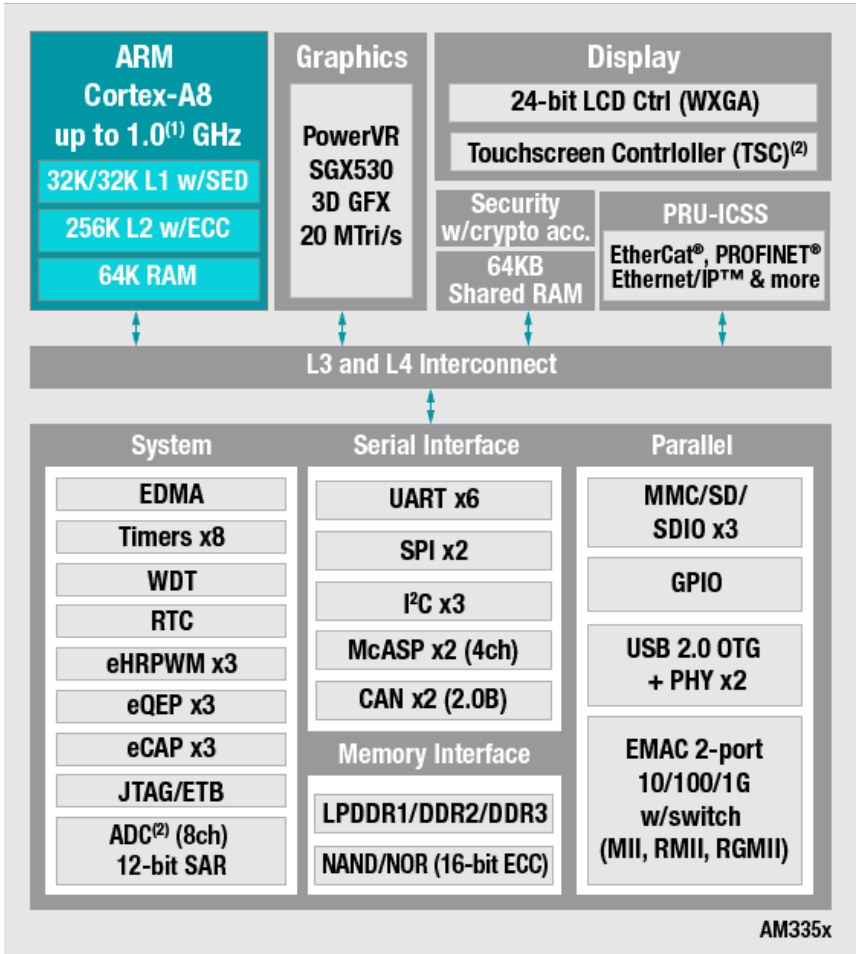
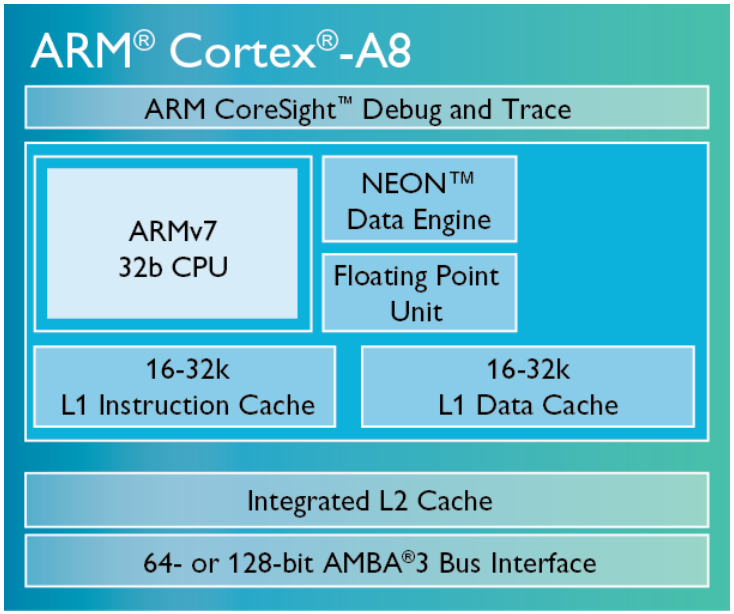
EEEN301 Lecture 2
Caching, JTAG, AMBA, AXI and ARM architecture

System on a chip (SOC)





Texas Instruments

ARM®

Cache

A Cache is a small block of fast memory that is placed between the CPU and the main memory. Cache memory is often static RAM, is located close to the processor or even on the same chip. Therefore it has significantly faster access times than the main DRAM system memory.

The cache memory is used to reduce the average memory access times. This is done by storing a local copy of data/instructions that are expected or are frequently accessed in main memory addresses therefore allowing the CPU to access the data faster.

There are different types of cache usually defined in a hierarchical manor (e.g. L1,L2 and L3). The main system memory itself can be considered as a cache as it stores a copy of the program/data that resides on the hard-drive.

The steps to access the data from cache memory are:

1. A data or instruction request is made by the CPU. (addressed)
2. The data/instruction is retrieved from cache if it is stored there. (called a cache hit)
3. If the required data/instruction is not in the cache (cache miss) then the data/instruction will need to accessed from the main memory. The cache controller will then start copying from the main memory.

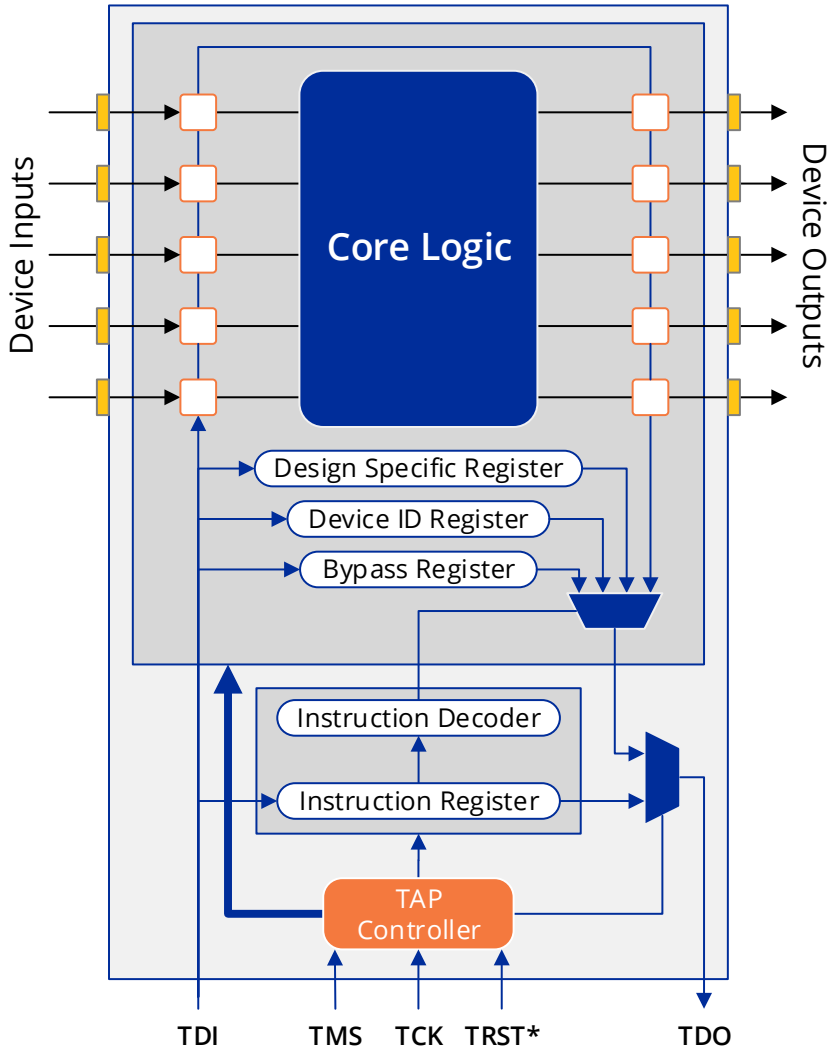
This method relies on the fact that program data or instructions typically reside in adjacent memory locations and are accessed in increasing address order.

JTAG Joint Test Action Group

Originally developed for boundary scan testing of VLSI devices. Later expanded to include debug and in system programming capabilities.

A special JTAG port and extra logic is included in the manufactured VLSI device, microprocessor or FPGA.

A JTAG connector is then placed on the circuit board to allow factory testing or programming. Most hardware development environments include a JTAG connect or in system programming and debugging. JTAG is a synchronous serial interface like SPI.

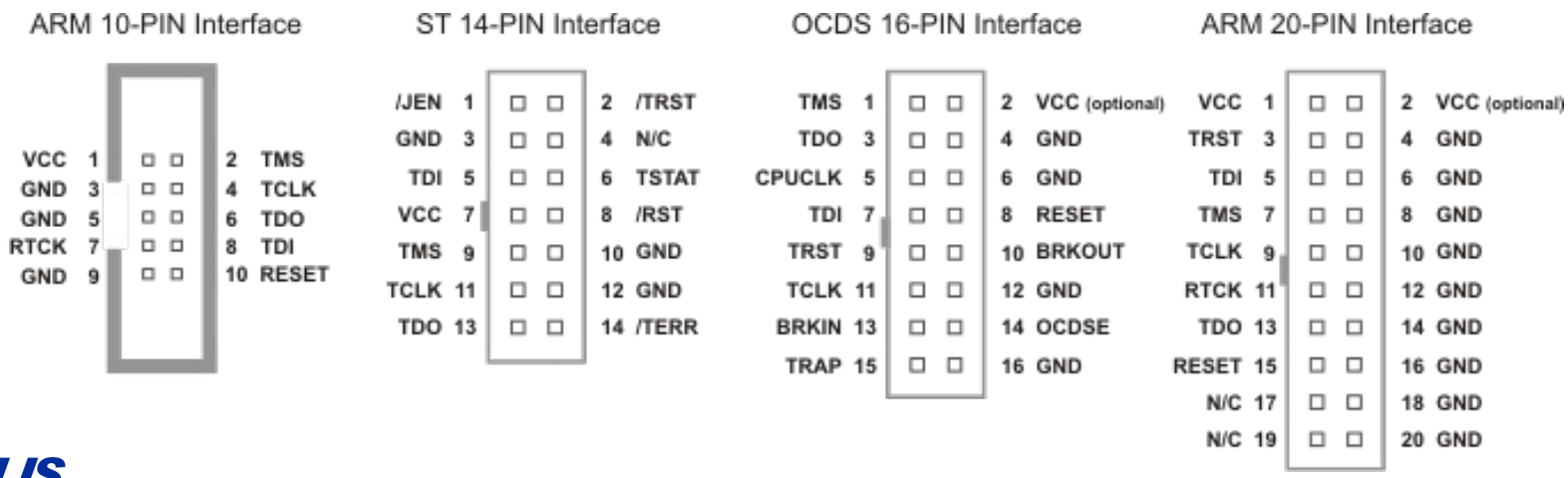


CORELIS An EWA Company

JTAG Interface

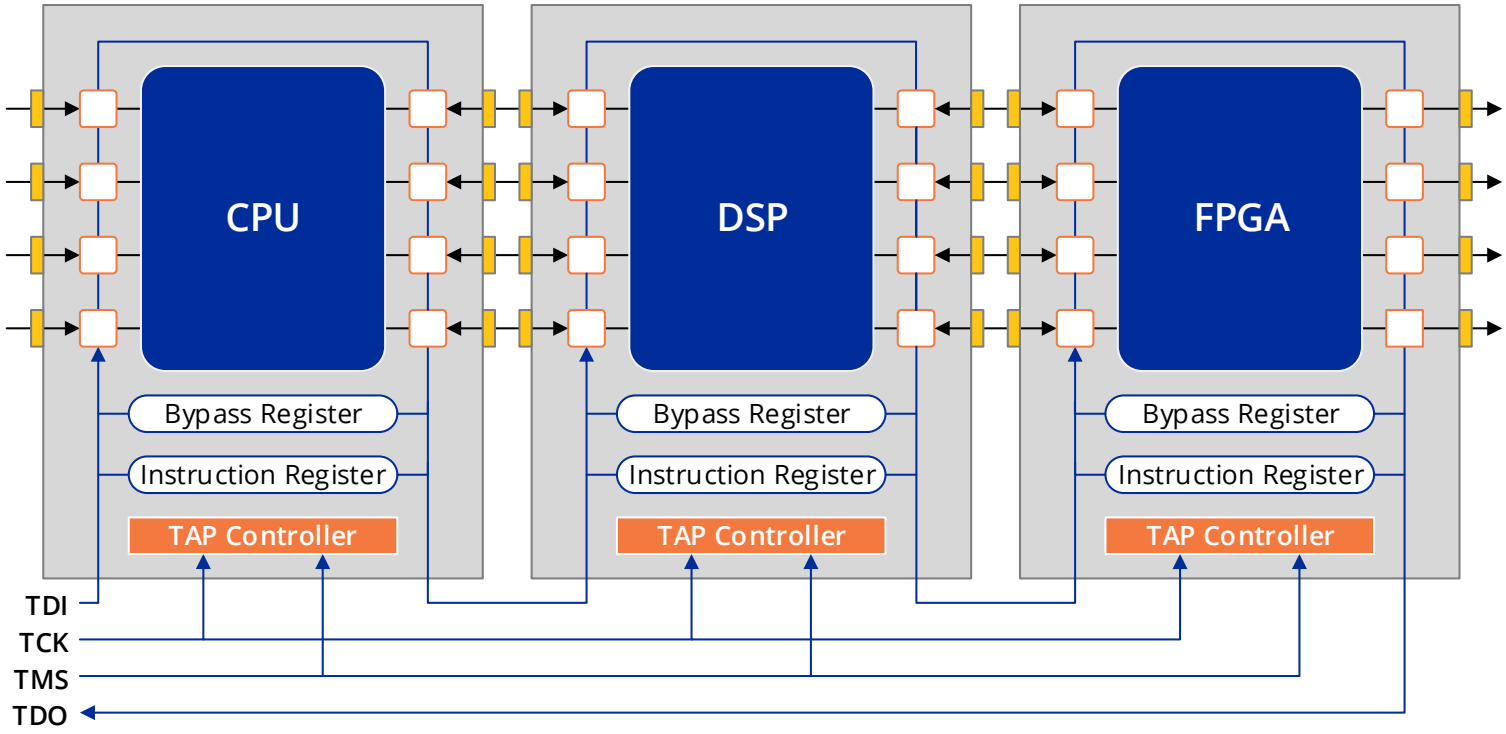
The physical JTAG interface, or test access port (TAP) consists of four mandatory signals and one optional asynchronous reset signal. Table 1 below summarizes the JTAG TAP signals.

Abbreviation	Signal	Description
TCK	Test Clock	Synchronizes the internal state machine operations.
TMS	Test Mode Select	Sampled at the rising edge of TCK to determine the next state.
TDI	Test Data In	Represents the data shifted into the device's test or programming logic. It is sampled at the rising edge of TCK when the internal state machine is in the correct state.
TDO	Test Data Out	Represents the data shifted out of the device's test or programming logic and is valid on the falling edge of TCK when the internal state machine is in the correct state.
TRST	Test Reset	An optional pin which, when available, can reset the TAP controller's state machine.



Scan Chain

JTAG devices may be daisy-chained within a system and controlled simultaneously. Boundary-scan test software can utilize one component to drive signals that will be sensed on a second component, verifying continuity from pin-to-pin. Devices can be placed in BYPASS mode to shorten the overall length of the chain to reduce test time. More complex designs may utilize additional circuitry or a dedicated JTAG bridge to selectively configure a scan chain that contains multiple devices, or even multiple sub-assemblies.



CORELIS An EWA Company

In-System-Programming

In addition to test applications, JTAG is also frequently used as the primary method to program devices such as flash memory and CPLDs. To program flash devices, the pins of a connected boundary-scan-compatible component can be used to control the memory and erase, program, and verify the component using the boundary-scan chain. FPGA and CPLD devices that support IEEE-1532 standard instructions can be accessed and programmed directly using the JTAG port.

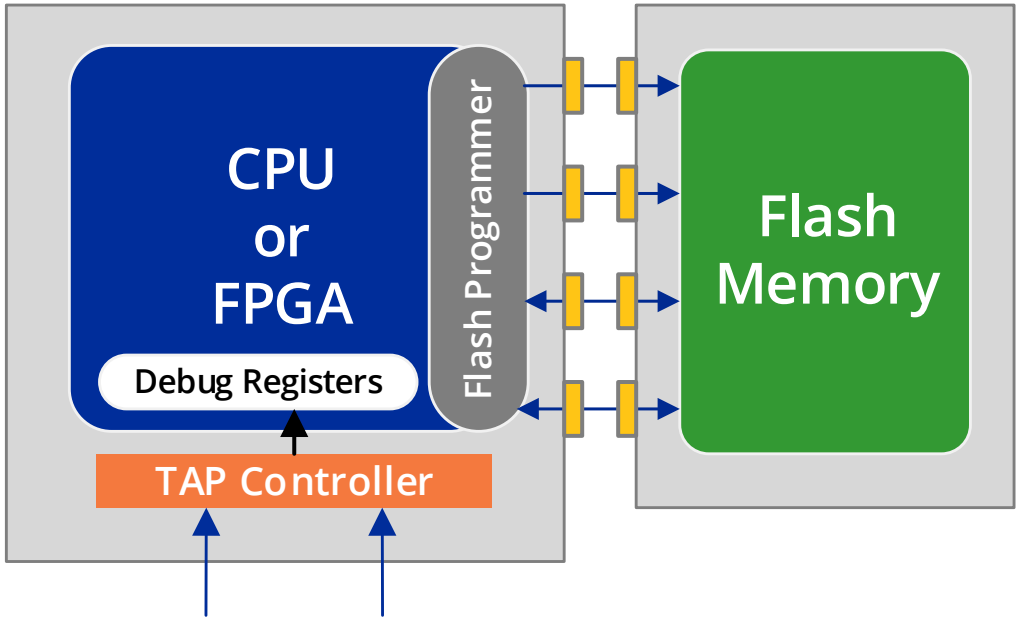
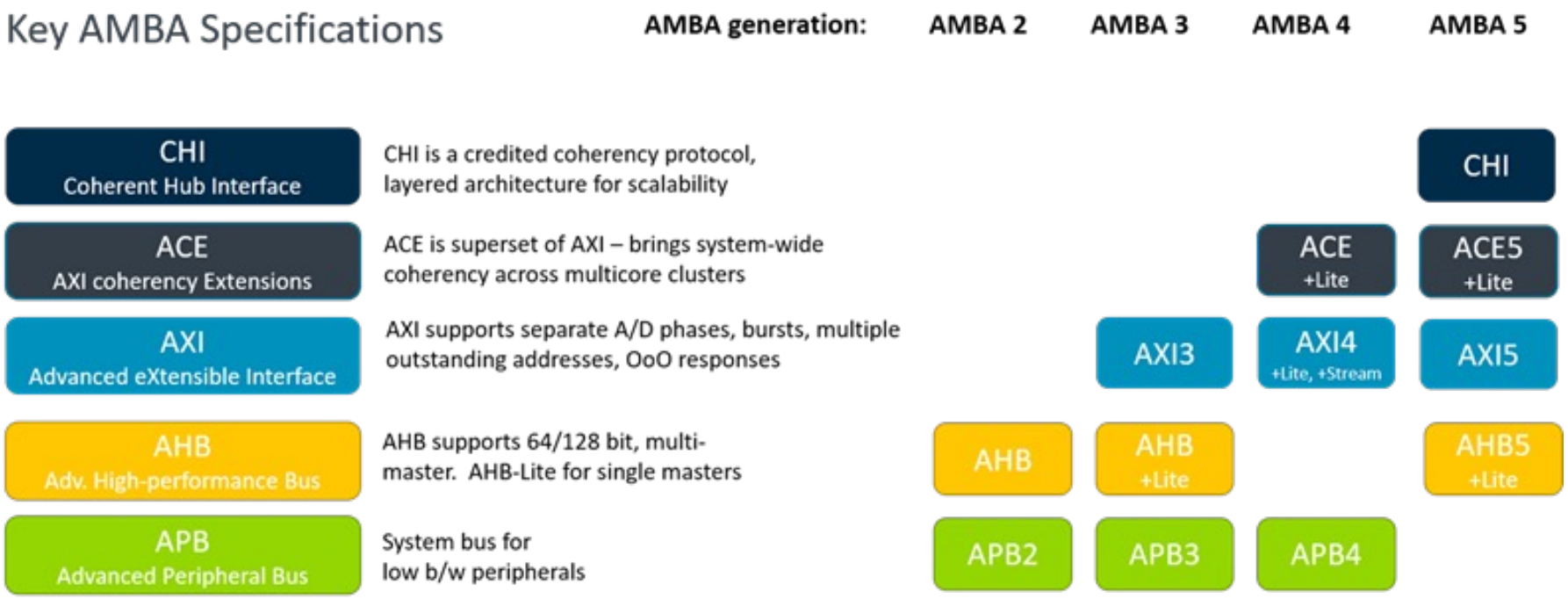


Figure 8. A CPU or FPGA under JTAG control can be used to program flash memory.

AMBA (Advanced Microcontroller Bus Architecture) is a freely-available, open standard for the connection and management of functional blocks in a system-on-chip (SoC). It facilitates right-first-time development of multi-processor designs, with large numbers of controllers and peripherals.

AMBA specifications are royalty-free, platform-independent and can be used with any processor architecture. Due to its widespread adoption, AMBA has a robust ecosystem of partners that ensures compatibility and scalability between IP components from different design teams and vendors.

Key AMBA Specifications



arm Developer

What is AXI?

AXI is part of ARM AMBA, a family of micro controller buses first introduced in 1996. The first version of AXI was first included in AMBA 3.0, released in 2003. AMBA 4.0, released in 2010, includes the second major version of AXI, AXI4.

There are three types of AXI4 interfaces:

- **AXI4**: For high-performance memory-mapped requirements.
- **AXI4-Lite**: For simple, low-throughput memory-mapped communication (for example, to and from control and status registers).
- **AXI4-Stream**: For high-speed streaming data.

Both AXI4 and AXI4-Lite interfaces consist of five different channels:

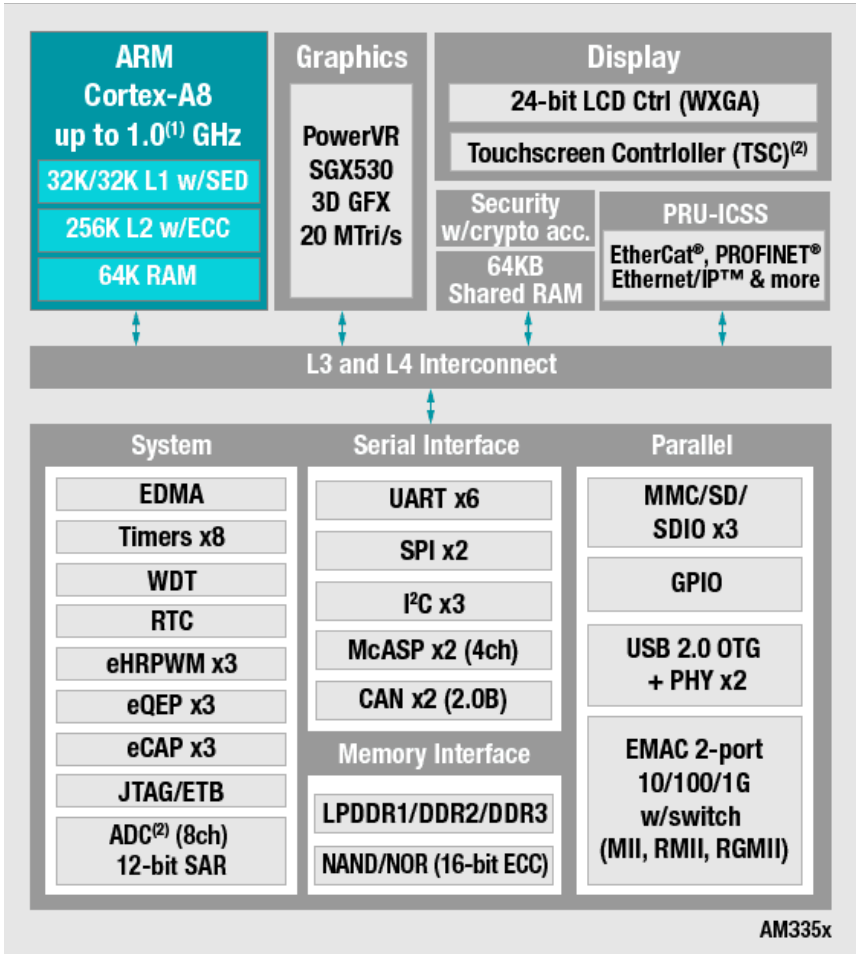
- Read Address Channel
- Write Address Channel
- Read Data Channel
- Write Data Channel
- Write Response Channel

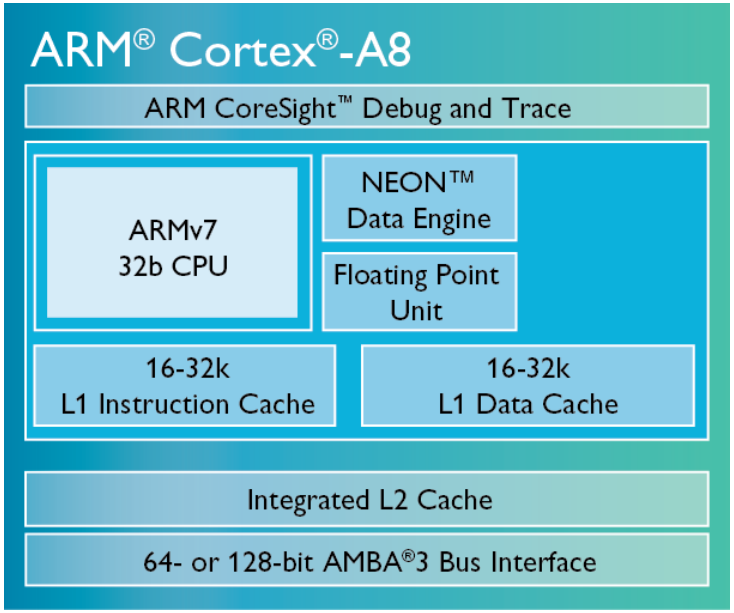
Data can move in both directions between the master and slave simultaneously, and data transfer sizes can vary. The limit in AXI4 is a burst transaction of up to 256 data transfers. AXI4-Lite allows only one data transfer per transaction.

The AXI4-Stream protocol defines a single channel for transmission of streaming data.

Back to the start

System on a chip (SOC)

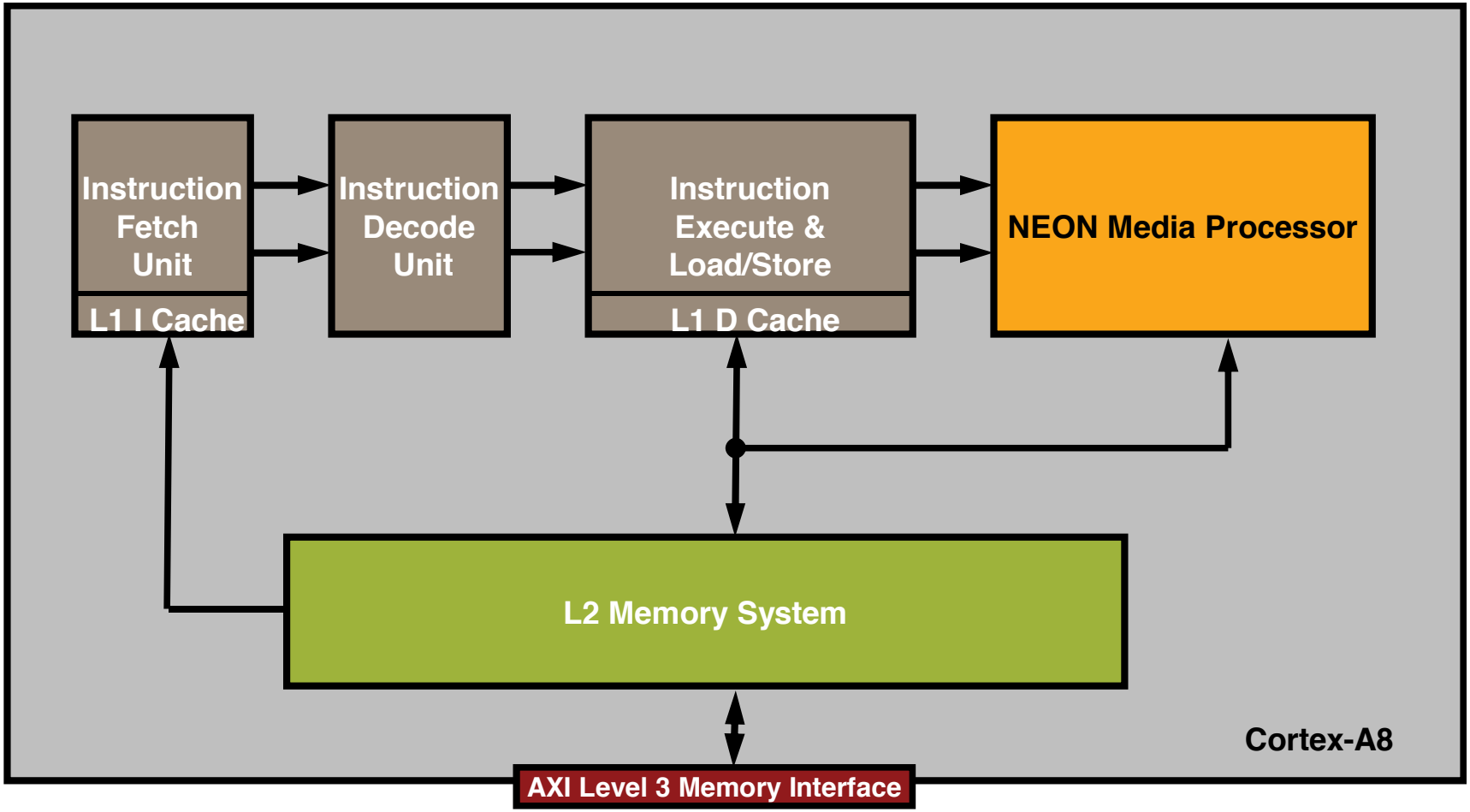




Texas Instruments

ARM®

Cortex-A8 Block Diagram

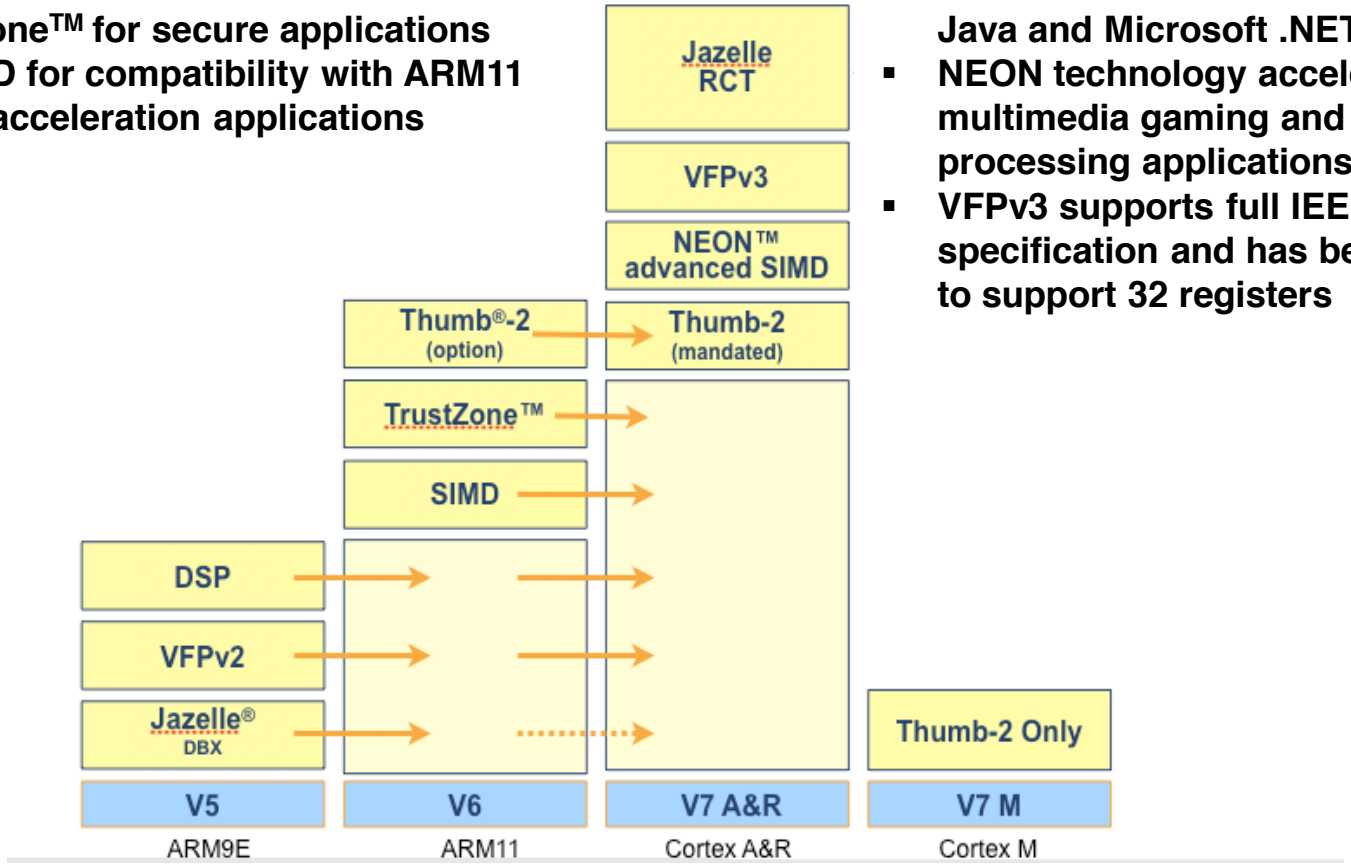


The Architecture for the Digital World®
ARM®

ARM Cortex-A Architecture

Cortex A Base Architecture

- Thumb-2 technology for power efficient execution
- TrustZone™ for secure applications
- v6 SIMD for compatibility with ARM11
- media acceleration applications



Cortex-A8 Extensions

- Jazelle-RCT for efficient acceleration of execution environments such as Java and Microsoft .NET
- NEON technology accelerating multimedia gaming and signal processing applications
- VFPv3 supports full IEEE 754 specification and has been expanded to support 32 registers

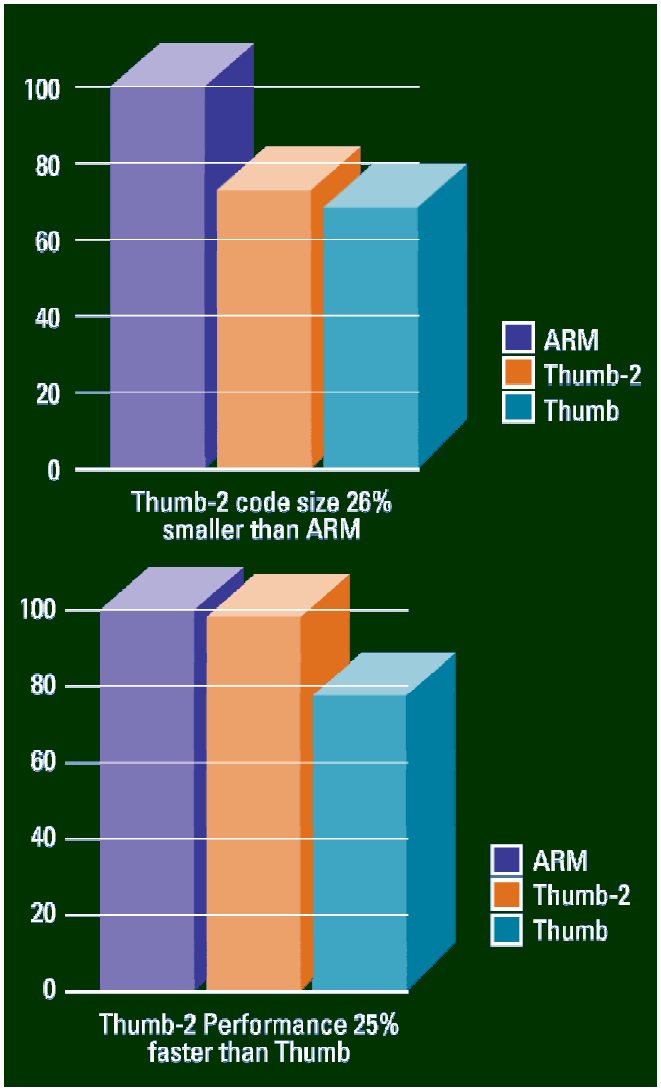
The Architecture for the Digital World®
ARM®

Data Sizes and Instruction Sets

- The ARM is a 32-bit architecture.
- When used in relation to the ARM:
  - **Byte** means 8 bits
  - **Halfword** means 16 bits (two bytes)
  - **Word** means 32 bits (four bytes)
- Most ARM's implement two instruction sets
  - 32-bit ARM Instruction Set
  - 16-bit Thumb Instruction Set
- Jazelle cores can also execute Java bytecode

The Thumb-2 instruction set

- Variable-length instructions
  - ARM instructions are a fixed length of 32 bits
  - Thumb instructions are a fixed length of 16 bits
  - Thumb-2 instructions can be either 16-bit or 32-bit
- Thumb-2 gives approximately 26% improvement in code density over ARM
- Thumb-2 gives approximately 25% improvement in performance over Thumb



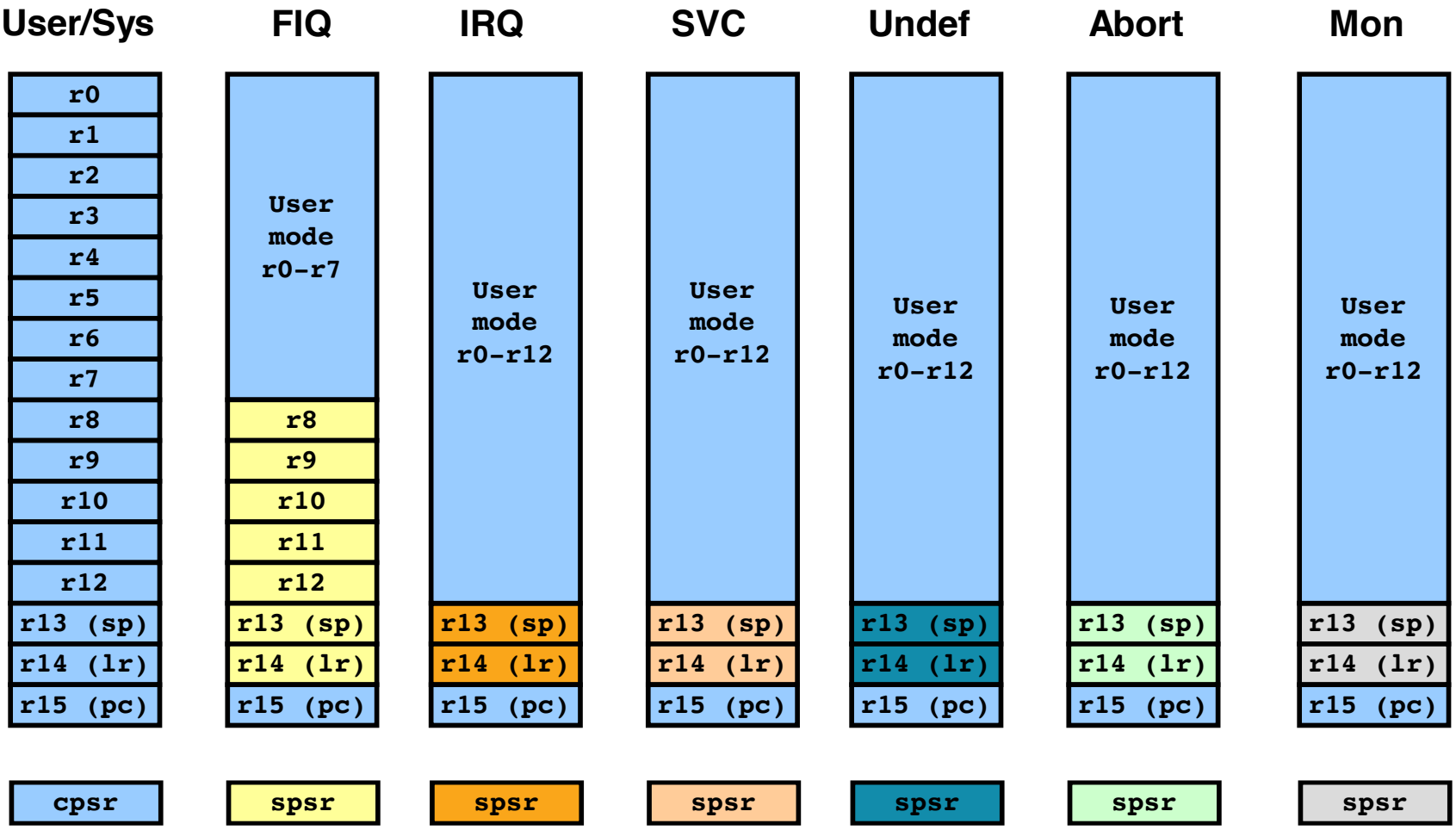
The Architecture for the Digital World®
ARM®

Cortex-A8 Processor Modes

- User - used for executing most application programs
- FIQ - used for handling fast interrupts
- IRQ - used for general-purpose interrupt handling
- Supervisor - a protected mode for the Operating System
- Undefined - entered upon Undefined Instruction exceptions
- Abort - entered after Data or Pre-fetch Aborts
- System - privileged user mode for the Operating System
- Monitor - a secure mode for TrustZone

The Architecture for the Digital World®
ARM®

Cortex-A8 Register File



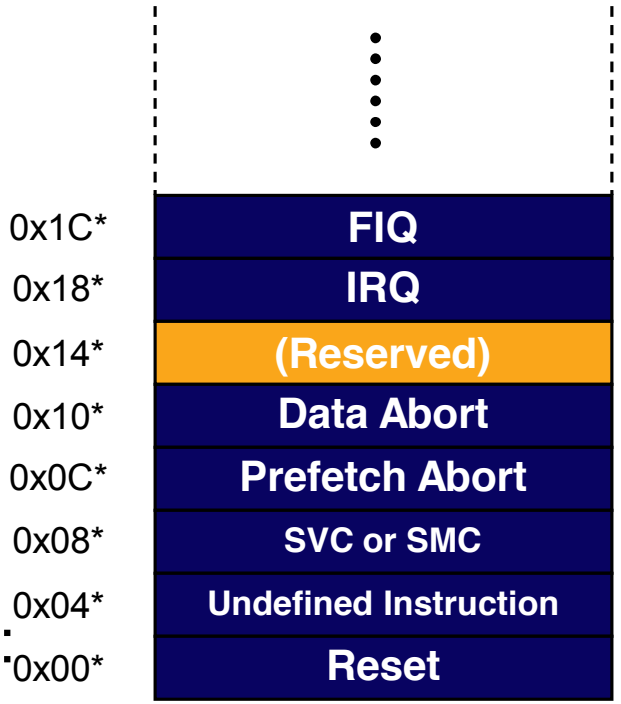
Note: System mode uses the User mode register set

The Architecture for the Digital World®
ARM®

Cortex-A8 Exception Handling

- When an exception occurs, the ARM:
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM state
    - Change to exception mode
    - Disable interrupts (if appropriate)
  - Stores the return address in LR_<mode>
  - Sets PC to vector address
- To return, exception handler needs to:
  - Restore CPSR from SPSR_<mode>
  - Restore PC from LR_<mode>

  This can only be done in ARM state.

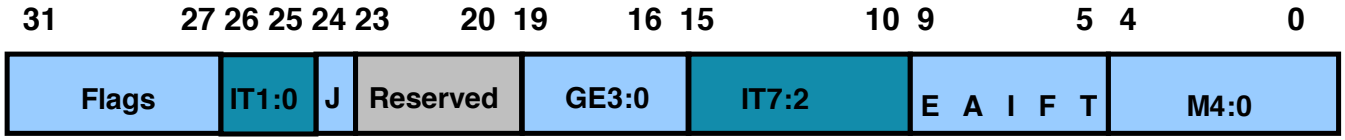


Vector Table

* Represents an offset, as vector table can moved to different base addresses

The Architecture for the Digital World®
ARM®

Cortex-A8 Program Status Register

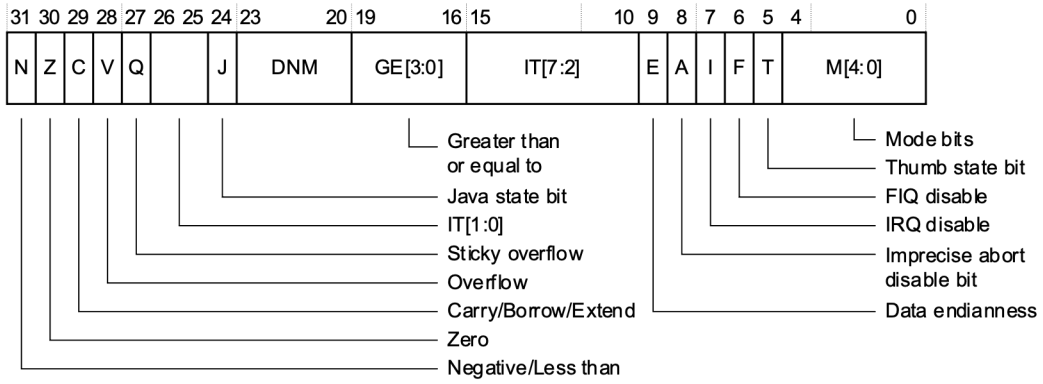


- New IT field in Program Status Registers
  - Bits 7:5 indicate base condition
  - Bits 4:0 indicate the number of instructions and condition/inverse condition
  - Updated by
    - IT, BX, BLX, BXJ instructions
    - Loads to PC (except in User mode)

- New execution state (CPSR/SPSR)

J bit	T bit	State
0	0	ARM
0	1	Thumb
1	0	Jazelle-DBX
1	1	Thumb2-EE

  - EnterX / LeaveX instructions



The Architecture for the Digital World®
ARM®