ARM8 Processor Architecture Advanced

Humayun Kabir
Professor, CS, Vancouver Island University, BC, Canada

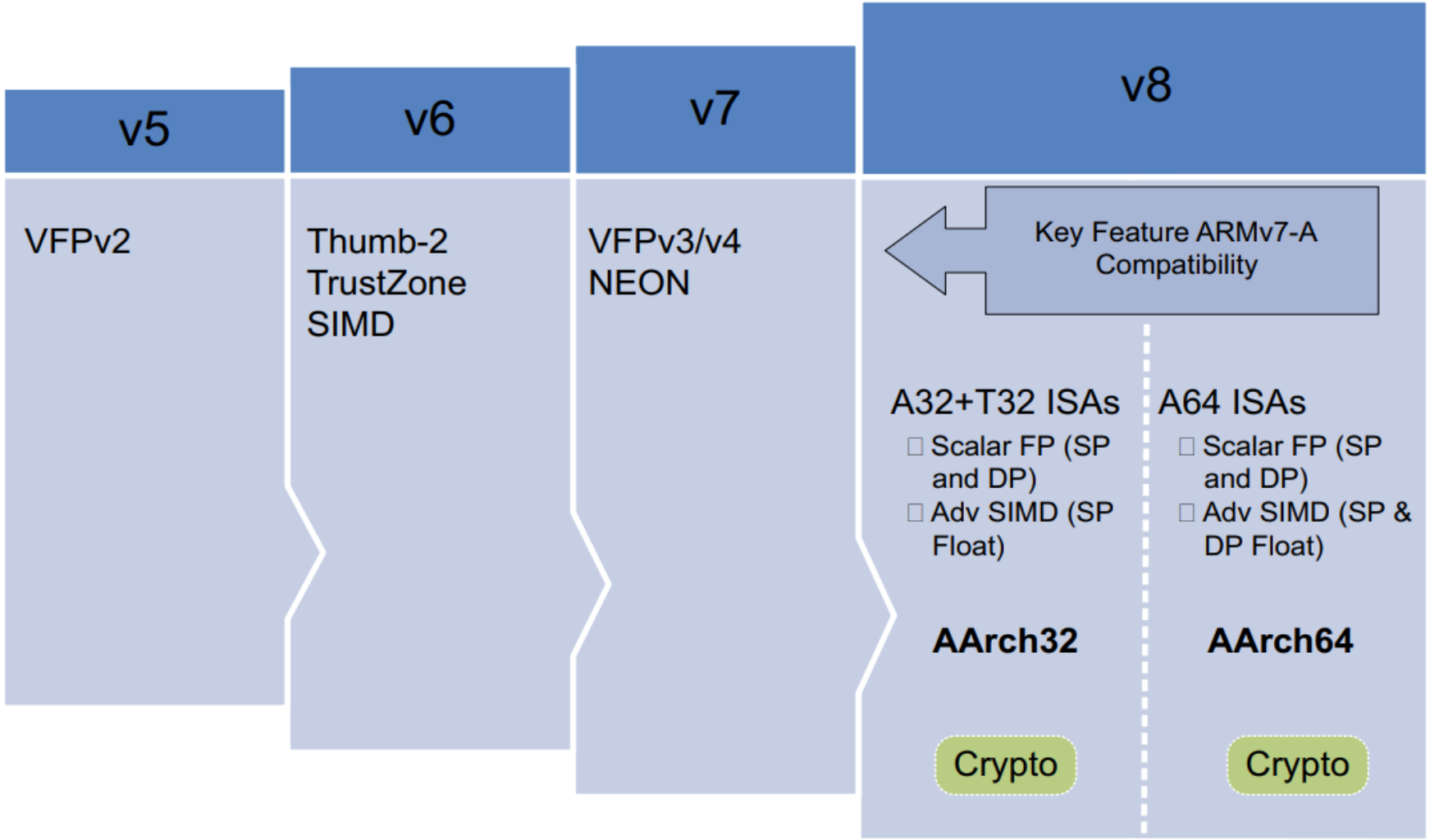
ARM8 Processor Architecture Continue: Outline

- ARM Processor Architecture
- ARMv8 Exception and Privilege Levels
- ARMv8 Security States
- ARMv8 Execution States
- ARMv8 AArch64 Special Registers

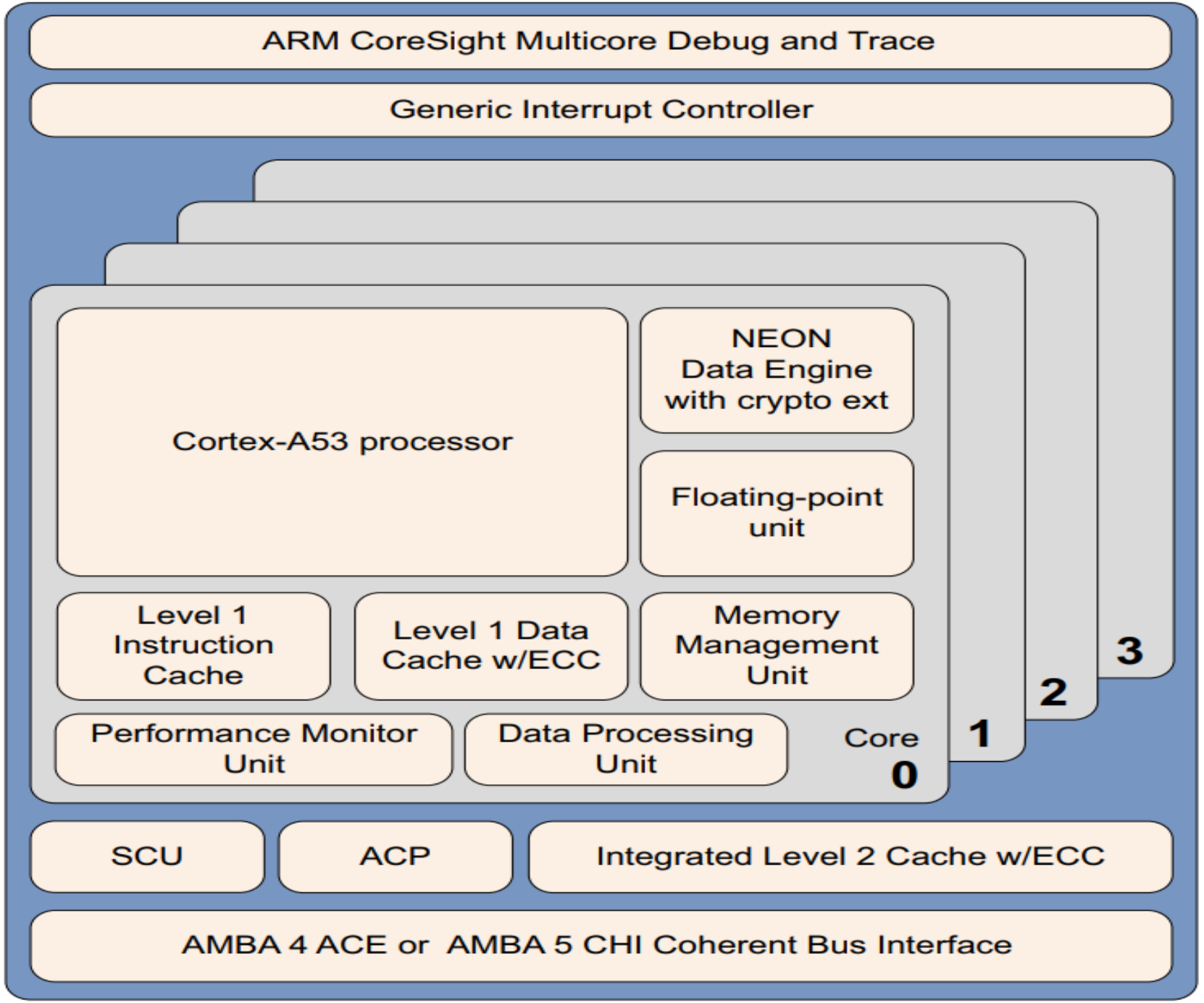
ARM Processor Architecture

- ARM, primarily a RISC system with the following attributes:
  - Moderate array of uniform registers
  - A load/store model of data processing in which operations only perform on operands in registers and not directly in memory
  - A uniform fixed-length instruction of 32 bits
  - Separate arithmetic logic unit (ALU), Multiply, and Shifter units
  - A small number of addressing modes with all load/store addresses determined from registers and instruction fields
  - Auto-increment and auto-decrement addressing modes are used to improve the operation of program loops

Development of the ARMv8 architecture



ARMv8 (Cortex-A53) Processor

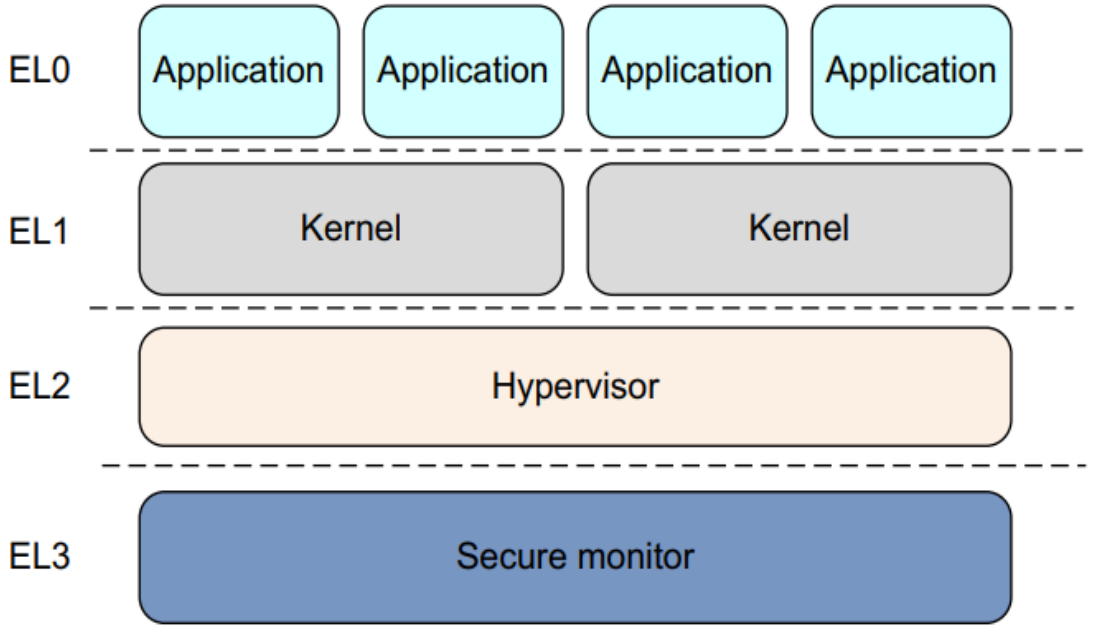


ARMv8 Exception and Privilege Levels

- Execution occurs in **4 Exception Levels**: **EL0**, **EL1**, **EL2**, and **EL3**.
- Execution in exception level **ELn** usually corresponds to privilege level **PLn**.
- The **higher the number** in **ELn/PLn** the **higher the privilege**.
- Multiple exception/privilege levels support the concept of hierarchical protection domains.

ARMv8 Exception Levels

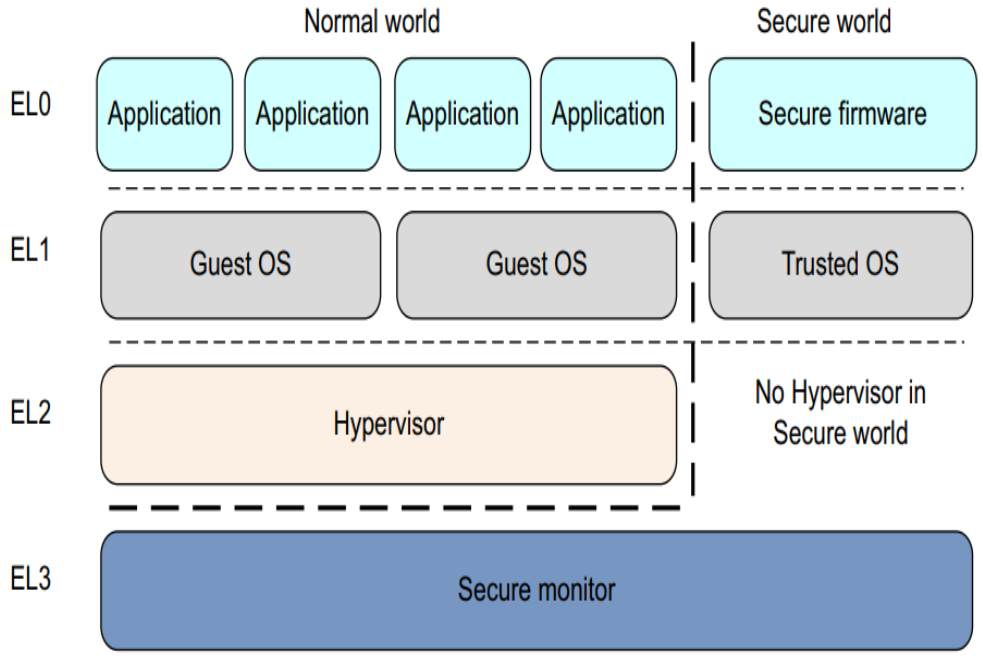
- Typical example of what software runs at each exception level:
  - EL0 Normal user applications, not privileged.
  - EL1 Operating system kernel, privileged.
  - EL2 Hypervisor, more privileged.
  - EL3 Low-level firmware, including the Secure Monitor, highly privileged



ARMv8 Security States

- Provides two **security states**: non secured state (**normal world**) and secured state (**secured world**)

- Enables **Guest OS** in **normal world** runs in parallel with a **Trusted OS** in **secured world** on the same hardware.
- Enables a **Hypervisor** running in **normal world** to host multiple **Guest OS** on the same hardware.
- **Secure Monitor** acts as a **gateway** between **normal** and **secured** world.



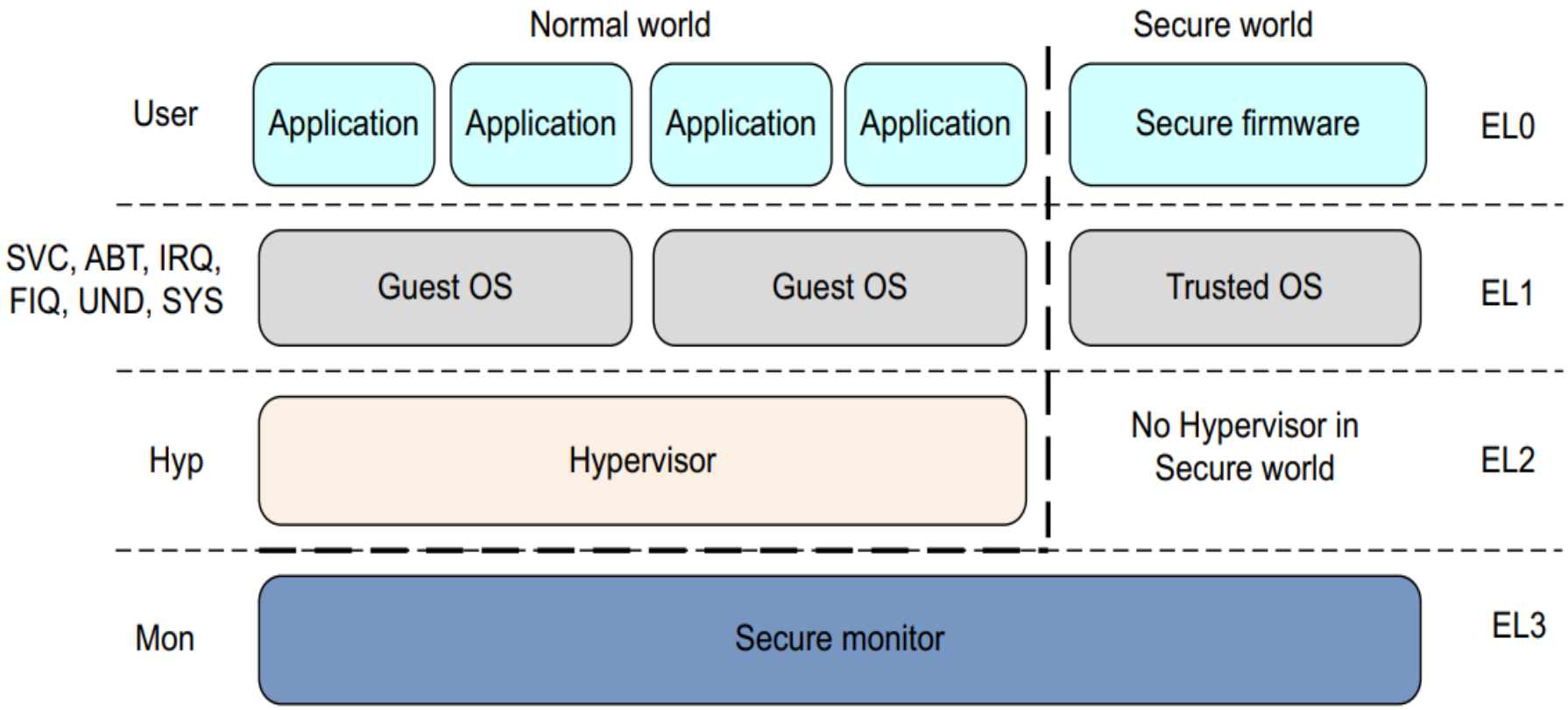
ARMv8 Processor Execution States

- An ARMv8 processor can be **configured** as executing in one of two **execution states**: **AArch32** (32 bits) and **AArch64** (64 bits).

- **AArch32**: **32-bit instructions**, **data**, and **address space**. Uses the **instruction set**, the **register set**, and the **processor modes** like **ARMv7**.

- **AArch64**: **32-bit instructions** but **64-bit data** and **address space**. Uses **enhanced instruction set**, a **larger register set** with **wider registers**, and the **processor modes** that are different from **ARMv7**.

AArch64 Exception Modes



AArch64 Exception Modes

Mode	Function	Security state	Privilege level
User (USR)	Unprivileged mode in which most applications run	Both	PL0
FIQ	Entered on an FIQ interrupt exception	Both	PL1
IRQ	Entered on an IRQ interrupt exception	Both	PL1
Supervisor (SVC)	Entered on reset or when a Supervisor Call instruction (SVC) is executed	Both	PL1
Monitor (MON)	Entered when the SMC instruction (Secure Monitor Call) is executed or when the processor takes an exception which is configured for secure handling. Provided to support switching between Secure and Non-secure states.	Secure only	PL3
Abort (ABT)	Entered on a memory access exception	Both	PL1
Undef (UND)	Entered when an undefined instruction is executed	Both	PL1
System (SYS)	Privileged mode, sharing the register view with User mode	Both	PL1
Hyp (HYP)	Entered by the Hypervisor Call and Hyp Trap exceptions.	Non-secure only	PL2

AArch64 Exception Modes

- **USR (User mode)** – Most of the application codes usually run in this mode. It is an unprivileged level (PL0) mode.
- **SYS (System mode)** – Privileged level (PL1) and the **Operating System Kernel** codes usually run in this mode.
- **SVC (Supervisor mode)** – When an **application** (running in PL0) needs a particular **service** from an **Operating System** (running in PL1), it executes ***SVC #syscall_number*** instruction. Required service is specified by *#syscall_number*. This execution of SVC instruction allows the application to enter in Supervisor mode and to get the specified service. Once the service is received, the application defaults to user mode.

AArch64 Exception Modes

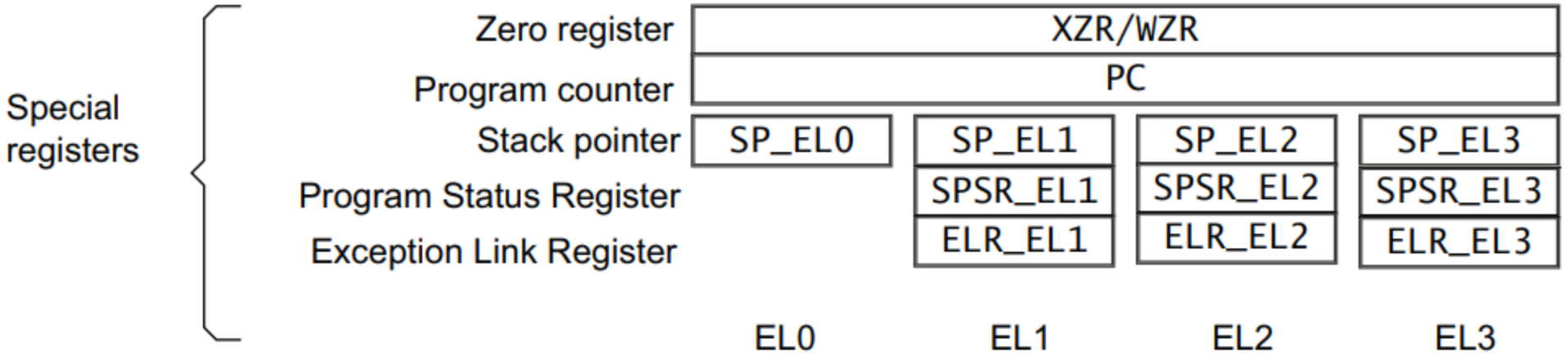
- **HYP (Hypervisor mode)** – When a **guest operating system** (running in PL1) needs a particular **service** from a **hypervisor** (running in PL2), it executes ***HVC #hvcservice_number*** instruction. Required service is specified by *#hvcservice_number*. This execution of HVC instruction allows the OS to enter in Hypervisor mode and to get the specified service. Once the service is received, the OS defaults to System mode.

- **MON (Secured Monitor mode)** – When a **hypervisor**(running in PL2) needs a particular **service** from a **secured monitor** (running in PL3), it executes ***SMC #smcservice_number*** instruction. Required service is specified by *#smcservice_number*. This execution of SMC instruction allows the hypervisor to enter in Secure Monitor mode and to get the specified service. Once the service is received, the hypervisor defaults to hypervisor mode.

AArch64 Exception Modes

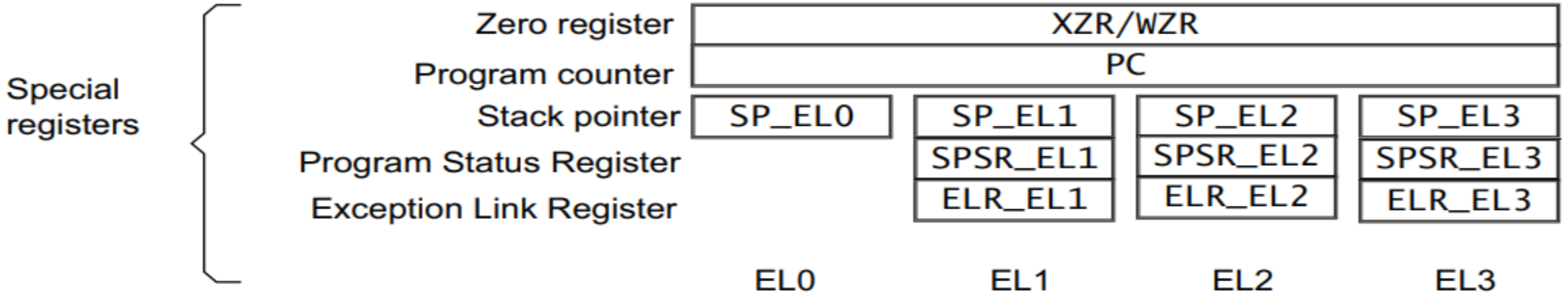
- **IRQ (Interrupt Request mode)** – Entered in to provide a service against an **Interrupt Request** asserted by an external hardware. Service routine codes run in privilege level (PL1). Once the service is provided, it falls back to the interrupted mode.
- **FIQ (Fast Interrupt Request mode)** – Entered in to provide a service against a **Fast Interrupt Request** asserted by an external hardware. Service routine codes run in privilege level (PL1). Once the service is provided, it falls back to the interrupted mode.
- **ABT (Abort mode)** – Entered in when a **memory access exception** occurs. Abort exception service codes run in privilege level (PL1) and usually terminate the **application** that causes the exception.
- **UND (Undefined mode)** – Entered in when an **undefined instruction** is being attempted to execute. Undefined exception service codes run in privilege level (PL1) and usually terminate the **application** that causes the exception.

AArch64 Special Registers: XZR and PC



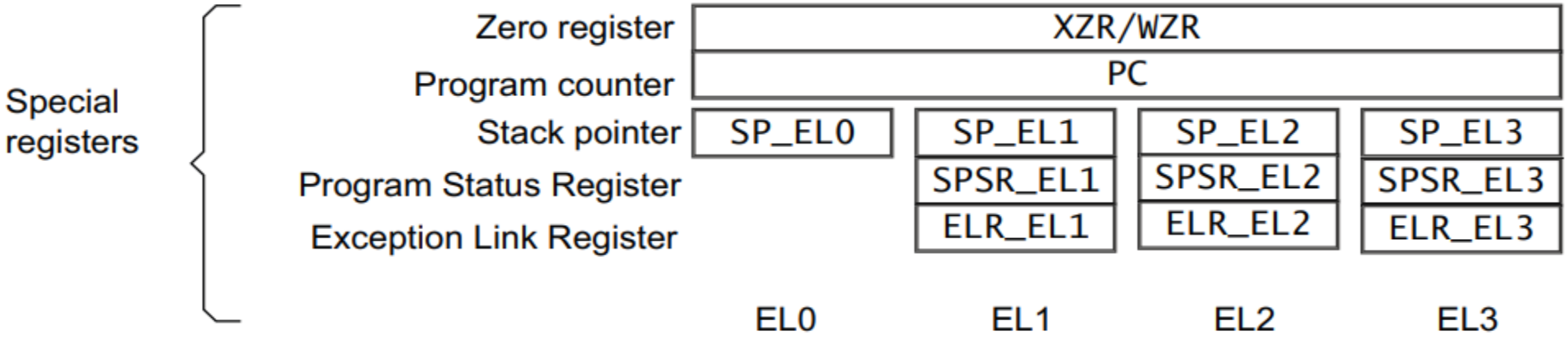
- There is **no actual register** numbered x31. **Reading** from **xzr** in code reads **64 bit zeros** and **writing** into **xzr** does not do anything.
- **PC** is not a general purpose register and is **not accessible** in code.

AArch64 Special Registers: Stack Pointer



- There are dedicated **stack pointers** (**SP_EL0**, **SP_EL1**, **SP_EL2**, and **SP_EL3**) for each exception level.
- **SP** refers to the **current stack pointer**.
- Program running in exception levels **EL1**, **EL2**, and **EL3** can use either the dedicated stack pointer or **EL0** stack pointer.
  - Suffix **t** (SP_ELn*t*) in the name indicates SP_EL0 has bee selected.
  - Suffix **h** (SP_ELn*h*) in the name indicates SP_Eln has been selected.

AArch64 Special Registers: Linked Register



- There are dedicated **linked register** (**ELR_EL1**, **ELR_EL2**, and **ELR_EL3**) for exception levels **EL1**, **EL2**, and **EL3**.
- Before taking into an exception level **return address** is saved into that level's **ELR**.
- The saved return address is used from the level's ELR at the return from the exception.

AArch64 PSTATE Fields

- The **processor state** or **PSTATE Fields** in **AArch64** are **independently accessible** in **groups** through some special purpose registers.

NZCV	Holds the condition flags
DAIF	Specifies the current interrupt mask bits.
CurrentEL	Holds the current Exception level.
SPSel	At EL1 or higher, this selects between the SP for the current Exception level (SP_Eln) and SP_EL0

The **PSTATE.{N**, **Z**, **C**, **V}** fields can be accessed at **EL0**. All other PSTATE fields can be executed at **EL1** or higher and are UNDEFINED at EL0.

AArch64 PSTATE Fields

Name	Description
N	Negative Condition flag
Z	Zero Condition flag
C	Carry Condition flag
V	Overflow Condition flag
SS	Software Step bit
IL	IL Execution state bit
D	Debug Mask bit
A	SError mask bit
I	IRQ mask bit
F	FIQ mask bit
nRW	Execution Mode (0=64-bit, 1=32-bit)
EL(2)	Exception Level (00, 01, 10, 11)
SP	SP Selector (0= SP_EL0, 1= SP_ELn)

AArch64 PSTATE Fields

- **PSTATE** field groups are read directly using **Move** to **Register** from **System** (**MRS)** instruction.

```
MRS <Xt>, NZCV          // bits [31:28]
MRS <Xt>, DAIF          // bits [9:6]
MRS <Xt>, CurrentEL     // bits [3:2]
MRS <Xt>, SPSel         // bit [0]
```

AArch64 PSTATE Fields

- **PSTATE** field groups are written directly using **Move** to **System** from **Register** (**MSR**) instruction.

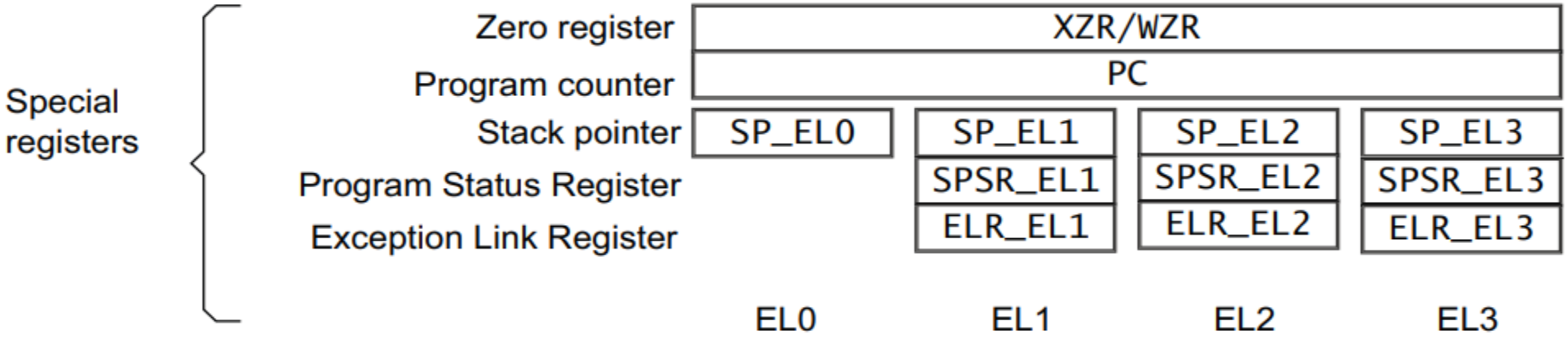
  MSR NZCV, <Xt>

  MSR DAIF, <Xt>

  MSR CurrentEL, <Xt>

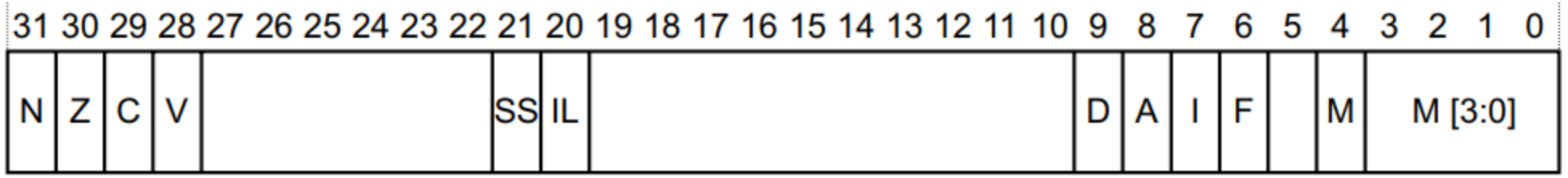
  MSR SPSel, <Xt>

AArch64 Special Registers: SPSR



- There are dedicated **saved program status register** (**SPSR_EL1**, **SPSR_EL2**, and **SPSR_EL3**) for exception levels **EL1**, **EL2**, and **EL3**.
- Before taking into an exception level **current processor state (PSTATE)** is saved into that level's **SPSR**.
- The saved PSTATE is restored from the level's SPSR at the return from the exception.

AArch64 Special Registers: SPSR



N Negative result (N flag).

Z Zero result (Z) flag.

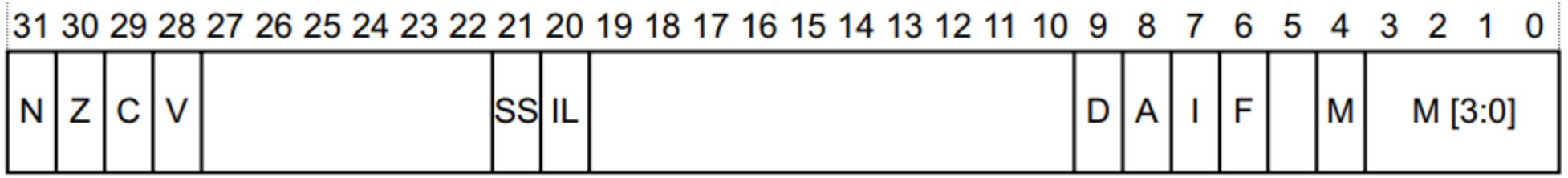
C Carry out (C flag).

V Overflow (V flag).

SS Software Step. Indicates whether software step was enabled when an exception was taken.

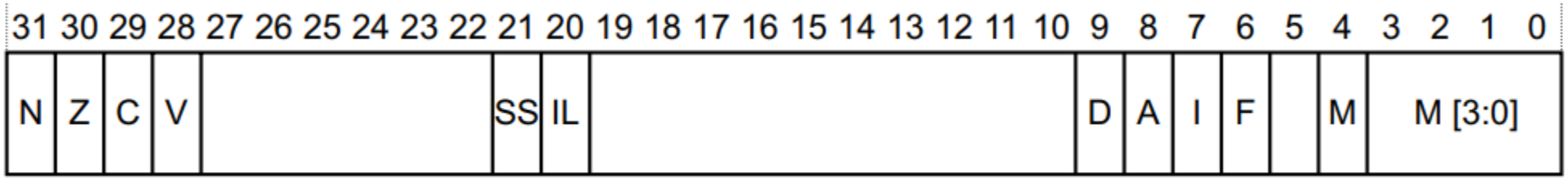
IL Illegal Execution State bit. Shows the value of PSTATE.IL immediately before the exception was taken.

AArch64 Special Registers: SPSR



D	Process state Debug mask. Indicates whether debug exceptions from watchpoint, breakpoint, and software step debug events that are targeted at the Exception level the exception occurred in were masked or not.
A	SError (System Error) mask bit.
I	IRQ mask bit.
F	FIQ mask bit.
M[4]	Execution state that the exception was taken from. A value of 0 indicates AArch64.
M[3:0]	Mode or Exception level that an exception was taken from.

AArch64 Special Registers: SPSR



M[3:0]	Meaning
0b0000	EL0.
0b0100	EL1 with SP_EL0 (ELt).
0b0101	EL1 with SP_EL1 (EL1h).
0b1000	EL2 with SP_EL0 (EL2t).
0b1001	EL2 with SP_EL2 (EL2h).