSH and POP operations.
- Main Stack Pointer (MSP) is the default Stack Pointer.
- Process Stack Pointer (PSP) can only be used in Thread Mode. (an embedded OS is involved)
- The selection of Stack Pointer is determined by a special register called CONTROL.
- In normal programs, only one of these Stack Pointers will be visible.

Link Register (LR) – R14

- Link Register (LR) is used for holding the return address when calling a function or subroutine.

- Return - loading the value of LR into the Program Counter (PC)

- If a function needs to call another function or subroutine, it needs to save the value of LR in the stack first.

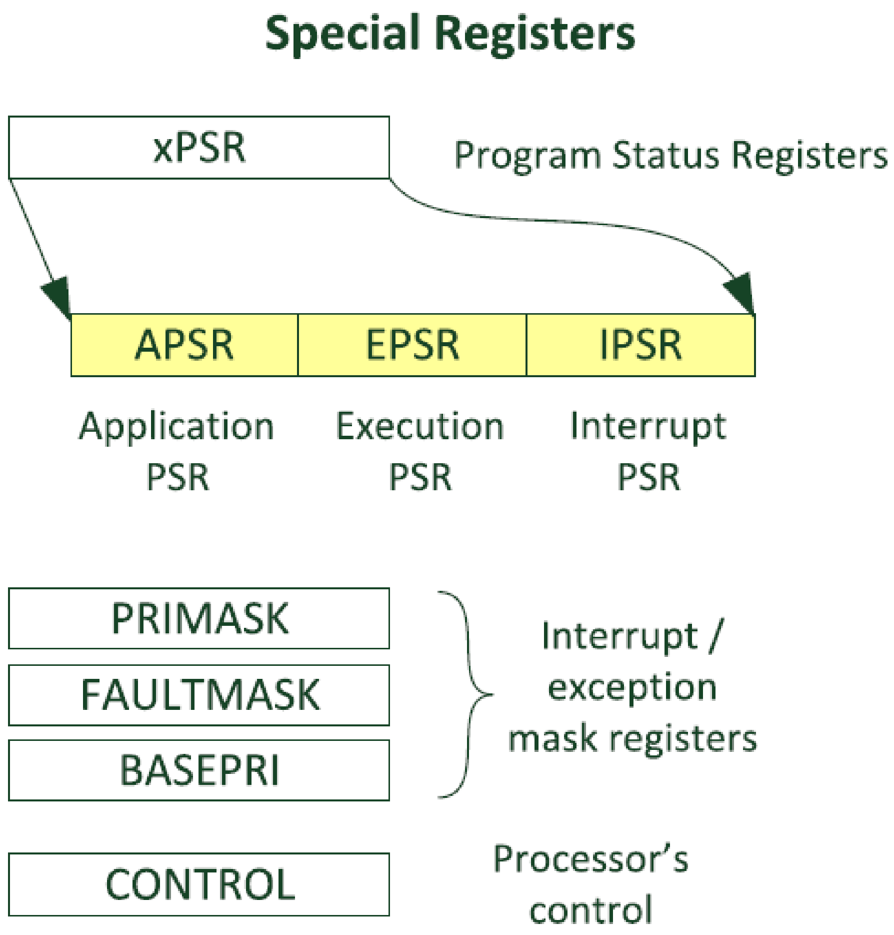
Program Counter (PC) – R15

- Reading PC returns the current instruction address.

- Writing to PC causes a branch operation.

- In most cases, branches and calls are handled by instructions dedicated to such operations. It is less common to use data processing instructions to update the PC.

Special Registers



Interrupt Mask Register

- The PRIMASK, FAULTMASK, and BASEPRI registers are used for exception or interrupt masking.

- The PRIMASK register is a 1-bit wide interrupt mask register.

When set, it blocks all exceptions (including interrupts) apart from the Non-Maskable Interrupt (NMI) and the HardFault exception.

Interrupt Mask Register

- The FAULTMASK register is very similar to PRIMASK, but it also blocks the HardFault exception

- Priority level “-1”!

Interrupt Mask Register

- BASEPRI masks exceptions or interrupts based on priority level.

- Most Cortex-M3 or Cortex-M4 microcontrollers have

8 programmable exception priority levels or
16 levels.

BASEPRI is 3 bits or 4 bits, respectively.

Control Register

- CONTROL register defines:
  - The selection of stack pointer (Main Stack Point/Process Stack Pointer)
  - Access level in Thread mode (Privileged/Unprivileged)

		31:3	2	1	0
Cortex-M3 Cortex-M4	CONTROL			SPSEL	nPRIV

		31:3	2	1	0
Cortex-M4 with FPU	CONTROL		FPCA	SPSEL	nPRIV

Control Register

- nPRIV (bit 0): the privileged level in Thread mode

- SPSEL (bit 1): the Stack Pointer selection
  - 0 : Main Stack Pointer
  - 1 : Process Stack Pointer

- FPCA (bit 2): Floating Point Context Active

Floating Point Registers

- The Cortex-M4 processor has an optional floating point unit.

- This provides additional registers for
  - floating point data processing,
  - a Floating Point Status, and
  - Control Register (FPSCR)

Behavior of the Application Program Status Register (APSR)

- Status flags for integer operations (N-Z-C-V bits)

- Status flags for saturation arithmetic (Q bit)

- Status flags for SIMD operations (GE bits)

Integer Status Flags

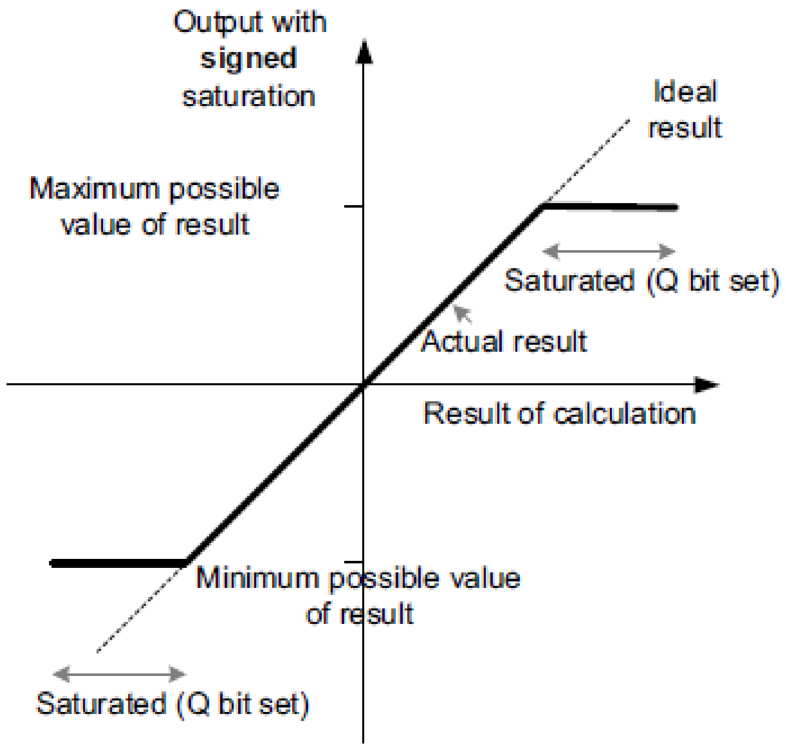
- Being affected by general data processing instructions
- Controlling conditional branches and conditional executions
  - N (bit 31) – 1: negative result
  - Z (bit 30) – 1: zero result
  - C (bit 29) – Carry flag. 1: unsigned overflow occurred
  - V (bit 28) – 1: signed overflow occurred

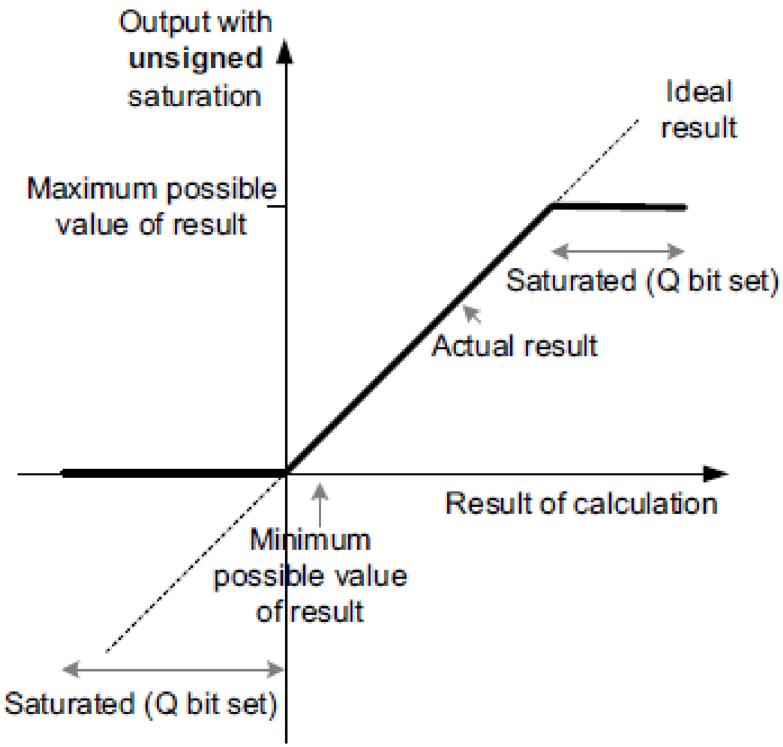
- Carry bit of APSR:

```
Z[31:0] = X[31:0] + Y[31:0];
Z[63:32] = X[63:32] + Y[63:32] + Carry;
```

Q Status Flag

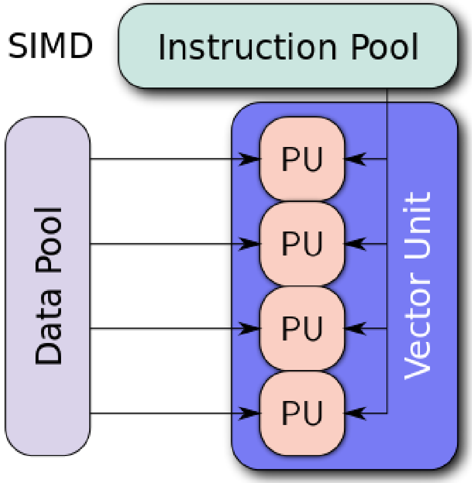
- The Q is used to indicate an occurrence of saturation


Output with **signed** saturation
Ideal result
Maximum possible value of result
Saturated (Q bit set)
Actual result
Result of calculation
Minimum possible value of result
Saturated (Q bit set)


Output with **unsigned** saturation
Ideal result
Maximum possible value of result
Saturated (Q bit set)
Actual result
Result of calculation
Minimum possible value of result
Saturated (Q bit set)

Status Flags for SIMD Operations (GE bits)

Single instruction, multiple data (SIMD) describes computers with multiple processing elements that perform the same operation on multiple data points simultaneously.



Greater-Equal (GE) bits

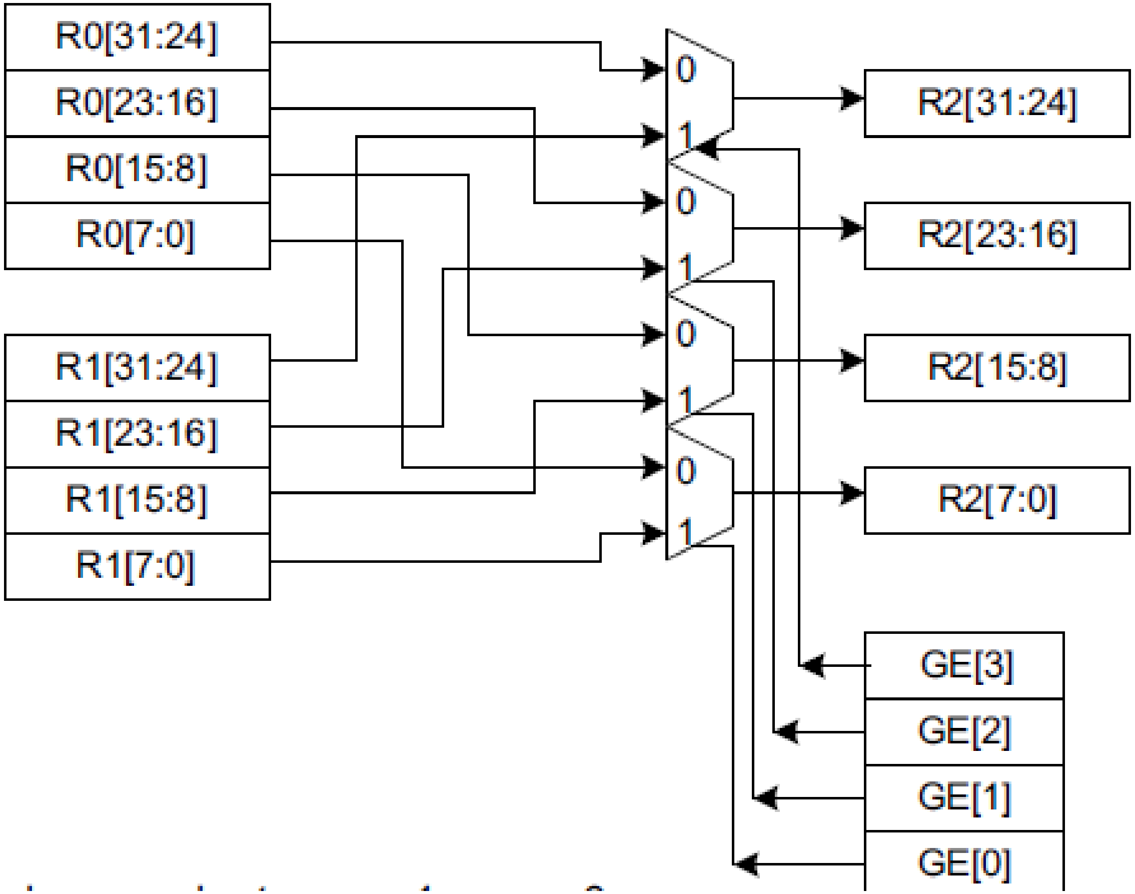
- 4-bit wide field in the APSR in the Cortex-M4

- It is updated by a number of SIMD instructions

- Each bit represents positive or overflow of SIMD operations for each byte

Greater-Equal (GE) bits

- Used in SEL instruction


R0[31:24]
R0[23:16]
R0[15:8]
R0[7:0]
R1[31:24]
R1[23:16]
R1[15:8]
R1[7:0]
0
1
0
1
0
1
0
1
R2[31:24]
R2[23:16]
R2[15:8]
R2[7:0]
GE[3]
GE[2]
GE[1]
GE[0]